例題学習と問題演習

Java の手ほどき

演習編

問題解答・解説

■第 1 章 §1 復習問題 (p.11)

1．プログラムの入力と実行
①プログラムの保存　②コンパイル　③エラーの修正
2．プログラムの構成
①クラス　②クラス名　③クラス宣言　④修飾子　⑤メソッド名　⑥返却値の型
⑦引数　⑧ main()　⑨数字　⑩ _（アンダースコア）　⑪ $（ドル記号）　⑫ System
⑬ out　⑭ println()　⑮ System　⑯ out　⑰ print()　⑱終端子

■第 1 章 §1 練習問題 (p.12)

1．　不適当なもの　(2)　(5)　(6)　(10)　(12)

(2)　(6)　(12)　記号でクラス名として使えるのは、$ と _ だけ。
(5)　数字は冒頭には使えない。
(10)　クラス名の先頭の文字は大文字が Java の慣行

2．（略）

クラス宣言、メソッド宣言、println() 文は、今後学習するほとんどすべてのプログラムに出てくる Java の最重要の構文。キーボードから繰り返し入力するだけでなく、身体を使ってしっかり覚える。

■第 1 章 §1 実習問題 (p.12)

【文字列の表示】

(1)
```
class ClassName{
    public static void main(String[] args){
        System.out.println(" クラス名の先頭文字は大文字である。");
    }
}
```

(2)
```
class ClassBody{
    public static void main(String[] args){
        System.out.println(" クラスはクラス宣言とクラス本体とで構成される。");
    }
}
```

(3)
```
class ObjectOriented{
    public static void main(String[] args){
        System.out.println("Java はオブジェクト指向言語である。");
    }
}
```

(4)
```
class JVM{
    public static void main(String[] args){
        System.out.println("JVM は Java Virtual Machine の略である。");
    }
}
```

(5)
```
class NoMercy{
    public static void main(String[] args){
        System.out.println(
          "Computers are like Old Testament gods ;" +
          "lots of rules and no mercy. ");
    }
}
```

■第 1 章 §2 復習問題 (p.13)

1．コメントの利用
①コメント　②//　③/*　④*/　⑤/**　⑥*/　⑦HTML
2．文字列の連結
①数値式　②+　③文字列
3．算術演算子
①算術演算子　②+　③-　④*　⑤/　⑥%　⑦剰余　⑧左　⑨右

■第 1 章 §2 練習問題 (p.13)

1．（略）
■実行結果
(1)

```
8-4*5%3+4/2 = 8
(8-5)*5%3+4/2 = 2
8-5*5%(3+4)/2 = 6
(8-4)*5%(3+4)/2 = 3
```

(2)

```
15-6+7-8 = 8
15-(6+7)-8 = -6
15-(6+(7-8)) = 10

16/4*2 = 8
(16/4)*2 = 8
16/(4*2) = 2

17+20%3 = 19
(17+20)%3 = 1
17+(20%3) = 19
```

■第 1 章 §2 実習問題 (p.14)

1．【計算式と計算結果】

```
/*
    CalcEx3.java
    澄川希穂
    20XX.4.18
*/
class CalcEx3{
    public static void main (String[] args) {
        System.out.println("7+3*(6/2-1) = " + (7+3*(6/2-1)));
        System.out.println("(7+3)*6/2-1 = " + ((7+3)*6/2-1));
        System.out.println("11%2+5*2-18/2 = " + (11%2+5*2-18/2));

        System.out.println("11%(2+5)*2-18/2 = " + (11%(2+5)*2-18/2));
        System.out.println("3*(9-3)*(9+6)/3 = " + (3*(9-3)*(9+6)/3));
        System.out.println("3*9-(3*(9+6/3)) = " + (3*9-(3*(9+6/3))));
    }
}
```

2．【アステリスクによる図形表示】

(1)

```
class MyDiamond{
    public static void main(String[] args){
        System.out.println("   *");
        System.out.println("  * *");
        System.out.println(" *   *");
        System.out.println("*     *");
        System.out.println(" *   *");
        System.out.println("  * *");
        System.out.println("   *");
    }
}
```

(2)

```
class MyTree{
    public static void main(String[] args){
        System.out.println("   *");
        System.out.println("  ***");
        System.out.println(" *****");
        System.out.println("*******");
        System.out.println("   *");
        System.out.println("   *");
    }
}
```

3．【ギザのピラミッドの体積 1】

```
class Pyramid{
    public static void main(String[] args){
        System.out.println(
            "このピラミッドの体積は、" + 230*230*139/3 + "立方メートルです。");
    }
}

/*
■実行結果
このピラミッドの体積は、2451033 立方メートルです。
*/
```

4．【Y 市の水道料金】

```
class WaterFee{
    public static void main(String[] args){
        System.out.println("5 月、6 月の水道料金は" +
            (1580+43*(20-16)+158*(40-20)+226*(45-40)) + "円です。");
    }
}

/*
■実行結果
5 月、6 月の水道料金は 6042 円です。
*/
```

■第 2 章 §1 復習問題 (p.16)

1．【変数とデータ型 I】

(1) ① 変数　② 変数名　③ データ型　④ 値
(2) ⑤ 数字　⑥ _（アンダースコア）　⑦ $（ドル記号）　⑧ 小文字
(3) ⑨ 基本データ　⑩ 参照
(4) ⑪ char 型　⑫ 真偽値
(5) ⑬ short　⑭ float　⑮ long
(6) ⑯ int　⑰ double

2.【変数とデータ型Ⅱ】

(1) ① 変数の宣言
(2) ② 代入演算子
(3) ③ リテラル
(4) ④ int　⑤ double　⑥ f(F)　⑦ d(D)
(5) ⑧ 初期値の設定（初期化）　⑨右
(6) ⑩ 文字コード　⑪ 2　⑫ true　⑬ false

■第 2 章 §1 練習問題 (p.17)

1.
- ✓ (1) `@125`　@ 記号は使用できない。
- ✓ (2) `MyData`　変数名の冒頭文字に大文字を使ってもエラーにはならないが、Java の慣行で必ず小文字を使う。
- (3) `$dollar`　$ 記号は使用できる。
- ✓ (4) `¥yen`　¥ 記号は使用できない。
- (5) `tall_coffee`　_ 記号は使用できる。
- (6) `売上高`　漢字は使用できる。
- ✓ (7) `5w1h`　冒頭の文字に数字は使用できない。
- (8) `ぽち`　ひらがな・カタカナも使用できる。
- (9) `next`　next は Java のキーワードではないので使用できる。
- ✓ (10) `return`　return は Java のキーワードなので使用できない。

2.
- ✓ (1) byte 型の値の範囲は -128 ～ +127
- (2) short 型の値の範囲は -32768 ～ +32767
- (3) int 型の最大値は 2147483647
- ✓ (4) 浮動小数点数リテラルは f 指定がなければ暗黙のうちに double 型と解釈される。
- (5) double 型は符号 1 bit、絶対値 63bit で表現
- (6) char 型に整数は代入できる。
- ✓ (7) char 型に文字列は代入できない。
- ✓ (8) boolean 型には true、false 以外は代入できない。

3.
(1) ② z = 1.0
(2) ④コンパイルエラー
(3) ② 25
(4) ③ 93558 まで

(1) 右辺の x/y の値は 1 （int 型）。それを左辺の double 型に代入するので 1.0 になる。
(2) 3*myByte は自動的に int 型に昇格するので short 型に代入できない。

(3) char 型の実態は文字コード（数値）である。

(4) char 型変数の前に文字があると char 型変数は文字として表示されるが、それがないと文字コードの数値加算として処理される。

例

```
class Opp04{
    public static void main(String[] args){
        char x = '子';
        char y = '孫';
        System.out.println("私の代から" + x + x + y + y + "まで");
    }
}
```

■実行結果

```
私の代から子子孫孫まで
```

■第 2 章 §1 実習問題 (p.19)

1．【変数を使った計算】

(1)

```
class CalcTest01{
    public static void main(String[] args){
        int a = 3;
        int b = 5;
        int c = 11;
        int m;

        m = a + b * c;
        System.out.println("m = " + m);
    }
}
```

■実行結果

```
m = 58;
```

(2)

```
class CalcTest02{
    public static void main(String[] args){
        int a = 3;
        int b = 5;
        int c = 11;
        int m;

        m = a*a + b*b + c*c;
        System.out.println("m = " + m);
    }
}
```

■実行結果

```
m = 155;
```

(3)

```
class CalcTest03{
    public static void main(String[] args){
        int a = 3;
        int b = 5;
        int c = 11;
        double m;

        m = 3.0 * (a + c) / (c - b);
        System.out.println("m = " + m);
    }
}
```

■実行結果

```
m = 7.0;
```

2．【期末テストの合計と平均】

```
class KimatsuTest{
    public static void main(String[] args){
        // 変数の宣言
        int kokugo;
        int sugaku;
        int eigo;
        int gokei;
        double heikin;

        // 変数にデータを入れる（初期値の設定）。
        kokugo = 76;
        sugaku = 63;
        eigo = 82;

        // 合計・平均を計算する。
        gokei = kokugo + sugaku + eigo;
        heikin = gokei/3.0;

        // 計算結果を表示する。
        System.out.println("合計 " + gokei + " 点");
        System.out.println("平均 " + heikin + " 点 ");
    }
}
```

■実行結果

```
合計 221 点
平均 73.66666666666667 点
```

3.【ギザのピラミッドの体積 2】

(1)

```
class Pyramid01{
    public static void main(String[] args){
        int length = 230;
        int height = 139;
        int volume;

        volume = length * length * height / 3;
        System.out.println("このピラミッドの体積は、" + volume +
                                              "立方メートルです。");
    }
}
```

(2)

```
class Pyramid02{
    public static void main(String[] args){
        int length = 230;
        int height = 139;
        int volume;
        int touch = 105;     // タッチラインの長さ
        int goal = 68;       // ゴールラインの長さ
        int newHeight;       // 直方体の高さ

        volume = length * length * height / 3;
        newHeight = volume / (105 * 68);

        System.out.println("このピラミッドの体積は、" + volume +
                                              "立方メートルです。");
        System.out.println("この体積をサッカーのピッチに積み上げると" +
                    newHeight + "メートルの高さの直方体になります。");
    }
}
```

■実行結果

```
このピラミッドの体積は、2451033 立方メートルです。
この体積をサッカーのピッチに積み上げると 343 メートルの高さの直方体になります。
```

4.【ハンバーガーショップの売上計算】

```
class BurgerShop{
    public static void main(String[] args){
        // 変数の宣言
        int burger;
        int cheese;
        int big;
        int hot;
        int soup;
        int drink;
        int sum;
        double average;

        // 商品別売上額
        burger = 160 * 569;
        cheese = 190 * 241;
        big = 260 * 176;
        hot = 220 * 317;
        soup = 240 * 74;
        drink = 110 * 488;
        // 本日の売上総額と客平均売上額
        sum = burger + cheese + big + hot + soup + drink;
        average = sum / 1154.0;
        // 計算結果の表示
        System.out.println("本日の売上総額：" + sum + "円");
        System.out.println("本日の客平均売上額：" + average + "円");
    }
}
```

■実行結果

```
本日の売上総額：323770 円
本日の客平均売上額：280.5632582322357 円
```

■第 2 章 §2 復習問題 (p.21)

1．定数の宣言
① 定数　② final　③ 大文字
2．データの代入と変数の型
① char　② boolean　③ short　④ long　⑤ float
3．型変換とキャスト演算子
① 型変換　② 拡張変換　③ 縮小変換　④ キャスト　⑤ キャスト演算子
4．複合代入演算子
① 複合代入演算子　② a-=b　③ a/=b　④ a%=b　⑤ インクリメント演算子
⑥ デクリメント演算子　⑦ 前置型　⑧ 後置型

■第 2 章 §2 練習問題 (p.22)

1．コンパイルエラーになるもの　(1) (2) (4)

(1) byte は 1 バイト、short は 2 バイトなので、キャストを使わないと byte ← short の代入段階で縮小変換エラーが発生する。
(2) byte 型は 1 バイト、char 型は 2 バイトなので拡張変換で問題なく代入できるそうだが、char 型は char 型データ以外受け付けないので、エラーとなる。
(3) char 型 2 バイト、float 型 4 バイトなので問題なく代入できる。
(4) 浮動小数点数は自動的に double 型と見なされる。double 型は float 型にそのまま代入できないのでエラーになる。
(5) キャストによる float → long 変換。コンパイルできるが、小数部分は切り捨てられる。
(6) 精度が落ちる（有効桁数 16 けた→ 7 けた）がコンパイルできる。

2．コンパイルエラーになるもの　① ④

①文字リテラルは '（シングルクォーテーション）で囲む。"（ダブルクォーテーション）で囲むと 1 字でも文字列になる。
② char 型の中身は文字コード（数値）なので、整数はキャストにより char 型へ代入できる。
④ boolean 型へ代入できるリテラルは true と false だけである。

3．(1) ① [16.55]　② [16]
(2) ③ x = [11]　④ y = [14]　⑤ z = [17]
(3) ⑥ a += b → a = [8]　⑦ a -= b → a = [3]　⑧ a *= --b → a = [12]
⑨ a /= b++ → a = [3]　⑩ a %= b → a = [3]

計算途中で変数の値が変化することに注意する。

■第 2 章 §2 実習問題 (p.24)

1．【豪華客船の航海時間】

```
class Ikaruga{
    public static void main(String[] args){
        double knotPerHour = 17.6;        // 平均航海速力（ノット）
        int distance = 6212;              // 横浜、ホノルル間の距離
        double knot = 1852/1000d;         // 1 ノットの km 換算値
        int time;                         // 総所要時間
        int days, hours;

        time = (int)(distance/(knotPerHour*knot));
        days = time/24;
        hours = time%24;
        System.out.printf("航海日時は %d 日と %d 時間です。", days, hours);
    }
}
```

■実行結果

```
航海日時は 7 日と 22 時間です。
```

2．【株式会社の自己資金比率】

```
class Tenraku{
    public static void main(String[] args){
        long asset = 1949516816655L;
        long debt = 1700183225604L;
        int stocks = 13181697;
        double capital;
        double equity;

        capital =(double)(asset - debt) / asset * 100;
        equity = (double)(asset - debt) / stocks;

        System.out.println(" 天楽株式会社 ");
        System.out.println(" 自 己 資 本 比 率 : " + capital + "%");
        System.out.println("1 株当たり純資産額 : " + equity + " 円 ");
    }
}
```

■実行結果

```
天楽株式会社
自 己 資 本 比 率 : 12.789507067644024%
1 株当たり純資産額 : 18915.13596853273 円
```

■第 3 章 §1 復習問題 (p.26)

1．データの入力
① Scanner　② パッケージ　③ util　④ import　⑤ インスタンス
⑥ new 演算子　⑦ オブジェクト　⑧ nextInt()

2．Scanner クラスの入力用メソッド
① nextShort()　② nextLong()　③ nextLine()

3．数値の書式付き出力
① 書式指定子　② %f　③ %,10d　④ %-10d　⑤ %,+15f　⑥ 15.4f

4．エスケープシークエンス
① 改行　② ¥t　③ ¥¥　④ ¥"%d¥¥n¥"

■第 3 章 §1 練習問題 (p.28)

(1) ① import java.util.Scanner　② new Scanner(System.in)
③ input.nextInt()　④ printf

(2) ⑤ %20d¥t%20d¥n　⑥ Integer.MIN_VALUE　⑦ Integer.MAX_VALUE)
⑧ 127　⑨ 2147483647

■第 3 章 §1 実習問題 (p.29)

1.【基数変換】

```
import java.util.Scanner;

class RadixConverter{
    public static void main(String[] args){
        int intValue;
        Scanner scan = new Scanner(System.in);

        System.out.print("10 進数を入力してください。:");
        intValue = scan.nextInt();
        System.out.printf("10 進数：%d¥n", intValue);
        System.out.printf(" 8 進数：%o¥n", intValue);
        System.out.printf("16 進数：%X¥n", intValue);
    }
}
```

2.【不快指数の計算】

```
import java.util.Scanner;

class DisIndex{
    public static void main(String[] args){
        double temp;
        double hum;
        double disIndex;
        Scanner scan = new Scanner(System.in);

        System.out.print(" 気温を入力してください。: ");
        temp = scan.nextDouble();
        System.out.print(" 湿度を入力してください。: ");
        hum = scan.nextDouble();

        disIndex = 0.81*temp+0.01*hum*(0.99*temp-14.3)+46.3;

        System.out.printf(" 不快指数は、%5.2f です。", disIndex);
    }
}
```

3.【歩行による消費エネルギー】

```
import java.util.Scanner;

class MyWalking{
    public static void main(String[] args){
        int minute;
        double weight;
        int kcal;
        Scanner scan = new Scanner(System.in);

        System.out.print(" 歩行時間を入力してください ( 分 )。: ");
        minute = scan.nextInt();
        System.out.print(" 体重を入力してください (kg)。: ");
        weight = scan.nextDouble();

        kcal = (int)(3 * minute/60 * weight);

        System.out.printf(
            " 本日の歩行による消費カロリーは、%d kcal です。", kcal);
    }
}
```

■第 3 章 §2 復習問題 (p.30)

1．文字列変数とデータの入力
① String ② %s
2．String クラスのメソッド
① myString.length() ② myString.charAt(6) ③ stra ④ 2 ⑤ 7
⑥ -1 ⑦ 4
3．Math クラスのメソッド
① Math ② lang ③ Math.pow(2, 3) ④ sqrt(double x) ⑤ 12.34
⑥ 13.00 ⑦ -13.00 ⑧ Math.random()*11 ⑨ toRadians ⑩ E

■第 3 章 §2 練習問題 (p.32)

1． (1) ① 17.0 ② 8.0 ③ 3.0
(2) ④ 1.0 ⑤ -1.0 ⑥ -6.0 ⑦ -9.0
2． ①ウ ②ア ③オ

イ 10 12 0 8 4 4 16 18 14 6 (int)(Math.random()*10)*2
エ 11 16 10 14 11 12 15 19 13 15 (int)(Math.random()*10)+10
3．① substring ② indexOf ③ str.charAt(7) ④ subStr1.length()
4．① nextLine() ② replace ③ ¥n

■第 3 章 §2 実習問題 (p.34)

1．【入力数値のけた数】

```
import java.util.Scanner;

public class Digits{
    public static void main(String[] args){
        String inputStr;
        Scanner scan = new Scanner(System.in);

        System.out.print("正の整数を入力してください。:");
        inputStr = scan.next();
        System.out.printf(
            "入力された数値のけた数は %d です。", inputStr.length());
    }
}
```

2．【正三角形の面積計算】

```
import  java.util.Scanner;

class TriangleSpace{
    public static void main(String[] args){
        double side;
        double area;
        Scanner scan = new Scanner(System.in);

        System.out.print("正三角形の 1 辺を入力してください。: ");
        side = scan.nextDouble();
        area = Math.sqrt(3)/4*Math.pow(side, 2);
        System.out.printf(
            "1 辺が %.4f の正三角形の面積は %.4f です。", side, area);
    }
}
```

3．【東京スカイツリーの高さ】

```
class SkyTree{
    public static void main(String[] args){
        double distance = 1300.0;
        double degree = 26.0;
        int height;

        height = (int)(distance * Math.tan(Math.toRadians(degree)));
        System.out.printf(
            "東京スカイツリーの高さは %d メートルです。", height);
    }
}
```

■実行結果

```
東京スカイツリーの高さは 634 メートルです。
```

■第 4 章 §1 復習問題 (p.35)

1．制御構造
① 制御構造　② 順次処理　③ 判定処理　④ 繰り返し処理
2．if 文
① 真　② 偽
3．比較演算子と論理演算子
① 大小比較　② 関係演算子　③ 代入演算子　④ 論理　⑤ 論理積
⑥ ！　⑦ 論理積（and）　⑧ true　⑨ ||　⑩ !=

■第 4 章 §1 練習問題 (p.36)

1．(1)

```
if(degree<90)
    System.out.println("鋭角");
else
    System.out.println("鈍角");
```

(2)

```
if(i%10==0)
    System.out.print("¥n");
```

(3)

```
if(!isFemale)
    male++;
```

(4)

```
if(age<25)
    System.out.println("Good morning!");
else
    System.out.println("Good bye!");
```

(5)

```
if(url.charAt(i-1) == "@")
    System.out.println(url.substring(i));
```

2．① true　② false　③ false　④ true　⑤ true　⑥ false　⑦ true

3．① 3　　② 8　　③ 6　　④ 1

(2)、(3) では論理演算子 || の左の項が true なので、その評価段階で式全体が true と評価され、|| の右の項の評価は省略される。したがって、q に 1 が加算されないことに注意する。

■第 4 章 §1 実習問題 (p.38)

1．【暗証番号のけた数】

```
import java.util.Scanner;

class FourDigits{
    public static void main(String[] args){
        Scanner scan = new Scanner(System.in);
        System.out.print("暗証番号を入力してください。:");
        int digits = scan.nextInt();
        if(digits<1000 || digits>9999)
            System.out.println("4 けたの数字を入力してください。");
        else
            System.out.println("****");
    }
}
```

【別解】

```
import java.util.Scanner;

class FourDigits{
    public static void main(String[] args){
        Scanner scan = new Scanner(System.in);
        System.out.print("暗証番号を入力してください。:");
        String digits = scan.nextLine();
        if(digits.length() != 4)
            System.out.println("4 けたの数字を入力してください。");
        else
            System.out.println("****");
    }
}
```

2．【入力数値の判定 1】

```
import java.util.Scanner;

class SimpleIf01{
    public static void main(String[] args){
        int num1, num2, sum;
        Scanner scan = new Scanner(System.in);

        System.out.print("はじめの数を入力してください。: ");
        num1 = scan.nextInt();
        System.out.print("２番目の数を入力してください。: ");
        num2 = scan.nextInt();
        sum = num1 + num2;

        if(sum%2==0)
            System.out.printf("合計値 %d は偶数です。¥n", sum);
        if(sum%3==0)
            System.out.printf("合計値 %d は３の倍数です。¥n", sum);
        if(sum%5==0)
            System.out.printf("合計値 %d は５の倍数です。¥n", sum);
    }
}
```

2.【三角形の 3 辺の長さと面積】

```
import java.util.Scanner;

class HeronTest{
    public static void main(String[] args){

        double a, b, c, s;
        double area;

        Scanner scan = new Scanner(System.in);

        System.out.print("辺 a の長さを入力してください :");
        a = scan.nextDouble();
        System.out.print("辺 b の長さを入力してください :");
        b = scan.nextDouble();
        System.out.print("辺 c の長さを入力してください :");
        c = scan.nextDouble();

        if (Math.abs(a-b) < c && c < a + b){
                s = (a + b + c) / 2;
                area = Math.sqrt(s * (s - a) * (s - b) * (s - c));
                System.out.printf(
                    "%.2f,%.2f,%.2f を３辺とする三角形の面積は、%.2f です。",
                    a, b, c, area);
        }else
            System.out.printf(
                "%.2f, %.2f, %.2f は三角形の３辺を構成しません。",
                 a, b, c);
    }
}
```

■第 4 章 §2 復習問題 (p.39)

1. if～else 文
   ① if～else　　② else if
2. コードブロック
   ① コードブロック　　② 制御

■第 4 章 §2 練習問題 (p.40)

1.

```
import java.util.Scanner;

class FlowIf01{
    public static void main(String[] args){
        Scanner scan = new Scanner(System.in);
        System.out.print("整数を入力してください。:");
        int inputValue = scan.nextInt();

        if(inputValue%2 == 0){
            System.out.printf("%d は偶数 ¥n", inputValue);
            if(inputValue%5 == 0)
                System.out.printf("%d は 10 の倍数 ¥n",inputValue);
        }else
            System.out.printf("%d は奇数 ¥n", inputValue);
    }
}
```

2.
(1) 今日の昼食はローストビーフ　　①〔 イ 〕 ②〔 ウ 〕 ③〔 ア 〕
(2) 今日の昼食は特製カレー　　①〔 ア 〕 ②〔 ア 〕 ③〔 ア 〕
(3) 今日の昼食はスイートコーン　　①〔 ア 〕 ②〔 イ 〕 ③〔 ア 〕
(4) 今日の昼食はカリフォルニアロール　　①〔 ウ 〕 ②〔 ウ 〕 ③〔 イ 〕

(1)　ローストビーフを表示するには、①を true、③を false にする必要がある。②は①が true ならスキップされるので、true、false どちらでもよい。
(2)　特製カレーを表示するには①を false、②を true、③を false にする必要がある。
(3)　スイートコーンを表示するには、①、②、③すべてを false にする必要がある。
(4)　カリフォルニアロールを表示するには、③を true にすればよい。③が true であれば、myLunch に何が記憶されていても、最後にカリフォルニアロールで上書きする。①、②の真偽は結果に無関係になる。

■第 4 章第 §2 実習問題 (p.41)

1.【入力数値の判定 2】

```
import java.util.Scanner;

class SimpleIf{
    public static void main(String[] args){
        int num1, num2, sum;
        Scanner scan = new Scanner(System.in);

        System.out.print(" はじめの数を入力してください．：");
        num1 = scan.nextInt();
        System.out.print(" ２番目の数を入力してください．：");
        num2 = scan.nextInt();
        sum = num1 + num2;

        if(sum%2==0)
            System.out.printf(" 合計値 %d は偶数です。¥n", sum);
        else if(sum%3==0)
            System.out.printf(" 合計値 %d は３の倍数です。¥n", sum);
        else if(sum%5==0)
            System.out.printf(" 合計値 %d は５の倍数です。¥n", sum);
        else
            System.out.printf(
                " 合計値 %d は偶数でも３、５の倍数でもありません。¥n", sum);
    }
}
```

次のように、上記の if 文で途中の else を省略すると、最後の else 文はその直上の if 文にのみ対応するので、偶数ないし３の倍数のときも、「合計値 %d は偶数でも３、５の倍数でもありません。」と表示してしまう。if と else の対応関係には常に注意する必要がある。

```
                                    )

        if(sum%2==0)
            System.out.printf(" 合計値 %d は偶数です。¥n", sum);
        if(sum%3==0)
            System.out.printf(" 合計値 %d は３の倍数です。¥n", sum);
        if(sum%5==0)
            System.out.printf(" 合計値 %d は５の倍数です。¥n", sum);
        else
            System.out.printf(
                " 合計値 %d は偶数でも３、５の倍数でもありません。¥n", sum);
    }
}
```

2.【プロボクシングの階級判定】

```
import java.util.Scanner;

class ProBoxing{
    public static void main(String[] args){
        double weight;
        int lbWeight;
        String rank;
        final double LB = 0.45359237;
        Scanner scan = new Scanner(System.in);

        System.out.print("体重を入力してください（kg）。:");
        weight = scan.nextDouble();
        lbWeight =(int)Math.ceil(weight/LB);
        if(lbWeight<=105)
            rank = "ミニマム";
        else if(lbWeight<=108)
            rank = "ライト・フライ";
        else if(lbWeight<=112)
            rank = "フライ";
        else if(lbWeight<=118)
            rank = "バンタム";
        else if(lbWeight<=122)
            rank = "スーパー・バンタム";
        else if(lbWeight<=126)
            rank = "フェザー";
        else if(lbWeight<=130)
            rank = "スーパー・フェザー";
        else if(lbWeight<=135)
            rank = "ライト";
        else if(lbWeight<=140)
            rank = "スーパー・ライト";
        else if(lbWeight<=147)
            rank = "ウェルター";
        else if(lbWeight<=154)
            rank = "スーパー・ウェルター";
        else if(lbWeight<=160)
            rank = "ミドル";
        else if(lbWeight<=168)
            rank = "スーパー・ミドル";
        else if(lbWeight<=175)
            rank = "ライト・ヘビー";
        else if(lbWeight<=190)
            rank = "クルーザー";
        else
            rank = "ヘビー";

        System.out.printf("ランクは%s級です。", rank);
    }
}
```

3.【入力数値による水道料金計算】

```
import java.util.Scanner;

class WaterRates{
    public static void main(String[] args){
        int myRates;
        int wFlow = 0;
        Scanner scan = new Scanner(System.in);

        System.out.print("使用量を入力してください。:");
        wFlow = scan.nextInt();
```

（次ページへ続く）

```
        if(wFlow<17)
            myRates = 1580;
        else if(wFlow<21)
            myRates = 43*(wFlow-16) +1580;
        else if(wFlow<41)
            myRates = 158*(wFlow-20) + 43*(20-16) + 1580;
        else if(wFlow<61)
            myRates = 226*(wFlow-40) + 158*(40-20) + 43*(20-16)
                      + 1580;
        else if(wFlow<101)
            myRates = 269*(wFlow-60) + 226*(60-40)
                      + 158*(40-20) + 43*(20-16) + 1580;
        else if(wFlow<201)
            myRates = 293*(wFlow-100) + 269*(100-60) + 226*(60-40)
                      + 158*(40-20) + 43*(20-16) + 1580;
        else
            myRates = 320*(wFlow-200) + 293*(200-100) + 269*(100-60)
                      + 226*(60-40) + 158*(40-20) + 43*(20-16) + 1580;
        System.out.printf(
            "水道使用量は %d 立方メートル、料金は％d円です。¥n", wFlow, myRates);
    }
}
```

4. 【郵便料金の計算】

```
import java.util.Scanner;

class Postage{
    public static void main(String[] args){
        boolean isRegular = false;
        int weight = 0;
        int fee = 0;
        boolean isLetter = true;

        Scanner input = new Scanner(System.in);
        System.out.print("定形郵便物ですか（定型 1：定形外 2）？：");
        if(input.nextInt()==1)
            isRegular = true;
        else
            isRegular = false;

        System.out.print("封書の重さは何グラム？ :");
        weight = input.nextInt();
        if(isRegular && weight<=50){
            if(weight <= 25)
                fee = 82;
            else
                fee = 92;
        }else{
            if(weight <= 50)
                fee = 120;
            else if(weight <= 100)
                fee = 140;
            else if(weight <= 150)
                fee = 205;
            else if(weight <= 250)
                fee = 250;
            else if(weight <= 500)
                fee = 400;
            else if(weight <= 1000)
                fee = 600;
            else if(weight <= 2000)
                fee = 870;
            else if(weight <= 4000)
                fee  = 1180;
            else
                isLetter = false;
        }
        if(isLetter)
            System.out.printf("料金は %d 円です。", fee);
        else
            System.out.println("手紙ではありません。５番窓口へどうぞ。");
    }
}
```

■第 4 章 §3 復習問題 (p.43)

1.【Switch 文】
① 整数値　② case　③ break　④ default　⑤ long　⑥ boolean
⑦ float　⑧ double　⑨ case 6: case 7: case8:　⑩ default

2.【ブール型変数による判定】
⑪ is ～

■第 4 章 §3 練習問題 (p.44)

1. (1) 制御構造の 3 つの分類は、順次処理、判定処理、繰り返し処理。
(4) 条件演算子ではなく比較演算子（関係演算子でもよい）。
(7) 文字はコンピュータ内部では文字コードという整数値として記憶されている。
したがって、利用できる。
(8) default 句は省略してもエラーにはならない。

2. (1)

```
int num = (int)(Math.random()*10 + 1);
switch (num){
    case 1:
        System.out.println(" 1等賞！ ");
        break;
    case 2:case 3:
        System.out.println(" 2等賞！ ");
        break;
    case 4:case 5:case 6:
        System.out.println(" 3等賞！ ");
        break;
    default:
        System.out.println(" 残念賞 ");
        break;
}
```

(2)

```
System.out.print(" はじめの数を入力してください。:");
firstInt = scan.nextInt();
System.out.print("2 番目の数を入力してください。: ");
secondInt = scan.nextInt();
System.out.print(" 演算子を入力してください。: ");
op = scan.next().charAt(0);

switch(op){
    case '+':
        System.out.printf("%d + %d = %d",
            firstInt, secondInt, firstInt + secondInt);
        break;
    case '-':
        System.out.printf("%d - %d = %d",
            firstInt, secondInt, firstInt - secondInt);
        break;
    case '*':
        System.out.printf("%d * %d = %d",
            firstInt, secondInt, firstInt * secondInt);
        break;
    case '/':
        System.out.printf("%d / %d = %d",
            firstInt, secondInt, (int)firstInt / secondInt);
        break;
    case '%':
        System.out.printf("%d %% %d = %d",
            firstInt, secondInt, firstInt % secondInt);
        break;
    default:
        System.out.printf(" 演算子が不正です。");
        break;
}
```

■第 4 章 §3 実習問題 (p.45)

1.【switch 文による曜日判定】

```
import java.util.Scanner;

class DayOfWeek{
    public static void main(String[] args){
        int day;
        String week;
        Scanner scan = new Scanner(System.in);
        System.out.print(" 日にちを入力してください。: ");
        day = scan.nextInt();

        int num = day % 7;
        switch(num){
            case 0:
                week = " 日曜日 ";
                break;
            case 1:
                week = " 月曜日 ";
                break;
            case 2:
                week = " 火曜日 ";
                break;
            case 3:
                week = " 水曜日 ";
                break;
            case 4:
                week = " 木曜日 ";
                break;
            case 5:
                week = " 金曜日 ";
                break;
            case 6:
                week = " 土曜日 ";
                break;
            default:
                week = " 誕生日 ";
                break;
        }
        System.out.printf("%d 日は %s です。", day, week);
    }
}
```

このプログラムでは、入力値を 7 で割った余りを各 case の飛び先の値としている。したがって、num は 0 ～ 6 の整数値以外の値をとることはなく、default 句が実行されることは論理的にありえない。それにもかかわらず、default 句で week に何か文字列を設定しないと、コンパイル時に「変数 week は初期化されていない可能性があります。」と警告が出てエラーになる。Java のコンパイラのエラーチェックは、プログラムの文法上・形式上の範囲にとどまり、その演算結果まで含めてエラーかどうか判断しているわけでないことに注意する。

2.【switch 文による干支判定】

```
import java.util.Scanner;

class AnimalYear{
    public static void main(String[] args){
        String animals = " 子丑寅卯辰巳午未申酉戌亥 ";
        int year;
        char eto;
        Scanner scan = new Scanner(System.in);

        System.out.print(" 西暦年を入力してください。:");
        year = scan.nextInt();
```

（次ページへ続く）

```
        switch(year%12){
             case 0:
                 eto = animals.charAt(8);
                 break;
             case 1:
                 eto = animals.charAt(9);
                 break;
             case 2:
                 eto = animals.charAt(10);
                 break;
             case 3:
                 eto = animals.charAt(11);
                 break;
             case 4:
                 eto = animals.charAt(0);
                 break;
             case 5:
                 eto = animals.charAt(1);
                 break;
             case 6:
                 eto = animals.charAt(2);
                 break;
             case 7:
                 eto = animals.charAt(3);
                 break;
             case 8:
                 eto = animals.charAt(4);
                 break;
             case 9:
                 eto = animals.charAt(5);
                 break;
             case 10:
                 eto = animals.charAt(6);
                 break;
             case 11:
                 eto = animals.charAt(7);
                 break;
             default:
                 eto = '猫';
                 break;
         }
         System.out.printf("%d年は%s年です。", year, eto);
    }
}
```

3.【基礎代謝量の計算】

```
import java.util.Scanner;

class Metabolism{
    public static void main(String[] args){
        int sx;
        double weight;
        double height;
        int age;
        double met = 0;
        Scanner input = new Scanner(System.in);

        System.out.print("性別を入力してください（女→'1'　男→'2'）。:");
        sx = input.nextInt();

        System.out.print("体重を入力してください(kg)。:");
        weight = input.nextDouble();
        System.out.print("身長を入力してください(cm)。:");
        height = input.nextDouble();
        System.out.print("年齢を入力してください。:");
        age = input.nextInt();
```

（次ページへ続く）

```
        switch(sx){
            case 1:
                met = 655.1+9.56*weight+1.85*height-4.68*age;
                break;
            case 2:
                met = 66.47+13.75*weight+5.0*height-6.76*age;
                break;
            default:
                System.out.println("入力値が範囲外です。");
                break;
        }
        System.out.printf("あなたの基礎代謝量は %6.2fkcal です。", met);
    }
}
```

■第 5 章 §1 復習問題 (p.46)

1．繰り返し処理の制御文
① for ② while ③ do ～ while
2．for 文
① 初期値 ② 継続条件 ③ 増分
④ 制御変数 ⑤ boolean
3．いろいろな for 文
① i+=5 ② i-=11 ③ int i=1; i<=m; i++ ④ 6
⑤ ch++ ⑥ %.2f

■第 5 章 §1 練習問題 (p.47)

1．① i+=x++ ② i*=i*0.5 ③ char ch='あ'; ch<='ん'; ch++

2．① += (long)Math.pow(num, 3); ② += (long)i;

■第 5 章 §1 実習問題 (p.49)

1．【文字コードの進数表記】

```
class CharCode{
    public static void main(String[] args){
        System.out.println("文字  コード  8 進表記  16 進表記");
        for(char ch='A'; ch<='Z'; ch++)
            System.out.printf("%c%10d%10o%10x¥n",
                              ch, (int)ch, (int)ch, (int)ch);
    }
}
```

5 行目の printf 文は「 < 」を使って、

```
System.out.printf("%c%10d%<10o%<10x¥n", ch, (int)ch);
```

と書いてもよい。「< 」フラグは、直前の書式指定子の引数の再利用を指示するので、同一の引数を連記しなくても済む。
→ JavaAPI java.util.Formatter クラスの「引数のインデックス」

2.【2 乗の総和】

```
import java.util.Scanner;

class Power2Sum{
    public static void main(String[] args){
        int max;
        long sum = 0L;
        Scanner scan = new Scanner(System.in);
        System.out.println(
               "1 から入力された数までの各整数の２乗の総和を求めます。");
        System.out.print(" 上限の数を入力してください。:");
        max = scan.nextInt();
        for(int i=0; i<=max; i++)
            sum += (long)Math.pow(i, 2);
        System.out.printf(
               "1 ～ %d の各整数の２乗の総和は %d です。¥n", max, sum);
    }
}
```

3.【２整数間の数値累計】

```
import java.util.Scanner;

class SumOfInterval{
    public static void main(String[] args){
        int startInt;
        int endInt;
        int sum = 0;
        Scanner scan = new Scanner(System.in);

        System.out.print(" はじめの数を入力してください。:");
        startInt = scan.nextInt();
        System.out.print(" 最後の数を入力してください。: ");
        endInt = scan.nextInt();

        for(int i=startInt; i<=endInt; i++)
            sum += i;

        System.out.printf(
            "%d ～ %d の合計は、%d です。", startInt, endInt, sum);
    }
}
```

4.【閏（うるう）年の回数】

```
import java.util.Scanner;

class GregorianLeapYear{
    public static void main(String[] args){
        int leapYear = 0;
        int startYear = 1582;
        int thisYear;

        Scanner scan = new Scanner(System.in);
        System.out.print(" 今年は西暦何年？ :");
        thisYear = scan.nextInt();

        for(int year=startYear; year<=thisYear; year++){
            if (year%400 == 0 || year%4 == 0 && year%100 != 0)
                leapYear++;
        }
        System.out.printf(
             "%d 年から今年までの間に閏年は %d 日ありました。",
                                startYear, leapYear);
    }
}
```

5．【いろいろな数列・級数】

(1) ・・・・・・・・・・・ → 2？

```
class Series01{
    public static void main(String[] args){
        double sum = 0.0;

        for(int i=0; i<20; i++){
            sum += 1/Math.pow(2, i);
            System.out.printf("第%2d項までの合計：%.15f¥n", i+1, sum);
        }
    }
}
```

(2) ・・・・・・・・・・・ → 2/3？

```
class Series02{
    public static void main(String[] args){
        double sum = 0.0;

        for(int i=0; i<20; i++){
            if(i%2 == 0)
                sum += 1/Math.pow(2,i);
            else
                sum -= 1/Math.pow(2,i);
            System.out.printf("第%2d項までの合計:%.15f¥n", i+1, sum);
        }
    }
}
```

(3) ・・・・・・・・・・・ →$\sqrt{2}$？

```
class Series03{
    public static void main(String[] args){
        int head = 1;
        int body = 1;

        for(int i=1; i<=20; i++){
            System.out.printf(
                "第%2d項:%.15f¥n", i, (double)head/body);
            int temp = head;
            head += body * 2;
            body += temp;
        }
    }
}
```

(4) ・・・・・・・・・・・ → π ？

```
class Series04{
    public static void main(String[] args){
        double num = 4.0;
        System.out.printf("第%4d項：%.15f¥n", 1, num);

        for(int i=2; i<=1000; i++){
            if(i%2 == 0)
                num *= (double)i/(i+1);
            else
                num *= (double)(i+1)/i;
            System.out.printf("第%4d項：%.15f¥n", i, num);
        }
    }
}
```

(5) …… → e ?

```
class Series05{
    public static void main(String[] args){
        double num = 1.0;
        double factorial = 1.0;

        for(int i=1; i<=15; i++){
            factorial *= (double)i;
            num += 1/factorial;
            System.out.printf("第%2d項：%.15f¥n", i, num);
        }
        System.out.println();
        System.out.printf("Math.E：%.15f¥n", Math.E);
    }
}
```

■第5章 §2 復習問題 (p.52)

1．多重ループの利用

① 内側　② 外側

2．繰り返し処理の中断

① continue　② break　③ ラベル　④ 直前

3．処理の効率

① `int num=3; num<=50; num+=2`　② `int i=3; i<=(int)Math.sqrt(num); i+=2`

■第5章 §2 練習問題 (p.54)

1．(1)

```
ixj= 0  i=1  j=0
ixj= 1  i=1  j=1
ixj= 2  i=1  j=2
ixj= 3  i=1  j=3
ixj= 5  i=1  j=5
ixj= 0  i=2  j=0
ixj= 2  i=2  j=1
ixj= 4  i=2  j=2
ixj= 6  i=2  j=3
ixj=10  i=2  j=5
ixj= 0  i=4  j=0
ixj= 4  i=4  j=1
ixj= 8  i=4  j=2
ixj=12  i=4  j=3
ixj=20  i=4  j=5
ixj= 0  i=5  j=0
ixj= 5  i=5  j=1
ixj=10  i=5  j=2
ixj=15  i=5  j=3
ixj=25  i=5  j=5
```

(2)

```
ixj= 0  i=1  j=0
ixj= 1  i=1  j=1
ixj= 2  i=1  j=2
ixj= 3  i=1  j=3
ixj= 0  i=2  j=0
ixj= 2  i=2  j=1
ixj= 4  i=2  j=2
ixj= 6  i=2  j=3
ixj= 0  i=4  j=0
ixj= 4  i=4  j=1
ixj= 8  i=4  j=2
ixj=12  i=4  j=3
ixj= 0  i=5  j=0
ixj= 5  i=5  j=1
ixj=10  i=5  j=2
ixj=15  i=5  j=3
```

(3)

```
ixj=0  i=1  j=0
ixj=1  i=1  j=1
ixj=2  i=1  j=2
ixj=3  i=1  j=3
```

2．① `int i=1; i<20; i+=2`　② `int j=1; j<10; j++`

③ `int j=height-i; j>=0; j--`　④ `int k=1; k<=i*2-1; k++`

■第5章 §2 実習問題 (p.57)

1．【星屑バナー】

```
class StarBanner{
    public static void main(String args[]){
        for(int i=1; i<=5; i++){
            for(int j=1; j<=5; j++)
                System.out.print("☆.,_.-*'~`★");
            System.out.println();
        }
    }
}
```

2．【入力値以下のすべての素数】

```
import java.util.Scanner;

class PrimeNums{
  public static void main(String[] args){
      int upper;
      Scanner scan = new Scanner(System.in);

      System.out.print(" 求める範囲の上限を入力してください。:");
      upper = scan.nextInt();

      // 唯一の偶数の素数「２」を表示
      System.out.printf("%10d¥n", 2);

      nextNum:
      for(int num=3; num<=upper; num+=2){
          for(int i=3; i<=(int)Math.sqrt(num); i+=2){
              if(num%i == 0)
                  continue nextNum;
          }
          System.out.printf("%10d¥n", num);
      }
   }
}
```

3．【乗算表と行合計】

```
import java.util.Scanner;

class LineSum{
    public static void main(String[] args){
        Scanner scan = new Scanner(System.in);
        System.out.print(" 表の大きさを入力してください。: ");
        int scale = scan.nextInt();

        for (int i=1; i<=scale; i++){
            int sum = 0;
            for (int j=1; j<=scale; j++){
                System.out.printf("%3d  ", i*j);
                sum += i*j;
            }
            System.out.printf("%5d¥n", sum);
        }
    }
}
```

4．【山形物産展】

```
class YamagataFesta{
    public static void main(String[] args){
        int apple = 450;
        int cherry = 2125;
        int uriage = 142000;

        for(int i=0; i<=uriage/apple; i++){
            for(int j=0; j<=uriage/cherry; j++){
                if(apple*i + cherry*j == uriage)
                    System.out.printf(
                        " 出羽光 %3d 個 %, 7d 円 ¥n 佐藤誉 %3d 個 %, 7d 円 ¥n¥n",
                         i, apple*i, j, cherry*j );
            }
        }
    }
}
```

5.【不足数・過剰数・完全数】

(1)

```
class AbundantOdd{
    public static void main(String[] args){
        int upper = 10000;

        for(int i=1; i<=upper; i+=2){
            int sum = 0;
            int top = i/2;                  // ある数の約数はその数の 1/2 以下
            for(int j=1; j<=top; j+=2){ // 奇数だけ …… 偶数の約数はない
                if(i%j==0)
                    sum += j;               // 約数を集計
            }
            if(i<sum){                      // 過剰数なら表示
                System.out.printf("%7d:%7d¥n", i, sum);
            }
        }
    }
}
```

(2)

```
class SumOfDivisor{
    public static void main(String[] args){
        int upper = 100000;
        int oddSum, evenSum;                // 奇数、偶数の約数の合計
        int oddAbu = 0, oddDef = 0;         // 奇数の過剰数と不足数
        int evenAbu = 0, evenDef = 0;       // 偶数の過剰数と不足数
        int perfect = 0;                    // 完全数

        // 奇数の不足数・過剰数・完全数を集計
        for(int i=1; i<=upper; i+=2){
            oddSum = 0;
            for(int j=1; j<=(i/2); j++){
                if(i%j == 0)
                    oddSum += j;
            }
            if(i>oddSum)
                oddDef++;
            else if(i<oddSum)
                oddAbu++;
            else
                perfect++;
        }
        // 偶数の不足数・過剰数・完全数を集計
        for(int i=2; i<=upper; i+=2){
            evenSum = 0;
            for(int j=1; j<=(i/2); j++){
                if(i%j == 0)
                    evenSum += j;
            }
            if(i>evenSum)
                evenDef++;
            else if(i<evenSum)
                evenAbu++;
            else
                perfect++;
        }
        System.out.printf
            (" 奇数の過剰数：%6d¥n 奇数の不足数：%6d¥n", oddAbu, oddDef);
        System.out.printf
            (" 偶数の過剰数：%6d¥n 偶数の不足数：%6d¥n", evenAbu, evenDef);
        System.out.printf(" 完　全　数：%6d¥n", perfect);
    }
}
```

■実行結果

```
奇数の過剰数：   210
奇数の不足数： 49790
偶数の過剰数： 24585
偶数の不足数： 25411
完　全　数：     4
```

■第 5 章 §3 復習問題 (p.59)

1．while 文
　① i>=0　② i--　③ 偽
2．条件演算子
　① 3 項演算子　② 真　③ 偽　④ ?　⑤ :
3．do ～ while 文
　① 最後

■第 5 章 §3 練習問題 (p.60)

1．① int i=0; i<10; i++　② int i=0;　③ while(i<10)　④ i++;
　⑤ i=0;　⑥ do　⑦ while(i<10)

2．(1)

```
do{
    System.out.print("角度を0.00°～90.00°の範囲で入力してください。:");
    myInput = scan.nextDouble();
}while(myInput<0.00 || myInput>90.00);
```

(2)

```
do{
    System.out.print(100 以下の偶数を入力してください。:);
    myInput = scan.nextInt();
}while(myInput>=100 || myInput%2!=0);
```

(3)

```
do{
    System.out.print(8 文字以上 16 文字以下の文字列を入力してください。:);
    myInput = scan.next();
}while(myInput.length<8 || myInput.length>16);
```

3．　① ウ　② エ　③ ア

■第 5 章 §3 実習問題 (p.62)

1．【Collatz 数列の収束回数】

```
class CollatzTest{
    public static void main(String[] args){
        for(int i=2; i<=100; i++){
            int counter = 0;
            int num = i;
            do{
                if(num%2==0)
                    num /= 2;
                else
                    num = num*3+1;
                counter++;
            }while(num!=1);
            System.out.printf("%3d:%3d¥n", i, counter);
        }
    }
}
```

2．【数当てゲーム】

```
import java.util.Scanner;

public class NumQuiz{
    public static void main(String[] args){
        int rndNum;
        int inputNum;
        int counter = 0;

        rndNum = (int)(Math.random() * 1000) + 1;

        Scanner scan = new Scanner(System.in);
        System.out.println(" 数当てゲーム始めます。");

        do{
            System.out.print("1 ～ 1000 までの整数を入力してください。: ");
            inputNum = scan.nextInt();
        }while(inputNum<1 || inputNum>1000);

        while(inputNum != rndNum){
            counter++;
            if (inputNum>rndNum)
                System.out.print(" 大きすぎます。もう一度どうぞ : ");
            else
                System.out.print(" 小さすぎます。もう一度どうぞ : ");

            inputNum = scan.nextInt();
        }
        System.out.printf(" 大当たり !¥n%d 回で当たりました。", counter);
    }
}
```

3．【バーゼル問題】

```
class BaselProblem{
    public static void main(String[] args){
        int term = 0;
        double left = 0;
        double right = Math.pow(Math.PI, 2)/6;

        do{
            term++;
            left += 1/Math.pow(term, 2);
            System.out.printf(" 第 %4d 項 : %.16f¥n", term, left);
        }while(Math.abs(right - left) > 1E-2);

        System.out.println();
        System.out.printf("π²/6 = %1.15f¥n", right);
        System.out.printf(" 差     = %1.15f¥n", right-left);
        System.out.printf(
            " この数列とπ²/6 の差が 0.01 より小さくなるのは、第 %d 項です。", term);
    }
}
```

■第 6 章 §1 復習問題 (p.64)

1．配列の宣言と生成

① 要素　② 添字　③ データ型　④ 配列名　⑤ new　⑥ データ型
⑦ length　⑧ long　⑨ 0　⑩ length-1　⑪ new String[4]
⑫ char alfa[]

2．配列の初期値設定

① {"north","east","south","west"}　② false　③ null

3．配列要素の参照
① 配列名　② 添字　③ 要素数未満
4．参照型変数
① 基本データ型　② 参照型　③ 基本データ型　④参照型

■第 6 章 §1 練習問題 (p.66)

1．✓(1) 配列の初期化リテラルはブラケット（[ ]）ではなく、ブレース（{ }）で囲う。
(2) 初期化リテラルとして整数の 8 進表現、16 進表現も使用できる。
✓(3) 添字に浮動小数点数は使用できない。
(4) short 型の上限は 32767、byte 型の上限は 127 だが、要素の型にかかわらず要素数（入れ物の数）は 2147483647（int 型の上限）まで設定できる。
(5) 値が正の整数であれば、式も使用できる。
✓(6) 初期値の一括指定では、要素数を指定するとエラーになる。
✓(7) 初期値の一括指定では、宣言と初期化を分離するとエラーになる。
分離して定義したい場合は、次のように記述する。

```
b = new int[]{10000, 5000, 1000, 500, 100, 50, 10, 5, 1};
```

2．(1) ②　(2) ⑤

3．① ウ　② エ　③ イ　④ イ

?! ①② 文字列の文字数を返す length() は String クラスのメソッドである。一方、配列の要素数を保持する length は変数名（フィールド名）である。したがって、str.length() には () が付き、strChar.length には () が付かない。
③ charAt(i) は文字列の i+1 番目の文字を返す String クラスのメソッドである。したがって、charAt[i] ではなく、charAt(i)。
④ 配列 strChar の最大の添字は strChar. length-1。添字の後ろから前にアクセスしている。

4．① new int[item]　② new String[item]　③ lineForm[index]

■第 6 章 §1 実習問題 (p.69)

1．【配列要素の逆順表示】
(1)

```
class StringArray{
    public static void main(String[] args){
        String[] iceCream =
           {"chocolate", "strawberry", "vanilla", "mint", "mocha", "rumRaisin"};
        for(int i=iceCream.length-1; i>=0; i--)
            System.out.println(iceCream[i]);
    }
}
```

(2)

```
class IntArray{
    public static void main(String[] args){
        int[] powArray = new int[5];

        for(int i=0; i<powArray.length; i++)
            powArray[i] = (int)Math.pow(i+1, i+1);

        for(int i=powArray.length-1; i>=0; i--)
            System.out.println(powArray[i]);
    }
}
```

(3)

```
class DoubleArray{
    public static void main(String[] args){
        double[] sqrtArray = new double[9];

        for(int i=0; i<sqrtArray.length; i++)
            sqrtArray[i] = Math.sqrt(i+1);

        for(int i=sqrtArray.length-1; i>=0; i--)
            System.out.printf("%.5f¥n", sqrtArray[i]);
    }
}
```

(4)

```
class CharArray{
    public static void main(String[] args){
        String animals = "子丑寅卯辰巳午未申酉戌亥";
        char[] animalArray = new char[animals.length()];

        for(int i=0; i<animalArray.length; i++)
            animalArray[i] = animals.charAt(i);

        for(int i=animalArray.length-1; i>=0; i--)
            System.out.println(animalArray[i]);
    }
}
```

(5)

```
class BooleanArray{
    public static void main(String[] args){
        boolean[] booArray =
          {false, false, true, true, false, true,
                    false, true, false, false};

        for(int i=booArray.length-1; i>=0; i--)
            System.out.println(booArray[i]);
    }
}
```

2．【配列要素の最大・最小・合計・平均】

```
class ProcessArray{
    public static void main(String[] args){
        int[] myArray = {43, 66, 81, 58, 29, 52, 73};
        int max = Integer.MIN_VALUE;
        int min = Integer.MAX_VALUE;
        int total = 0;
        double ave;

        for(int i=0; i<myArray.length; i++){
            if(myArray[i] > max)
                max = myArray[i];
            if(myArray[i] < min)
                min = myArray[i];
            total += myArray[i];
        }
        ave = (double)total/myArray.length;

        System.out.printf("最大値：%3d¥n", max);
        System.out.printf("最小値：%3d¥n", min);
        System.out.printf("合　計：%3d¥n", total);
        System.out.printf("平　均：%.2f¥n", ave);
    }
}
```

3．【ピザホットの元旦の販売状況】

```
class PizzaHot{
    public static void main(String[] args){
        String[] item =
                    {"アマトリチャーナ    ", "ハワイアンデライト  ",
                     "イタリアンバジル    ", "スパイシーサルサ    ",
                     "シーフードハポン    ", "パーティーデラックス"};
        int[] price = {1750, 1890, 2150, 2280, 2415, 3800};
        int[] amount = {43, 35, 38, 23, 51, 19};
        int[] sales = new int[item.length];
        int total = 0;

        System.out.println("            本日の売上一覧");
        System.out.println(
                    "    ピザ名                  価格    数量      金額");
        for(int i=0; i<item.length; i++){
            sales[i] = price[i] * amount[i];
            total += sales[i];
            System.out.printf("%-16s%,8d%8d%,10d¥n",
                     item[i], price[i], amount[i], sales[i]);
        }
        System.out.printf("%38s%,10d", "売上合計", total);
    }
}
```

4．【力まかせの新入社員】

```
import java.util.Random;

class RollDice2{
    public static void main(String[] args){
        int[] frq = new int[11];
        int face1, face2, sum;
        Random rnd = new Random();

        for(int roll=1; roll<=10000; roll++){
            face1 = rnd.nextInt(6) + 1;
            face2 = rnd.nextInt(6) + 1;
            sum = face1 + face2;
            frq[sum-2]++;
        }
        System.out.println("出目  出現頻度");
        for(int i=0; i<frq.length; i++)
            System.out.printf("%2d¥t%5d¥n", i+2, frq[i]);
    }
}
```

■第 6 章 §2 復習問題 (p.72)

1．2 次元配列の宣言と初期値設定

① 1 次元配列　② 2 次元配列　③ 行数　④ 列数　⑤ {1,2,3}
⑥ {4,5,6}　⑦ {7,8,9}

2．配列要素の参照方法

① myArray.length　② myArray[行].length　③ j<myArray[i].length

■第 6 章 §2 練習問題 (p.73)

1. ✓ (1) int[][] myInt = {{3, 4}, {5, 6}};
あるいは、int[][] myInt = new int[][]{{3, 4}, {5, 6}};
✓ (3) 添字に浮動小数点数は使用できない。
✓ (4) byte[][] myByte = new byte[3][2];

2.　　i　　j　　要素
①〔 5 〕〔 4 〕〔 1 〕
②〔 8 〕〔 5 〕〔 3 〕
③〔 9 〕〔10 〕〔 9 〕

3. (1) ① new int[9][9]　② plate[i].length　③ plate[2][3] = 9
④ plate[2][5] = 4　⑤ plate[6][3] = 7　⑥ plate[6][5] = 6
⑦ plate[i].length
(2) ① エ　② ア　③ イ　④ ウ


①は左右反転、②は行・列変換、③は天地反転、④は中央 [4][4] の点対称である。

■第 6 章 §2 実習問題 (p.76)

1.【2 次元配列による矢印表示】

(1)

```
class Arrow01{
    public static void main(String[] args){
        char[][] arrow = new char[8][8];
        for(int row=0; row<arrow.length; row++){
            for(int col=0; col<arrow[0].length; col++){
                if(row == col || row == 7 || col == 7)
                    arrow[row][col] = '*';
            }
        }
        // 表示
        for(int row=0; row<arrow.length; row++){
            for(int col=0; col<arrow[0].length; col++){
                System.out.print(arrow[row][col]);
            }
            System.out.println();
        }
    }
}
```

(2)

```
class Arrow02{
    public static void main(String[] args){
        char[][] arrow = new char[8][8];
        for(int row=0; row<arrow.length; row++){
            for(int col=0; col<arrow[0].length; col++){
                if(row + col == 7 || row == 7 || col == 0)
                    arrow[row][col] = '*';
            }
        }
        // 表示
        for(int row=0; row<arrow.length; row++){
            for(int col=0; col<arrow[0].length; col++){
                System.out.print(arrow[row][col]);
            }
            System.out.println();
        }
    }
}
```

2．【サイの出目の確率計算】

```
class TwoDice{
    public static void main(String[] args){
        int[][] twoDice = new int[6][6];
        int[] deme = new int[11];    // 出目合計 2 ～ 12、合計 11 個の記憶場所
        int counter = 0;

        for(int i=1; i<=6; i++){
            for(int j=1; j<=6; j++){
                // 縦横の合計を記憶、出目 (1 ～ 6) より添字 (0 ～ 5) は 1 小さい。
                //twoDice[0][0] には 2、twoDice[5][5] には 12 が記憶される。
                twoDice[i-1][j-1] = i + j;
            }
        }
        for(int i=0; i<twoDice.length; i++){
            for(int j=0; j<twoDice[0].length; j++){
                // 出目合計 (2 ～ 12) から 2 を引いてそれを添字 (0 ～ 10) として利用
                //deme[0] は出目 2 の出現回数を記憶しその値は 1、deme[5] は出目 7
                // の出現回数を集計しその値は 6 になる。
                deme[twoDice[i][j]-2]++;
                counter++;       // すべての目の出現回数を集計、合計は 6 × 6 の 36
            }
        }
        System.out.println(" ２つのサイの出目の合計 ");
        System.out.print("  ");
        for(int i=1; i<=6; i++)
            System.out.printf("%3d", i);         //1 行目を表示
        System.out.println();
        for(int i=0; i<twoDice.length; i++){
            System.out.printf(" %d", i+1);       //1 列目を表示
            for(int j=0; j<twoDice[0].length; j++){
                System.out.printf("%3d", twoDice[i][j]);  // 合計値を表示
            }
            System.out.println();
        }
        System.out.println();

        System.out.println(" ２つのサイの出目の合計の出現確率 ");
        for(int i=2; i<=12; i++)
            System.out.printf(
                "%2d:  %6.2f%%¥n", i, (double)deme[i-2]/counter*100);
    }
}
```

3．【耕地面積集計表の作成】

```
class LandArea{
    public static void main(String[] args){
        String[] prefName =
                 {" 青森 "," 岩手 "," 宮城 "," 秋田 "," 山形 "," 福島 "};
        int area[][] = {
                         {9644, 840, 737, 0},
                         {15279, 960, 586, 0},
                         {6862, 1108, 259, 0},
                         {11636, 1310, 200, 0},
                         {6652, 970, 266, 0},
                         {13782, 1059, 451, 0}
                        };
        double[] farmRatio = new double[prefName.length];    // 耕地率
        double[] paddyRatio = new double[prefName.length];   // 水田率
        int[] total = new int[area[0].length];
        int[] average = new int[area[0].length];
```

（次ページへ続く）

```
        // ヘッダーの表示
        System.out.printf("%25s¥n", " 東北地方の耕地面積集計表 ");
        System.out.println(
            " 県名　県面積　田面積　畑面積　耕地面積　耕地率　水田率 ");

        // 県ごとに 1 行ずつ処理
        for(int i=0; i<area.length; i++){
            area[i][3] = area[i][1] + area[i][2];   // 耕地面積
            farmRatio[i] = (double)area[i][3]/area[i][0] * 100; // 耕地率
            paddyRatio[i] = (double)area[i][1]/area[i][3] * 100;  // 水田率
            // 表示
            System.out.print(prefName[i]);
            for(int j=0; j<area[0].length; j++)
                System.out.printf("%,8d", area[i][j]);
            System.out.printf("%8.2f%%%8.2f%%¥n",
                              farmRatio[i], paddyRatio[i]);
        }
        // 合計と平均の計算
        for(int j=0; j<area[0].length; j++){
            for(int i=0; i<area.length; i++){
                total[j] += area[i][j];
            }
            average[j] = total[j]/area.length;
        }
        // 合計の表示
        System.out.print(" 合計 ");
        for(int j=0; j<area[0].length; j++)
            System.out.printf("%,8d", total[j]);
        System.out.println();
        // 平均の表示
        System.out.print(" 平均 ");
        for(int j=0; j<area[0].length; j++)
            System.out.printf("%,8d",average[j]);
        System.out.println();
    }
}
```

String 型（県名） int 型（県面積　田面積　畑面積　耕地面積） Double 型（耕地率　水田率）の配列を横に連結して表を作成している。異なるデータ型の集まりをまとめて 1 つの処理単位（データ型）として取り扱う方法は第 8 章以降で学習する。

4.【パラソルの回転】

(1)

```
01  class Rotation01{
02      public static void main(String args[]){
03          // 7×7の2次元配列にキャラクタでイメージを初期設定
04          char[][] myParasol =  {
05                  {' ',' ',' ',' ','*',' ',' ',' ',' '},
06                  {' ',' ',' ','*',' ','*',' ',' ',' '},
07                  {' ',' ','*',' ','*',' ','*',' ',' '},
08                  {' ',' ',' ',' ',' ','*',' ',' ',' '},
09                  {' ',' ',' ',' ',' ','*',' ',' ',' '},
10                  {' ',' ','*',' ',' ','*',' ',' ',' '},
11                  {' ',' ',' ','*',' ',' ',' ',' ',' '}
12                          };
13          char[][] newParasol = new char[7][7];
14
15          // 原図の表示
16          System.out.println(" 原図 ");
17          System.out.println();
18
19          for(int i=0; i<myParasol.length; i++){
20              for(int j=0; j<myParasol[i].length; j++){
21                  System.out.print(myParasol[i][j]);
22              }
23              System.out.println();
24          }
```

（次ページへ続く）

```
25            System.out.println();
26
27            // 時計回り 90° 回転
28            for(int i=0; i<myParasol.length; i++){
29                for(int j=0; j<myParasol[i].length; j++){
30                    newParasol[j][6-i] = myParasol[i][j];
31                }
32            }
33
34            // 回転した図の表示
35            System.out.println(" 変化図 ");
36            System.out.println();
37
38            for(int i=0; i<newParasol.length; i++){
39                for(int j=0; j<newParasol[i].length; j++){
40                    System.out.print(newParasol[i][j]);
41                }
42                System.out.println();
43            }
44        }
45    }
```

(2)　(1) の 27 ～ 32 行を次の 6 行で差し替える。

```
27            // 左右反転
28            for(int i=0; i<myParasol.length; i++){
29                for(int j=0; j<myParasol[i].length; j++){
30                    newParasol[i][6-j] = myParasol[i][j];
31                }
32            }
```

(3)　(1) の 27 ～ 32 行を次の 6 行で差し替える。

```
27            //180° 回転
28            for(int i=0; i<myParasol.length; i++){
29                for(int j=0; j<myParasol[i].length; j++){
30                    newParasol[6-i][6-j] = myParasol[i][j];
31                }
32            }
```

■第 6 章 §3 復習問題 (p.79)

1．バブルソート

① myArray.length-1　② isSwap = false　③ myArray[j-1]
④ swapCounter++　⑤ true　⑥ false　⑦ 8　⑧ 45　⑨ 190
⑩ 61,49,35,23,12　⑪ 2 乗

2．Random クラスのメソッド

① 1 ～ 6　② 4, 7, 10, 13, 16

3．同一乱数系列の生成

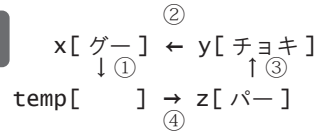
① long

■第 6 章 §3 練習問題 (p.81)

1．　誤っているもの

(1)　配列の要素数は「配列名 .length」である。
(2)　配列の添字の範囲外の要素にアクセスすると、「実行時に」
　ArrayIndexOutOfBoundsException というエラー（例外）が発生する。
(4)　boolean 型の生成時の初期値は false である。
(5)　変数で指定できる。

2．ア



①がないと、②の段階でグーはチョキで上書きされ、永遠に失われる。
　変数間のデータ交換には、一時的待避変数が必要なことを理解する。

3．① `int min = i`　② `int j=i+1; j<myArray.length; j++`　③ `myArray[min]`
④ `swapCounter++`　⑤ `compCounter++`

■第 6 章 §3 実習問題 (p.83)

1．【じゃんけんの確率】

(1)

```
import java.util.Random;

class Janken{
    public static void main(String[] args){
        int guu = 43;
        int choki = guu + 25;
        Random rnd = new Random();

        System.out.println(" じゃんけん ");
        int rndInt = rnd.nextInt(100) + 1;
        if(rndInt<=guu)
            System.out.println(" グー !");
        else if(rndInt<=choki)
            System.out.println(" チョキ !");
        else
            System.out.println(" パー !");
    }
}
```

(2)

```
import java.util.Random;

class Jankenpon{
    public static void main(String[] args){
        int guu = 43;
        int choki = guu + 25;

        int guuCounter = 0;
        int chokiCounter = 0;
        int paaCounter = 0;
        Random rnd = new Random();

        for(int i=1; i<=1000; i++){
            int rndInt = rnd.nextInt(100) + 1;
            if(rndInt<=guu)
                guuCounter++;
            else if(rndInt<=choki)
                chokiCounter++;
            else
                paaCounter++;
        }
        System.out.println(" グー・チョキ・パーの出現率 ");
        System.out.println("      回数    出現率 ");
        System.out.printf("%s:%d%8.1f%%¥n",
            " グー  ", guuCounter, 100*(double)guuCounter/1000);
        System.out.printf("%s:%d%8.1f%%¥n",
            " チョキ ", chokiCounter, 100*(double)chokiCounter/1000);
        System.out.printf("%s:%d%8.1f%%¥n",
            " パー  ", paaCounter, 100*(double)paaCounter/1000);
    }
}
```

2．【小数配列の整列】

(1)

```
class DoubleSort01{
    public static void main(String[] args){
        double[] myArray = {
            3.02, 2.97, 3.31, 4.01, 3.66, 3.52, 3.41, 2.92, 3.34, 2.84, 3.27};
        for(int i=0; i<myArray.length-1; i++){
            for(int j=1; j<myArray.length-i; j++){
                if(myArray[j-1]<myArray[j]){
                    double temp = myArray[j-1];
                    myArray[j-1] = myArray[j];
                    myArray[j] = temp;
                }
            }
        }
        for(int i=0; i<myArray.length; i++)
            System.out.printf("%6.2f", myArray[i]);
    }
}
```

(2)

```
import java.util.Random;

class DoubleSort02{
    public static void main(String[] args){
        double bottom = 2.80;
        double top = 5.05;
        double[] myDouble = new double[15];
        Random rnd = new Random();
        for(int i=0; i<myDouble.length; i++)
            myDouble[i] = rnd.nextDouble() * (top - bottom) + bottom;
        for(int i=0; i<myDouble.length-1; i++){
            for(int j=1; j<myDouble.length-i; j++){
                if(myDouble[j-1] > myDouble[j]){
                    double temp = myDouble[j];
                    myDouble[j] = myDouble[j-1];
                    myDouble[j-1] = temp;
                }
            }
        }
        for(int i=0; i<myDouble.length; i++)
            System.out.printf("%.2f  ", myDouble[i]);
    }
}
```

倍精度浮動小数点数の乱数を発生させる nextDouble() メソッドは、nextInt() メソッドと異なり引数をとらない。したがって、一定範囲の乱数を生成するには、上記のように nextDouble() * 生成範囲 + 移動値　とする。

3．【ハーブリストの整列】

```
class HerbList{
    public static void main(String[] args){
        String[] herbList = {"thyme", "rosemarry", "paseley", "basil",
                "coriander", "seiji", "laurel", "oregano", "mint", "chicory"};
        for(int i=1; i<herbList.length; i++){
            for(int j=0; j<herbList.length-i; j++){
                if(herbList[j].compareTo(herbList[j+1])>0){
                    String temp = herbList[j];
                    herbList[j] = herbList[j+1];
                    herbList[j+1] = temp;
                }
            }
        }
        for(int i=0; i<herbList.length; i++)
            System.out.println(herbList[i]);
    }
}
```

4．【2 つのソートの比較・交換回数】

(1)

```
import java.util.Random;

class Bubble100{
    public static void main(String[] args){
        int idx = 100;
        int[] myArray = new int[idx];
        int compCounter = 0;
        int swapCounter = 0;
        Random rnd = new Random(123456789L);

        for(int i=0; i<myArray.length; i++)
            myArray[i] = rnd.nextInt(idx*9) + idx;

        for(int i=1; i<myArray.length; i++){
            boolean isSwap = false;
            for(int j=0; j<myArray.length-i; j++){
                compCounter++;
                if(myArray[j] > myArray[j+1]){
                    int temp = myArray[j];
                    myArray[j] = myArray[j+1];
                    myArray[j+1] = temp;
                    isSwap = true;
                    swapCounter++;
                }
            }
        }
        System.out.println("BubbleSort･･･Done!");
        for(int i=0; i<myArray.length; i++)
            System.out.printf((i+1)%10==0?"%5d¥n":"%5d", myArray[i]);
        System.out.printf("比較回数：%d¥n", compCounter);
        System.out.printf("交換回数：%d¥n", swapCounter);
    }
}
```

(2)

```
import java.util.Random;

class Select100{
    public static void main(String[] args){
        int idx = 100;
        int[] myArray = new int[idx];
        int compCounter = 0;
        int swapCounter = 0;
        Random rnd = new Random(123456789L);

        for(int i=0; i<myArray.length; i++)
            myArray[i] = rnd.nextInt(idx*9) + idx;

        for(int i=0; i<myArray.length-1; i++){
            boolean isSwap = false;
            int min = i;
            for(int j=i+1; j<myArray.length; j++){
                compCounter++;
                if(myArray[min] > myArray[j]){
                    int temp = myArray[min];
                    myArray[min] = myArray[j];
                    myArray[j] = temp;
                    swapCounter++;
                    isSwap = true;
                }
            }
        }
        System.out.println("SelectionSort･･･Done!");
        for(int i=0; i<myArray.length; i++)
            System.out.printf((i+1)%10==0?"%5d¥n":"%5d", myArray[i]);
        System.out.printf("比較回数：%d¥n", compCounter);
        System.out.printf("交換回数：%d¥n", swapCounter);
    }
}
```

バブルソート、選択ソートとも交換回数は比較回数のほぼ 2 分の 1 で、交換回数は選択ソートのほうが少ない。このプログラムの変数 idx に、たとえば 1000 を初期設定すると、1000 ～ 9999 の範囲の数を 1000 生成して並べ替えるので、idx の値をいろいろ変えて回数がどう変わるか、調べてみるとよい。
なお、比較回数ではなく処理時間の比較は、『学習編』例題 22「ソートアルゴリズムの実行時間比較」を参照のこと。

■第 7 章 §1 復習問題 (p.86)

1．メソッド利用のメリット
① 開発効率　② 再利用性
2．メソッド定義の一般書式
① 引数　② 返却値　③アクセス修飾子
3．メソッドの修飾子
① public　② protected　③ private　④ public　⑤静的メソッド
⑥ 非静的メソッド　⑦ new 演算子　⑧ 静的　⑨ 非静的　⑩ オブジェクト変数名
⑪ static 修飾子
4．引数によるデータの受け渡し
① 実引数　② 仮引数　③ 式　④ 変数
5．ローカル変数とフィールド
① ローカル変数　② メソッド　③ フィールド
6．フィールドの修飾子
① private　② クラス変数　③ インスタンス変数
7 変数の有効範囲
① スコープ　② 0　③ false　④ null　⑤ ローカル変数

■第 7 章 §1 練習問題 (p.89)

1． (2)　(5)　(7)
(1) 同じクラス内であれば、main() 以外のメソッドからも呼び出すことができる。
(3) 実引数として、リテラル、変数、式が使用できる。
(4) ブール型のフィールドには false が生成時に初期値設定されるが、ブール型のローカル変数には特定の値が設定されるわけではないので、式の右辺として使うとコンパイル時にエラーとなる。
(6) フィールドとローカル変数が同じ名前の場合、メソッドはローカル変数を優先して参照する。
(8) 他のメソッドから main() メソッドは呼び出せない。
(9) 静的メソッドはメモリに常駐しているので呼び出せる。

2．(1) ① 3　② 18　(2) ① 3　② true　③ false　④ true
(1) フィールドの x（=2）、setX() メソッドの x（=3）および setY() メソッドの仮引数 x はすべて異なる変数で関連はない。ローカル変数の 2 つの x の有効範囲は、それぞれそのメソッド内だけである。フィールド x の有効範囲はこのクラス内であるが、setX()、setY() では同名のローカル変数 x が定義されていて、それらが優先参照されるので、これらのメソッド内では機能しない。したがって、setX() メソッド内の printf 文では、ローカル変数の値（=3）がそのまま表示される。クラス変数 sum の計算結果は 10 + 3 × 3 − 1 = 18 である。
(2) ブール型配列 boo[] は、生成段階で各要素が false に設定されている。はじめの setTrue() の呼び出しで、boo[0]=true になり、2 回目の呼び出しで boo[2]=true になる。checkWriter() メソッド内の if 文の条件判定では || より && が優先度が高いので、boo[1]&&boo[2] がまず判定され、false && true → false に、次に boo[0](true) || false → true になり、counter には 1 が加算される。

3．① static　② boolean isSwap　③ lineUp()　④ swap(min, j)
⑤ showLine(i, j)

このプログラムは、『演習編』p.83 の SelectionSortTest クラスの main() メソッドでまとめていた選択ソート機能を、整列、要素交換、画面表示の 3 つのメソッドに分けて処理している。
①②配列 myArray は、これらの 3 つのメソッドで共通して使用されているので、クラス変数（静的フィールド）として定義する必要がある。また、boolean 型の変数 isSwap が、2 つのメソッドで使用されているので、同様にクラス変数として定義する必要がある。
③⑤このプログラムでは、main() メソッドから lineUp() メソッドを呼び出し、lineUp() メソッドから swap() メソッドと showArray() メソッドを呼び出している。
④選択ソートは対象となる要素群のなかから最小のデータを順に選択して並べていく整列法である。

■第 7 章 §1 実習問題 (p.92)

1．【メソッドによる図形表示】

(1)

```
import java.util.Scanner;

class StarRect{
    public static void main(String[] args){
        int width;
        Scanner scan = new Scanner(System.in);

        System.out.print("1 辺の長さを指定してください。:");
        width = scan.nextInt();

        fillRect(width);
    }
    static void fillRect(int width){
        for(int i=0; i<width; i++){
            for(int j=0; j<width; j++){
                System.out.print('*');
            }
            System.out.println();
        }
    }
}
```

(2)

```
import java.util.Scanner;

class CharRect{
    public static void main(String[] args){
        int width;
        char chr;
        Scanner scan = new Scanner(System.in);

        System.out.print("1 辺の長さを指定してください。:");
        width = scan.nextInt();
        System.out.print(" 文字種を指定してください。:");
        chr = scan.next().charAt(0);
        fillRect(chr, width);
    }
    static void fillRect(char chr, int width){
        for(int i=0; i<width; i++){
            for(int j=0; j<width; j++){
                System.out.print(chr);
            }
            System.out.println();
        }
    }
}
```

2.【カードのシャッフルと整列】

```
import java.util.Random;

class DealCard{
    static String card[] =
        {"Ace","Deuce","Three","Four","Five","Six","Seven",
         "Eight","Nine","Ten","Jack","Queen","King"};
    public static void main(String[] args){
        show(" 元のカード ");
        shuffle();
        show(" シャッフルしました。");
        sort();
        show(" 辞書順でソートしました。");
    }
    static void shuffle(){
        Random rnd = new Random();
        for(int i=0; i<card.length; i++)
            swap(i, rnd.nextInt(13));
    }
    static void sort(){
        for(int i=1; i<card.length; i++){
            for(int j=0; j<card.length-i; j++){
                if(card[j].compareTo(card[j+1])>0)
                    swap(j, j+1);
            }
        }
    }
    static void swap(int left, int right){
        String temp = card[left];
        card[left] = card[right];
        card[right] = temp;
    }
    static void show(String msg){
        System.out.println(msg);
        for(int i=0; i<card.length; i++)
            System.out.printf("%s ", card[i]);
        System.out.println();
    }
}
```

3.【メソッドによる図形の回転】

```
import java.util.Scanner;

public class ParasolMethod{
    static char[][] parasol =  {
                    {' ',' ',' ',' ','*',' ',' ',' ',' '},
                    {' ',' ',' ','*','*','*',' ',' ',' '},
                    {' ',' ','*','*','*','*','*',' ',' '},
                    {' ',' ',' ',' ','*',' ',' ',' ',' '},
                    {' ',' ',' ',' ','*',' ',' ',' ',' '},
                    {' ',' ','*',' ','*',' ',' ',' ',' '},
                    {' ',' ',' ','*',' ',' ',' ',' ',' '}
                    };
    static char[][] newParasol = new char[7][7];
    static int lastIdx = parasol.length-1;

    public static void main(String[] args){
        Scanner scan = new Scanner(System.in);
        int inputNum;

        //menu の表示
        do{
            System.out.println(" 処理を番号で指定してください (1 ～ 3)。");
            System.out.println("1. 90° 右回転 ");
            System.out.println("2. 左右反転 ");
            System.out.println("3. 上下反転 ");
            inputNum = scan.nextInt();
        }while(inputNum<1 || inputNum>3);
```

(次ページへ続く)

```
        showParasol("原図", parasol);

        switch(inputNum){
            case 1:
                quarterRight();
                break;
            case 2:
                right2left();
                break;
            case 3:
                upset();
                break;
            default:
                break;
        }
    }
    // 図の表示
    static void showParasol(String str, char[][] parasol){
        System.out.println(str);
        for(int i=0; i<parasol.length; i++){
            for(int j=0; j<parasol[i].length; j++){
                System.out.print(parasol[i][j]);
            }
            System.out.println();
        }
        System.out.println();
    }
    // 右 90° 回転
    static void quarterRight(){
        for(int i=0; i<parasol.length; i++){
            for(int j=0; j<parasol[i].length; j++){
                newParasol[j][lastIdx-i]  = parasol[i][j];
            }
        }
        showParasol("右 90° 回転", newParasol);
    }
    // 左右反転
    static void right2left(){
        for(int i=0; i<parasol.length; i++){
            for(int j=0; j<parasol[i].length; j++){
                newParasol[i][lastIdx-j]  = parasol[i][j];
            }
        }
        showParasol("左右反転", newParasol);
    }
    // 上下反転
    static void upset(){
        for(int i=0; i<parasol.length; i++){
            for(int j=0; j<parasol[i].length; j++){
                newParasol[lastIdx-i][j] = parasol[i][j];
            }
        }
        showParasol("上下反転", newParasol);
    }
}
```

◆7 × 7 の図の回転

上下反転	[6-i][j] = [i][j];
右 90° 回転	[j][6-i] = [i][j];
180 度回転	[6-i][6-j] = [i][j]
左右反転	[i][6-j] = [i][j];
左 90° 回転	[6-j][i] = [i][j];

◆座標移動の一般形

上下反転を例にすれば、　[myArray.length-1-i][j] = [i][j];

■第 7 章 §2 復習問題 (p.94)

1．コマンドライン引数
① 空白
2．メソッドの返却値
① void ② return
3．ラッパークラス
① 基本データ型 ② 参照型 ③ ラッパークラス ④ Integer
⑤ Double ⑥ Character ⑦ Boolean ⑧ Integer ⑨ parseInt
⑩ toString() ⑪ 文字列 ⑫ Integer ⑬ valueOf
⑭ parseInt(String) ⑮ toString() ⑯ intValue()
⑰ Long.toString()
4．文字列処理のためのクラス
① StringBuilder ② new StringBuilder() ③ Long.parseLong
④ append() ⑤ length() ⑥ reverse() ⑦ toString()

■第 7 章 §2 練習問題 (p.97)

1. ① クラス ② Long.MAX_VALUE ③ インスタンス ④ オブジェクト変数
⑤ private ⑥ スコープ ⑦ StringBuilder ⑧ parseInt(String s)
⑨ toString(int i) ⑩ Integer ⑪ 文字列リテラル ⑫ Charactor
⑬ nextLine() ⑭ 空白

2． (1) ① 2 ② 8
(2) ① 15 ② 3 ③ 4 ④ 10

(1) 3 つの異なる変数 x の有効範囲と参照順に注意する。
(2) al[1] = 4 × 1　al[2] = 5 × 2　sum = 1 + 4 + 10

```
                                        ↑
                                    static int sum = 1
```

3． (注) 空欄に入る語句はイタリックで示した。

(1)

```
class GetAreaTest{
    public static void main(String[] args){
        int testData = 15;
        System.out.printf("半径 %d の円の面積は %.4f です。¥n",
                                    testData, getArea(testData));
    }
    static double getArea(int radius){
        return Math.PI * Math.pow(radius, 2);
    }
}
```

(2)

```
import java.util.Random;

class Roll2DiceTest{
    public static void main(String[] args){
        System.out.printf("出目は %d です。", roll2Dice());
    }
    static int  roll2Dice(){
        Random rnd = new Random();
        int die1 = rnd.nextInt(6) + 1;
        int die2 = rnd.nextInt(6) + 1;
        return die1 + die2;
    }
}
```

(3)

```
class GetAverageTest{
    public static void main(String[] args){
        int[] testArray = {23, 45, 56};
        System.out.printf("testArray の要素の平均値は %.4f です。",
                    getAverage(testArray));
    }
    static double getAverage(int[] intArray){
        int sum = 0;
        for(int i=0; i<intArray.length; i++)
            sum += intArray[i];
        return (double)sum / intArray.length;
    }
}
```

(4)

```
class AddedUpTest{
    public static void main(String[] args){
        int p = Integer.parseInt(args[0]);
        int q = Integer.parseInt(args[1]);
        System.out.printf(
            "%d ～ %d 間の整数の総和は %d です。", p, q, addedUp(p, q));
    }
    static int addedUp(int p, int q){
        int sum = 0;
        if(p<q)
            for(int i=p; i<=q; i++)
                sum += i;
        else
            for(int i=p; i>=q; i--)
                sum += i;
        return sum;
    }
}
```

(5)

```
class GetFactTest{
    public static void main(String[] args){
        int testData = 17;
        System.out.printf(
            "%d の階乗は %,d です。", testData, getFact(testData));
    }
    static long getFact(int num){
        long fact = 1;
        for(int i=1; i<=num; i++)
            fact *=(long)i;
        return fact;
    }
}
```

(6)

```
class GetGradeTest{
    public static void main(String[] args){
        int testData = 68;
        System.out.printf("得点：%d¥n%s", testData, getGrade(testData));
    }
    static String getGrade(int point){
        String message;
        if(point>=90)
            message = "Excellent!";
        else if(point>=70)
            message = "Nice work!";
        else if(point>=50)
            message = "Keep up good work!";
        else
            message = "Don't give up!";
        return message;
    }
}
```

■第 7 章 §2 実習問題 (p.100)

1.【文字列のコマンドライン入力】

(1)

```
class ArgsTest{
    public static void main(String[] args){
        for(int i=0; i<args.length; i++)
            System.out.printf("args[%d] : %s¥n", i, args[i]);
    }
}
```

(2)

```
class  ArgsCalc{
    public static void main(String[] args){
        int max = 0;
        int min = Integer.MAX_VALUE;
        int sum = 0;
        int[] inputVal = new int[args.length];
        for(int i=0; i<args.length; i++)
            inputVal[i] = Integer.parseInt(args[i]);
        for(int i=0; i<args.length; i++){
            sum += inputVal[i];
            if(max<inputVal[i])
                max = inputVal[i];
            if(min>inputVal[i])
                min = inputVal[i];
        }
        double avg = (double)sum/args.length;
        System.out.printf("最大値 :%d¥n", max);
        System.out.printf("最小値 :%d¥n", min);
        System.out.printf("合　計 :%d¥n", sum);
        System.out.printf("平　均 :%.2f¥n", avg);
    }
}
```

2.【素数の出現頻度】

(1)

```
class PrimeBand{
    public static void main(String[] args){
        int range = 100000;      //対象範囲
        int margin = 10000;      //区切り
        int[] freq = new int[range/margin];
        freq[0] = 1;         //「2」をあらかじめカウント
        int index = 0;

        //区切りごとに素数をカウント
        for(int i=3; i<range; i++){
            if(i%margin == 0)
                index++;
            if(isPrime(i))
                freq[index]++;
        }
        //表示
        for(int i=0; i<freq.length; i++)
            System.out.printf(
                "%6d ～ %6d:%6d¥n", i*margin, (i+1)*margin-1, freq[i]);
    }
    static boolean isPrime(int num){
        //2 以外の偶数は素数ではない
        if(num%2 == 0)
            return false;
        //奇数について約数の有無を調査
        for(int i=3; i*i<=num; i+=2)
            if(num%i == 0)
                return false;
        //調査終了→約数なし・素数確定
        return true;
    }
}
```

(2)

```
import java.util.Scanner;

public class TwoPrimes{
    public static void main(String[] args){
        int num;
        Scanner scan = new Scanner(System.in);
        System.out.print(" 整数を入力してください。: ");
        num = scan.nextInt();
        System.out.printf(
            "%d 未満の最大の素数は %d です。¥n", num, lowerMax(num));
        System.out.printf(
            "%d を超える最小の素数は %d です。¥n", num, upperMin(num));

    }
    static int lowerMax(int num){
        if(num%2 == 0)
            num--;
        else
            num-=2;

        while(!isPrime(num))
            num-=2;

        return num;
    }
    static int upperMin(int num){
        if(num%2 == 0)
            num++;
        else
            num+=2;
        while(!isPrime(num))
            num+=2;

        return num;
    }
    static  boolean isPrime(int num){
        //2 以外の偶数は素数ではない
        if(num == 2)
            return true;
        else if(num%2 == 0)
            return false;
        // 奇数について約数の有無を調査
        for(int i=3; i*i<=num; i+=2)
            if(num%i == 0)
                return false;
        // 調査終了 ··· 約数なし・素数確定
        return true;
    }
}
```

3.【暗証番号の暗号化】

```
import java.util.Scanner;

class Encoder4Paris{
    public static void main(String[] args){
        String idCode;
        Scanner scan  = new Scanner(System.in);

        do{
            System.out.print(" 4 けたの数字を入力してください。:");
            idCode = scan.next();
        }while(idCode.length() != 4);

        System.out.printf(" 変換後の番号は、%s です。", encode(idCode));
    }
```

（次ページへ続く）

```
    static String encode(String myCode){
        String digit;                       // 番号１けた
        int[] digits = new int[4];          // 番号４けた（配列）
        String shadow = "";                 // 変換文字列

        // 各けたに分解し、７を加え、11 で割った余りを求める
        for(int i=0; i<myCode.length(); i++){
            digit = myCode.substring(i, i+1);
            digits[i] = (Integer.parseInt(digit) + 7) % 11;
            if (digits[i] == 10)    //10 → 6
                digits[i] = 6;
         }
        // １と３けた目、２と４けた目を入れ替える
        for(int i=0; i<digits.length/2; i++){
            int temp = digits[i];
            digits[i] = digits[i+2];
            digits[i+2] = temp;
        }
        //int 配列を文字列に変換
        for(int i=0; i<digits.length; i++)
            shadow += Integer.toString(digits[i]);

        return shadow;
    }
}
```

4.【シェルピンスキーのガスケット】

```
class Gasket{
    static int scale = 33;
    static boolean[][] point = new boolean[scale][scale];
    public static void main(String[] args){
        //2 次元配列をクリア
        for(int i=0; i<point.length; i++)
            for(int j=0; j<point[i].length; j++)
                point[i][j] = false;

        //(0,1) に true を設定
        point[0][1] = true;

        //(1,1) より右下の各座標に true、false をセット
        for(int i=1; i<point.length; i++){
            for(int j=1; j<point[i].length; j++){
                point[i][j] = isPrint(i, j);
            }
        }
        // ２次元配列を表示
        for(int i=0; i<point.length; i++){
            for(int j=0; j<point[i].length; j++){
                if (point[i][j])
                    System.out.print("*");
                else
                    System.out.print(" ");
            }
            System.out.println();
        }
    }
    static boolean isPrint(int p, int q){
        // 左上と真上の排他的論理和をとる
        if(point[p-1][q-1]^point[p-1][q])
            return true;
        else
            return false;
    }
}
```

5.【クラップスゲーム】

```
class Craps{
    public static void main(String args[]){
        int sumOfDice = 0;
        int pointOfDice = 0;
        int counter = 1;
        final int WON = 0, LOST = 1, CONT = 2;
        int result = CONT;

        //FirstRoll
        System.out.println(" サイを振ります。1 回目です。¥n");
        sumOfDice = rollDice();

        switch(sumOfDice){
            case 7: case 11:
                result = WON;
                break;
            case 2: case 3: case 12:
                result = LOST;
                break;
            default:
                result = CONT;
                pointOfDice = sumOfDice;
                System.out.printf(
                    "%d が出ればあなたの勝ち、¥n7 が出ればわたしの勝ちです。¥n¥n",
                     pointOfDice);
                break;
        }
        //Continue
        while (result == CONT){
            System.out.printf(
                 " サイを振ります。:%d 回目です。¥n", ++counter);
            sumOfDice = rollDice();
            if (sumOfDice == pointOfDice)
                result = WON;
            else
                if (sumOfDice == 7)
                    result = LOST;
        }
        System.out.println(
            result == WON ? " あなたの勝ちです。" : " わたしの勝ちです。");
    }

    static int rollDice(){
        int die1 = (int)(Math.random() * 6) + 1;
        int die2 = (int)(Math.random() * 6) + 1;
        int workSum = die1 + die2;
        System.out.printf(
            " 出目は %d と %d、合計 %d です。¥n¥n", die1, die2, workSum);
        return workSum;
    }
}
```

■第 8 章 §1 復習問題 (p.104)

1．手続き型からオブジェクト指向へ
① 手続き ② 型 ③ インスタンス ④ オブジェクト

2．クラスの基本構成
① フィールド ② コンストラクタ ③ メソッド

3．コンストラクタ
① クラス名 ② 返却値 ③ デフォルトコンストラクタ ④ new 演算子

4．オブジェクト変数 this
① オブジェクト変数 ② インスタンス ③ 非静的メンバー ④ フィールド名

■第 8 章 §1 練習問題 (p.105)

1．(2) (4)


インスタンスを生成し、そのメソッドを呼び出す一般的な記述は (4) であるが、(2) のように、オブジェクト変数を省略した書き方もできる。

2．(1) ① `this.x = x`
② `Simple simple = new Simple(x)`
(2) ③ `asts.SquareOfAsterisks(side)`
④ `width = s`
(3) ⑤ `public BooSample(boolean isBoo)`
⑥ `BooSample sample = new BooSample(boo)`

3．(1)

```
class MyRectangle{
    int length;
    int height;
    double perimeter;
    int area;

    public MyRectangle(int length, int height){
        this.length = length;
        this.height = height;
    }

    public int getLength(){
        return length;
    }
    public int getHeight(){
        return height;
    }
    public double getPerimeter(){
        return Math.sqrt(Math.pow(length, 2) + Math.pow(height, 2));
    }
    public int getArea(){
        return length * height;
    }
}
```

(2)

```
class MyPizza{
    String item;
    int price;
    int amount;
    int sales;

    public MyPizza(String item, int price, int amount){
        this.item = item;
        this.price = price;
        this.amount = amount;
    }
    public String getItem(){
        return item;
    }
    public int getPrice(){
        return price;
    }
    public int getAmount(){
        return amount;
    }
    public int getSales(){
        return price * amount;
    }
}
```

■第 8 章 §1 実習問題 (p.108)

1.【長方形クラスの実行プログラム】

```
import java.util.Scanner;

public class MyRectangleTest{
    public static void main(String[] args){
        int length;
        int height;
        Scanner scan = new Scanner(System.in);

        System.out.print("長方形の底辺の長さを入力してください。:");
        length = scan.nextInt();
        System.out.print("長方形の高さを入力してください。:");
        height = scan.nextInt();

        MyRectangle myRec = new MyRectangle(length, height);

        System.out.printf(
            "%d×%d の長方形の対角線の長さは %.2f で、面積は %d です。",
            myRec.getLength(), myRec.getHeight(),
            myRec.getPerimeter(), myRec.getArea());
    }
}
```

2.【配列処理クラスの実行プログラム】

```
public class ArrayProcessorTest{
    public static void main(String[] args){
        int[] testArray = {66, 73, 40, 83, 95, 18, 49, 52, 26, 58};
        ArrayProcessor  ap = new ArrayProcessor(testArray);
        System.out.printf("要素数    :%d¥n", ap.getLength());
        System.out.printf("要素の最大:%d¥n", ap.getMax());
        System.out.printf("要素の最小:%d¥n", ap.getMin());
        System.out.printf("要素の合計:%d¥n", ap.getSum());
        System.out.printf("要素の平均:%.2f¥n", ap.getAverage());
    }
}
```

3.【遠投ボールの滞空時間】

```
import java.util.Scanner;

public class FlyingBallTest{
    public static void main(String[] args){
        double firstSpeed;
        double angle;
        Scanner scan = new Scanner(System.in);

        System.out.print("初速を入力してください(km/時)。:");
        firstSpeed = scan.nextDouble();
        System.out.print("投上角度を入力してください(0～90°)。:");
        angle = scan.nextDouble();

        FlyingBall fb = new FlyingBall(firstSpeed, angle);

        System.out.printf("初    速:%.2fm/s¥n", fb.getFirstSpeed());
        System.out.printf("投上角度:%.2f°¥n", fb.getAngle());
        System.out.printf("飛 距 離:%.2f m ¥n", fb.getDistance());
        System.out.printf("最 高 度:%.2f m ¥n", fb.getMaxHeight());
        System.out.printf("滞空時間:%.2f 秒¥n", fb.getTime());
    }
}
```

■第 8 章 §2 復習問題 (p.110)

1．クラスとインスタンス
① BodyMassIndex[5]　② 型（データ型　でもよい）
2．アクセサメソッドとカプセル化
① private　②フィールド更新メソッド　③カプセル化
④ private　⑤フィールド照会メソッド　⑥アクセサメソッド

■第 8 章 §2 練習問題 (p.112)

1．✓ (1)　コンストラクタが設定できないのは返却値である。
✓ (3)　静的フィールドの有効範囲はクラス内である。
✓ (4)　ローカル変数はアクセス修飾できない。
✓ (8)　参照型フィールドのデフォルトの初期値は null である。
✓ (10)　メソッドが 1 つもないクラスも定義できる。

2．① 10　② 12

Sample_A クラスから myA と myB の 2 つのインスタンスを生成して、静的フィールド（クラス変数）x と非静的フィールド y（インスタンス変数）に加算と乗算を行っている。この場合、x については、1 つの変数として 2 つのインスタンスで共有されるので、それぞれのインスタンスでの計算が反映されるが、y については、myA.y と myB.y は別の変数として認識され、myA での計算処理は、myB.y には反映されない。

3．回答例
(1)　nextLine
① パッケージ・クラス　java.util.Scanner
② 引　数　なし　返却値　String
③ スキャナを現在行の先に進めて、スキップした入力を返す。このメソッドは、最後の行区切り文字を除く、現在行の残りを返す。位置は次の行の最初に設定される。
(2)　pow
①パッケージ・クラス　java.lang.Math
②引　数　double、double　返却値　double
③ 1 番目の引数を、2 番目の引数で累乗した値を返す。
(3)　append
① パッケージ・クラス　java.lang.StringBuilder
② 引　数　String　返却値　StringBuilder
③ 引数で指定された文字列をこの文字シーケンスに追加する。
(4)　parseLong
① パッケージ・クラス　java.lang.Long
② 引　数　String　返却値　long
③ 引数で指定された文字列を long 型に変換する。
(5)　toUpperCase
① パッケージ・クラス　java.lang.String
② 引　数　なし　返却値　String
③ 文字列内の英小文字を英大文字に変換する。

■第 8 章 §2 実習問題 (p.113)

1.【国別森林面積表の作成】

```
class ForestArea{
    private String country;
    private int area;
    private int forest;
    private double ratio;

    public ForestArea(String country, int area, int forest){
        this.country = country;
        this.area = area;
        this.forest = forest;
    }

    public String getCountry(){
        return country;
    }
    public int getArea(){
        return area;
    }
    public int getForest(){
        return forest;
    }
    public double getRatio(){
        return (double)forest/area;
    }
}
```

2.【ピザホット端午の節句の販売状況】

```
public class MyPizzaTest{
    public static void main(String[] args){
        MyPizza[] myPizza = {
             new MyPizza("アマトリチャーナ    ", 1750, 29),
             new MyPizza("ハワイアンデライト  ", 1890, 32),
             new MyPizza("イタリアンバジル    ", 2150, 48),
             new MyPizza("スパイシーサルサ    ", 2280, 23),
             new MyPizza("シーフードハポン    ", 2415, 43),
             new MyPizza("パーティーデラックス", 3800, 38) };
        int total = 0;

        System.out.println("¥t¥t 本日の売上一覧 (5/5)");
        System.out.println(
                "    ピザ名                        価格   販売数量   販売金額 ");
        for(int i=0; i<myPizza.length; i++){
            System.out.printf("%s           %,d     %d    %,7d¥n",
                myPizza[i].getItem(), myPizza[i].getPrice(),
                myPizza[i].getAmount(), myPizza[i].getSales());
            total += myPizza[i].getSales();
        }
       System.out.printf(
            "                                    売上合計    %,d", total);
    }
}
```

3．【電機メーカー各社の財務状況】

```
class E_Company{
    private String companyName;
    private int asset;
    private int debt;
    private int capital;
    private double capitalRate;

    public E_Company(String companyName, int asset, int debt){
        this.companyName = companyName;
        this.asset = asset;
        this.debt = debt;
        this.capital = asset - debt;
        setCapitalRate();
    }

    public String getCompanyName(){
        return companyName;
    }
    public int getAsset(){
        return asset;
    }
    public int getDebt(){
        return debt;
    }
    public int getCapital(){
        return capital;
    }
    public void setCapitalRate(){
        capitalRate = (double)capital/asset * 100;
    }
    public double getCapitalRate(){
        return capitalRate;
    }
}
public class E_CompanyTest{
    public static void main(String[] args){
        E_Company[] ec =
                    {new E_Company("Ｈ製作所     ", 9185629, 6744240),
                     new E_Company("Ｍ電機       ", 3332679, 2223654),
                     new E_Company("Ｔ芝         ", 5536671, 4401061),
                     new E_Company("サニー       ", 12924988,9969086),
                     new E_Company("ポロソニック ", 7665004, 4798784),
                     new E_Company("藤通         ", 3024097, 2070318)};
        double max = 0.0;
        String maxCompany = "";

        System.out.println("              電機各社の財務状況と自己資本比率 ");
        System.out.println(
            "社名           資産        負債        純資産  自己資本比率 ");
        for(int i=0; i<ec.length; i++){
            System.out.printf("%s%,13d%,13d%,13d%10.2f%%¥n",
                ec[i].getCompanyName(), ec[i].getAsset(), ec[i].getDebt(),
                ec[i].getCapital(), ec[i].getCapitalRate());
            if(max < ec[i].getCapitalRate()){
                max = ec[i].getCapitalRate();
                maxCompany = ec[i].getCompanyName();
            }
        }
        System.out.println();
        System.out.printf("自己資本比率が一番高いのは %s で、%.2f%% です。",
                              maxCompany.trim(), max);
    }
}
```

■第 8 章 §3 復習問題 (p.116)

1．オーバーロード
① 並び順　② 多重定義　③ `cal.get(Calendar.DATE)`
④ `public static long abs(long a)`　⑤ `return (a < 0) ? -a : a;`
⑥ シグネチャ　⑦ `double add(double x, double y)`　⑧ `return x + y`
2．Calendar クラス
① `java.util`　② `protected`　③ `getInstance()`　④ ファクトリメソッド
⑤ `DAY_OF_WEEK`　⑥ `getInstance()`　⑦ `getTimeInMillis()`
3．カプセル化の具体的方法
① `private`　② 妥当性検査

■第 8 章 §3 練習問題 (p.118)

1．　誤っているもの
(1) `this` でアクセスできるのは、非静的フィールドで、静的フィールドにはアクセスできない。
(2) コンストラクタに返却値はない。返却値の型も記述不要である。
(3) フィールドもメソッドもクラス定義の必須要素ではない。
(4) 非静的メソッドもオーバーロードできる。
(7) `getInstance()` メソッドのようにインスタンスを生成するメソッドをファクトリメソッドという。
(8) カプセル化は一般に、フィールドの `private` 修飾とフィールド更新メソッドの妥当性検査で実現する。
(10) コンパイラが自動生成する、引数を持たないコンストラクタはデフォルトコンストラクタという。

2．オーバーロードできるもの：　(1)　(3)　(6)

?! オーバーロードが可能なのは、メソッド名が同じで引数の状態（型、数、順番）が異なる場合である。アクセス修飾の違いや `static` 修飾子の有無は、オーバーロードの可否とは無関係である。(2)、(7) のようにメソッド名が異なればまったく別のメソッドで、オーバーロードという関係は生じない。(4)、(5) のようにシグネチャが同じメソッドは、そもそも同一クラス内で定義できない。

3．オーバーロードできるもの：(1)　(2)　(4)　(6)　(7)

4．
① `Calendar.getInstance()`　② `cal.get(Calendar.YEAR)`
③ `cal.get(Calendar.MONTH) + 1`　④ `cal.get(Calendar.DATE)`
⑤ `cal.get(Calendar.HOUR_OF_DAY)`　⑥ `cal.get(Calendar.MINUTE)`
⑦ `cal.get(Calendar.SECOND)`

■第 8 章 §3 実習問題 (p.120)

1．【オーバーロードによる 3 数の最大値】

```
class Max3{
    public static int maxNum(int x, int y, int z){
        int max = x;
        if(y>max)
            max = y;
        if(z>max)
            max = z;
        return max;
    }
```

（次ページへ続く）

```
    public static long maxNum(long x, long y, long z){
        long max = x;
        if(y>max)
            max = y;
        if(z>max)
            max = z;
        return max;
    }
    public static float maxNum(float x, float y, float z){
        float max = x;
        if(y>max)
            max = y;
        if(z>max)
            max = z;
        return max;
    }
    public static double maxNum(double x, double y, double z){
        double max = x;
        if(y>max)
            max = y;
        if(z>max)
            max = z;
        return max;
    }
}
```

generics の機能（汎用データ型の定義機能）を使うと、以上の 4 つのメソッドを、次のように 1 つにまとめることができ、文字列の比較も可能になる。

```
class Max3Gen{
    public static <T extends Comparable<T>> T maxNum(T x, T y, T z){
        T max = x;
        if (y.compareTo(max)>0)
            max = y;
        if (z.compareTo(max)>0)
            max = z;
        return max;
    }
}
public class Max3GenTest{
    public static void main(String[] args){
        Max3Gen m3 = new Max3Gen();

        int xi=11, yi=7, zi=9;
        System.out.printf("%d, %d, %d の最大値は、%d です。¥n",
                          xi, yi, zi, m3.maxNum(xi, yi, zi));

        long xl=2955*2489L, yl=286*22074L, zl=244*13865L;
        System.out.printf("%d, %d, %d の最大値は、%d です。¥n",
                          xl, yl, zl, m3.maxNum(xl, yl, zl));

        float xf=12.85F, yf=153.66F, zf=-128.54F;
        System.out.printf("%.2f, %.2f, %.2f の最大値は、%.2f です。¥n",
                          xf, yf, zf, m3.maxNum(xf, yf, zf));

        double xd=355/113D, yd=Math.PI, zd=864/275D;
        System.out.printf("%.8f, %.8f, %.8f の最大値は、%.8f です。¥n",
                          xd, yd, zd, m3.maxNum(xd, yd, zd));

        String xs="appricot", ys="avocado", zs="apple";
        System.out.printf("%s, %s, %s の最大値は、%s です。¥n",
                          xs, ys, zs, m3.maxNum(xs, ys, zs));
    }
}
```

2．【時間表記と妥当性検査】

```
class MyTime{
    private int hour;
    private int minute;
    private int second;

    // コンストラクタ（オーバーロード）
    public MyTime(){
        setTime(0, 0, 0);
    }
    public MyTime(int h){
        setTime(h, 0, 0);
    }
    public MyTime(int h, int m){
        setTime(h, m, 0);
    }
    public MyTime(int h, int m, int s){
        setTime(h, m, s);
    }

    public void setTime(int h, int m, int s){
        setHour(h);
        setMinute(m);
        setSecond(s);
    }
    public void setHour(int h){
        hour = ((h>=0 && h<24)? h : 99);
    }
    public void setMinute(int m){
        minute = ((m>=0 && m<60) ? m : 99);
    }
    public void setSecond(int s){
        second = ((s>=0 && s<60)? s : 99);
    }

    public int getHour(){
        return hour;
    }
    public int getMinute(){
        return minute;
    }
    public int getSecond(){
        return second;
    }
}
```

3．【文字列切り取りクラスの作成】

(1) 【getLeft() メソッド】

```
// 文字列 str を左から span 文字切り取る
public String getLeft(String str, int span){
    newStr = str.substring(0, span);
    return newStr;
}
```

【getMid() メソッド】

```
// 文字列 str の begin 字目から end 字目までを切り取る
public String getMid(String str, int begin, int end){
    //substring の始点は 0 から始まるので始点を 1 字分左にずらす
    // 終点は index − 1 字目なので上の修正と結果的に相殺関係でそのまま
    newStr = str.substring(begin-1, end);
    return newStr;
}
```

(2)

```
class StringCutter{
    private String str, newStr;

    // コンストラクタのオーバーロード
    public StringCutter(){}              // デフォルトコンストラクタ
    public StringCutter(String str){     // 文字列型引数をとるコンストラクタ
        this.str = str;
    }

    // 文字列 str を左から span 文字切り取る
    public String getLeft(String str, int span){
              newStr = str.substring(0, span);
        return newStr;
    }
    public String getLeft(int span){
        newStr = str.substring(0, span);
        return newStr;
    }

    // 文字列 str を右から span 文字切り取る
    public String getRight(String str, int span){
        int len = str.length();
        int point = len - span;
        newStr = str.substring(point, len);
        return newStr;
    }
    public String getRight(int span){
        int len = str.length();
        int point = len - span;
        newStr = str.substring(point, len);
        return newStr;
    }

    // 文字列 str の begin 字目から end 字目までを切り取る
    public String getMid(String str, int begin, int end){
        newStr = str.substring(begin-1, end);
        return newStr;
    }
    public String getMid(int begin, int end){
        newStr = str.substring(begin-1, end);
        return newStr;
    }
}
```

(1) の StringCutter クラスでは黙示的に起動するデフォルトコンストラクタを利用している。(2) では文字列引数をとるコンストラクタを新たに追加(オーバーロード)したため、デフォルトコンストラクタは自動起動しなくなる。コンストラクタを多重定義した場合、デフォルトコンストラクタも明示的に記述する必要があることに注意する。

■第 9 章 §1 復習問題 (p.124)

1．継　承

①継承　②スーパークラス　③サブクラス　④ extends　⑤サブクラス
⑥スーパークラス　⑦スーパークラス　⑧サブクラス　⑨多重継承　⑩サブクラス
⑪コンストラクタ　⑫ protected　⑬コンストラクタ　⑭コンストラクタ

2．継承の問題点

①　カプセル化

3．this と super

①　super　②　this　③非静的　④コンストラクタ　⑤　20　⑥　21　⑦　16
⑧　17　⑨　07　⑩　08　⑪　04　⑫　09　⑬　18　⑭　22

■第 9 章 §1 練習問題 (p.126)

1．✓ (1)　サブクラスからスーパークラスのコンストラクタを呼び出すキーワードは super である。
✓ (3)　java.lang.Math クラスのように、静的メソッドで構成されているクラスのコンストラクタ は外部からインスタンスを生成する必要がないので private 修飾されている。
また、java.util.Calendar クラスは、インスタンス生成のためのファクトリメソッドを持っているので、コンストラクタは protected 修飾されている。このように、public 以外のアクセス修飾子をコンストラクタに付けることもできる。
✓ (6)　サブクラスは、2 つのスーパークラスを持つことはできない。

2．④

実行順序はつぎのとおり（< > 内は出力結果）。
16 17 07 08 03 04 11 12<7> 05<4> 09<6>

3.　(1)

```
class Baby01 extends Me{
    void myMethod(long x){
        System.out.print("Baby01 ");
        System.out.printf("%.2f¥n", Math.sqrt(x));
    }
}
```

```
class Baby02 extends Me{
    void myMethod(double x){
        System.out.print("Baby02 ");
        System.out.printf("%.2f¥n", Math.sqrt(x));
    }
}
```

(2)　④　⑥

① サブクラスのインスタンスはスーパークラスのオブジェクト変数に代入することができ、スーパークラスのメソッドは、サブクラスのメソッドでオーバーライドできる。しかし、サブクラスのメソッドはオーバーロードされない。
② スーパークラスのインスタンスをサブクラスのオブジェト変数に代入することはできない。
③ Baby02() のインスタンスを継承関係にない Baby01() のオブジェクト変数に代入することはできない。
④ サブクラスはそのインスタンスの生成段階で、継承関係の上位にあるクラスの private 修飾されていないすべてのメソッドの一覧表を作成し（オーバーロード）、実行段階で、その一覧表を参照してシグネチャの一致するメソッドを選択して実行する。
⑤ ①と同じ。
⑥ ④と同じ。

4．(1)　ア①　イ①　ウ④　エ②

(2)

```
class A{
    protected String x = "I am senior.";
    protected String y = "I am a stranger.";
}
class B extends A{
    public String x = "I am junior.";
    public void c(){
        System.out.printf("%s¥n", super.y);
    }
}
```

（次ページへ続く）

```
public class B_Test{
    public static void main(String[] args){
        B b = new B();
        b.c();
    }
}
```

または、

```
class A{
    protected String x = "I am senior.";
    protected String y = "I am a stranger.";
}
class B extends A{
    public String x = "I am junior.";
    public String z = super.y;
}
public class B_Test{
    public static void main(String[] args){
        B b = new B();
        System.out.printf("%s¥n", b.z);
    }
}
```

キーワード super は、同一インスタンスのなかでスーパークラスのメンバーを指し示す場合にしか使えない。したがって、B_Test クラスから A クラスのメンバーを super を使って直接呼び出すことはできない。呼び出すには、子クラスのなかで適切なアクセス修飾子を持つアクセサメソッドを用意する必要がある。

■第 9 章 §1 実習問題 (p.130)

1.【長方形クラスの継承】

(1)

```
class Rectangle{
    int width, height;
    public Rectangle(int width, int height){
        this.width = width;
        this.height = height;
    }
    public int getWidth(){
        return width;
    }
    public int getHeight(){
        return height;
    }
    public int getCircum(){
        return (width + height) * 2;
    }
    public int getArea(){
        return width * height;
    }
}
```

(2)

```
import java.util.Scanner;

public class RectangleTest{
    public static void main(String[] args){
        Scanner scan = new Scanner(System.in);
        System.out.print("横の長さを入力してください：");
        int myWidth = scan.nextInt();
        System.out.print("縦の長さを入力してください：");
        int myHeight = scan.nextInt();

        Rectangle rect = new Rectangle(myWidth,  myHeight);
        System.out.println();
        System.out.printf("横%d 縦%d の長方形の外周は %d、面積は %d です。¥n",
            myWidth, myHeight, rect.getCircum(), rect.getArea());
    }
}
```

(3)

```
class Square extends Rectangle{
    public Square(int length){
        super(length, length);
    }
}
```

```
import java.util.Scanner;

public class SquareTest{
    public static void main(String[] args){
        Scanner scan = new Scanner(System.in);
        System.out.print("辺の長さを入力してください：");
        int myLength = scan.nextInt();

        Square sqrt = new Square(myLength);

        System.out.println();
        System.out.printf("1 辺が %d の正方形の外周は %d、面積は %d です。¥n",
                myLength, sqrt.getCircum(), sqrt.getArea());
    }
}
```

2.【配列処理クラスの継承】

(1)

```
class ArrayDeviation extends ArrayProcessor{
    private int[] devArray;
    private double[] dev;           // 偏差の配列
    private double varia;           // 分散
    private double stdDev;          // 標準偏差
    private double[] devVal;        // 偏差値の配列

    public ArrayDeviation(int[] devArray){
        super(devArray);
        this.devArray = devArray;
        calcDev();
    }

    public void calcDev(){
        int len = getLength();                      // 配列の要素数
        dev = new double[len];                      // 偏差配列の生成
        devVal = new double[len];                   // 偏差値配列の生成

        for(int i=0; i<len; i++){
            dev[i] = devArray[i] - getAverage(); // 偏差の計算
            varia += Math.pow(dev[i], 2);           // 分散の計算
        }
        stdDev = Math.sqrt(varia)/len;              // 標準偏差の計算
        for(int i=0; i<len; i++){
            devVal[i] = 50 + dev[i] / stdDev;     // 偏差値の計算
        }
    }
    public double[] getDev(){
         return dev;
    }
    public double getVaria(){
        return varia;
    }
    public double getStdDev(){
        return stdDev;
    }
    public double[] getDevVal(){
         return devVal;
    }
}
```

```
public class ArrayDeviationTest{
    public static void main(String[] args){
        int[] testArray = {66, 73, 40, 83, 95, 18, 49, 52, 26, 58};
        ArrayDeviation  ad = new ArrayDeviation(testArray);
        double[] testDev = ad.getDevVal();

        System.out.printf("要素数    : %d¥n", ad.getLength());
        System.out.printf("要素の最大 : %d¥n", ad.getMax());
        System.out.printf("要素の最小 : %d¥n", ad.getMin());
        System.out.printf("要素の合計 : %d¥n", ad.getSum());
        System.out.printf("要素の平均 : %.2f¥n", ad.getAverage());
        System.out.printf("標準偏差   : %.2f¥n¥n", ad.getStdDev());
        System.out.println("要素    偏差値");
        for(int i=0; i<testArray.length; i++)
            System.out.printf("%4d%10.2f¥n", testArray[i], testDev[i]);
    }
}
```

(2)

```
class RainFall{
    String city;
    int[] rainFall;
    ArrayDeviation ad;

    public RainFall(String city, int[] rainFall){
        this.city = city;
        this.rainFall = rainFall;
        ad = new ArrayDeviation(rainFall);
    }
    public String getCity(){
        return city;
    }
    public int[] getRainFall(){
        return rainFall;
    }
    public int getMax(){
        return ad.getMax();
    }
    public int getMin(){
        return ad.getMin();
    }
    public double getAverage(){
        return ad.getAverage();
    }
    public double getStdDev(){
        return ad.getStdDev();
    }
}
public class RainFallTest{
    public static void main(String[] args){
        String[] city = {"東京          ", "ジャカルタ    ",
                         "ロンドン      ", "ウィーン      ",
                         "ニューヨーク  ", "シドニー      ",
                         "イスタンブール", "サンパウロ    "};
        int[][] rain = {{45, 61, 99, 125, 138, 185, 126, 148, 180, 164, 89, 46},
                        {385, 310, 99, 258, 133, 83, 31, 34, 29, 33, 175, 84},
                        {77, 51, 60, 54, 55, 57, 45, 55, 68, 73, 76, 80},
                        {38, 42, 41, 50, 61, 74, 62, 59, 45, 41, 51, 44},
                        {77, 73, 92, 96, 97, 91, 104, 95, 86, 76, 97, 86},
                        {103, 117, 134, 127, 120, 132, 98, 81, 69, 78, 83, 85},
                        {99, 61, 60, 49, 31, 22, 19, 26, 41, 72, 88, 122},
                        {232, 231, 168, 71, 76, 55, 43, 47, 67, 126, 143, 198}
                       };
        double maxDev = 0;
        double minDev = 100;
        String maxCity = "";
        String minCity = "";
        String header1 =
            "都市名          Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec";
        String header2 = " 最大 最小 平均 標準偏差 ";
```

（次ページへ続く）

```
            System.out.printf("%s%s¥n", header1, header2);
            for(int i=0; i<city.length; i++){
                RainFall rf = new RainFall(city[i], rain[i]);
                if(rf.getStdDev() > maxDev){
                    maxCity = city[i];
                    maxDev = rf.getStdDev();
                }
                if(rf.getStdDev() < minDev){
                    minCity = city[i];
                    minDev = rf.getStdDev();
                }
                System.out.printf("%s:", city[i]);
                for(int j=0; j<rain[i].length; j++)
                    System.out.printf("%4d", rain[i][j]);
                System.out.printf("%4d%4d%8.2f%8.2f¥n",
                                  rf.getMax(), rf.getMin(),
                                  rf.getAverage(), rf.getStdDev());
            }
            System.out.println();
            System.out.printf(
                    "月ごとの降雨量の差が最大の都市は%sで、最小の都市は%sです。¥n",
                    maxCity.trim(), minCity.trim());
        }
    }
```

■第 9 章 §2 復習問題 (p.134)

1．抽象メソッドと抽象クラス
① 抽象メソッド　② 具象メソッド　③ abstract　④ 抽象クラス　⑤ 未実装
2．オーバーライド
① オーバーライド　② オーバーロード　③広
3．ポリモーフィズム
① オブジェクト変数　② スーパークラス　③ サブクラス　④ オーバーライド
⑤ ポリモーフィズム　⑥ カプセル化

■第 9 章 §2 練習問題 (p.137)

1．① オーバーロード　② オーバーライド　③ オーバーライド　④ オーバーライド
⑤ オーバーロード　⑥ オーバーロード　⑦ オーバーライド

2．②

`MySenior ss = new MyJunior()` のオブジェクト変数 ss は MySenior 型だが、MyJunior 型のインスタンスを参照している。Java はポリモーフィズムによって、その時点でオブジェクト変数 ss が参照しているクラス内のメンバーを呼び出す。

3．✓①引数リストが同一なのでオーバーライドだが、返却値の型が異なるのでコンパイルエラーになる。
②引数リストが異なるのでオーバーロード。アクセス修飾、返却値は任意。
③オーバーライド。アクセス修飾がスーパークラスと同じなのでコンパイル可。
✓④オーバーライド。アクセス修飾がスーパークラスより狭い（弱い）のでコンパイルエラーになる。
⑤オーバーライド。アクセス修飾がスーパークラスより広いのでコンパイル可。
⑥オーバーロード。コンパイル可。

■第 9 章 §2 実習問題 (p.139)

1.【会員分類別特別販売価格】

```
abstract class Zenobia{
    public void showPrice(String grade, int discount){
        System.out.printf(
            "この商品の %s 会員の特別販売価格は %, d 円です。¥n",
             grade, discount);
    }
    abstract void calcPrice(int price);
}
```

2.【抽象クラスによるクラスの統合】

```
public class RollingImageTest{
    public static void main(String[] args){
        char[][] charImg= {
                          {' ',' ',' ',' ',' ',' ',' ',' '},
                          {' ',' ',' ',' ',' ','*',' ',' '},
                          {' ',' ',' ',' ',' ','*','*',' '},
                          {' ',' ',' ',' ',' ','*','*','*'},
                          {'*','*','*','*','*','*','*','*'},
                          {' ',' ',' ',' ',' ','*','*',' '},
                          {' ',' ',' ',' ',' ','*',' ',' '},
                          {' ',' ',' ',' ',' ',' ',' ',' '}
                         };

        System.out.println("原図");
        for(int i=0; i<charImg.length; i++){
            for(int j=0; j<charImg[i].length; j++){
                System.out.print(charImg[i][j]);
            }
            System.out.println();
        }
        System.out.println();

        RollingImage[] img = {
                         new RightQuater(charImg),
                         new Upset(charImg),
                         new LeftQuater(charImg)
                         };
        for(int i=0; i<img.length; i++){
            img[i].rolling();
            img[i].showImg();
        }
    }
}
```

3.【衣料品小売業の在庫管理システム】

a　イ　　b　カ　　c　ウ　　d　エ

a、b

メソッドの返却値がともに boolean なので、仮引数と何か（この場合はフィールド値）を比較して、それが等しいかどうかを真偽値で返却する式が必要である。2 つの基本データ型の数値が等しいかどうかは等値演算子（==）で判断するが、文字列のような参照型の内容の比較には String クラスの equals() メソッド（object クラスの equals() メソッドをオーバーライド）を用いる。

c

サブクラスのコンストラクタでは、スーパークラスのコンストラクタを呼び出す必要がある。スーパークラスのコンストラクタの呼び出しは、サブクラスのコンストラクタの冒頭に書く。

d

Jeans クラスのフィールド buttonFront は boolean 型なので、実行結果のように「zipper」か「button」かを表示するには、その真偽によって「zipper」か「button」かを判定・表示するコードが必要になる。

■第 9 章 §3 復習問題 (p.144)

1．インターフェースの意味と特徴
① 具象メソッド ② 抽象メソッド ③ オーバーライド ④ フィールド定数
⑤ public ⑥ 静的定数 ⑦ abstract ⑧ 多重継承 ⑨ extends
⑩ オーバーライド ⑪ public ⑫ implements
2．インターフェースの機能
① 抽象クラス ② インターフェース ③ (抽象) メソッド ④ 定数（静的）
⑤ 具象メソッド ⑥ 抽象メソッド ⑦ × ⑧ 単一継承
3．toString() メソッド
① Object ② ハッシュコード ③ soloid[i] ④ format()
4．static import
① static ② System.

■第 9 章 §3 練習問題 (p.147)

1．✓ (1) インターフェースのアクセス修飾は public か「なし」（パッケージデフォールト）で、パッケージ外からアクセスを認めるためには public を省略できない。
(2) インターフェースは「何か」を「実装」することはできない。
✓ (3) インターフースのメソッドは常に public で abstract になる。
(4) インターフェースのフィールドは常に public で static で final である。
(5) インターフェースのメソッドは常に非静的メソッドである。
(6) インターフェースが宣言できる変数は静的定数だけである。
✓ (7) インターフェースが継承できるのはインターフェースだけである。
(8) 付けるのは冗長である。

2． 誤っているもの (2) (4) (5)
(2) インターフェースのメソッドは、実装クラスのメソッドによって必ずオーバーロードされる。したがって、final 修飾はできない。
(4) インターフェースのメソッドのアクセス修飾は常に public で、protected 修飾はできない。
(5) インターフェースのメソッドは常に非静的（non-static）である。
要するに、インターフェースのメソッドにつけることができる修飾子は abstract と public だけである。引数と返却値については、一般のメソッドと同様で、特に制約はない。

3． (3) (5)
(1) インターフェース（SubFace）はインターフェース（SuperFace）を実装（implements）することはできない。
(2) クラスはインターフェースを継承できない。
(4) SubFace が実装されないので、「変数 b が見つからない」というコンパイルエラーになる。

■第 9 章 §3 実習問題 (p.149)

1．【球・円錐・円柱の表面積】
【SolidFace.java（インターフェース）】

```
interface SolidFace{
    double PAI = 3.14159;    // フィールド定数
    double getSurface();     // 表面積を求める抽象メソッド
    String toString();       // 文字列変換用抽象メソッド
}
```

【SphereFace.java】

```
import static java.lang.Math.*;

class SphereFace implements SolidFace{
    private double myRadius;
    public SphereFace(double myRadius){
        this.myRadius = myRadius;
    }
    public double getSurface(){
        return PAI * pow(myRadius, 2) * 4;
    }
    public String toString(){
        return String.format("半径と高さが%.2fの球の表面積は%6.2fです。¥n",
             myRadius, getSurface());
    }
}
```

【ConeFace.java】

```
import static java.lang.Math.*;

class ConeFace implements SolidFace{
    private double myRadius;
    public ConeFace(double myRadius){
        this.myRadius = myRadius;
    }
    public double getSurface(){
        return PAI * pow(myRadius, 2) * (sqrt(5) + 1);
    }
    public String toString(){
        return String.format("半径と高さが%.2fの円錐の表面積は%6.2fです。¥n",
            myRadius, getSurface());
    }
}
```

【CylinderFace.java】

```
import static java.lang.Math.*;

class CylinderFace implements SolidFace{
    private double myRadius;
    public CylinderFace(double myRadius){
        this.myRadius = myRadius;
    }
    public double getSurface(){
        return PAI * pow(myRadius, 2) * 6;
    }
    public String toString(){
        return String.format("半径と高さが%.2fの円柱の表面積は%6.2fです。¥n",
            myRadius, getSurface());
    }
}
```

【SolidFaceTest.java】

```
import java.util.Scanner;

public class SolidFaceTest{
    public static void main(String[] args){
        double radius;
        Scanner scan = new Scanner(System.in);

        System.out.print("半径を入力してください。:");
        radius = scan.nextDouble();
        System.out.println();
        SolidFace[] solidFace = {
                          new SphereFace(radius),
                          new ConeFace(radius),
                          new CylinderFace(radius)
                         };
        for(int i=0; i<solidFace.length; i++)
            System.out.println(solidFace[i]);
    }
}
```

2.【消費税計算】

```
interface PriceCounter{
    double TAX = 0.08;
    int getPrice(int input);
}
class NetPrice implements PriceCounter{
    public int getPrice(int input){
        return  (int)(input*(1+TAX) + 0.5);
    }
}
class TaxPrice implements PriceCounter{
    public int getPrice(int input){
        return (int)(input/(1+TAX) + 0.5);
    }
}
```

NetPrice()、TaxPrice() の返却値はそれぞれ、
(int)(Math.round(input*(1+TAX)))、 (int)(Math.round(input/(1+TAX))) でもよい。

3.【円に内接する多角形の外周と面積】

(1)

```
// インターフェース  Polygon01
interface Polygon01{
    double getCircum();
    double getArea();
    String toString();
}
```

```
// 三角形 Trigon01
class Trigon01 implements Polygon01{
    private int myRadius;
    private String myShape = "三角形";
    public Trigon01(int myRadius){
        this.myRadius = myRadius;
    }
    public String getShape(){
        return myShape;
    }
    public double getCircum(){
        return myRadius * Math.sqrt(3) * 3;
    }
    public double getArea(){
        return 3*Math.sqrt(3)/4*Math.pow(myRadius, 2);
    }
    public String toString(){
        return String.format(
            "半径が%dの円に内接する%sの外周は%.2f, 面積は%.2fです。",
                myRadius, myShape, getCircum(), getArea());
    }
}
```

```
//四角形 Tetragon01
class Tetragon01 implements Polygon01{
    private int myRadius;
    private String myShape = "四角形";
    public Tetragon01(int myRadius){
        this.myRadius = myRadius;
    }
    public double getCircum(){
        return myRadius * Math.sqrt(2) * 4;
    }
    public double getArea(){
        return 2*Math.pow(myRadius, 2);
    }
    public String toString(){
        return String.format(
            "半径が%dの円に内接する%sの外周は%.2f, 面積は%.2fです。",
                myRadius, myShape, getCircum(), getArea());
    }
}
```

```
//  五角形  Hexagon01
class Hexagon01 implements Polygon01{
    private int myRadius;
    private String myShape = "五角形";
    public Hexagon01(int myRadius){
        this.myRadius = myRadius;
    }
    public double getCircum(){
        return   Math.sqrt(10-2*Math.sqrt(5))/2*myRadius * 5;
    }
    public double getArea(){
        return 5 * Math.sqrt(10+2*Math.sqrt(5)) / 8 *
                              Math.pow(myRadius, 2);
    }
    public String toString(){
        return String.format(
            "半径が%dの円に内接する%sの外周は%.2f, 面積は%.2fです。",
                myRadius, myShape, getCircum(), getArea());
    }
}
```

```
// 六角形  Pentagon01
class Pentagon01 implements Polygon01{
    private int myRadius;
    private String myShape = "六角形";
    public Pentagon01(int myRadius){
        this.myRadius = myRadius;
    }
    public double getCircum(){
        return myRadius * 6;
    }
    public double getArea(){
        return 3*Math.sqrt(3)*Math.pow(myRadius, 2)/2;
    }
    public String toString(){
        return String.format(
            "半径が%dの円に内接する%sの外周は%.2f, 面積は%.2fです。",
                myRadius, myShape, getCircum(), getArea());
    }
}
```

```
// テストプログラム  Polygon01Test
import java.util.Scanner;

public class Polygon01Test{
    public static void main(String[] args){
        Scanner scan = new Scanner(System.in);
        System.out.print("図形の外接円の半径を整数で入力してください。:");
        int radius = scan.nextInt();
        Polygon01[] pg = {new Trigon01(radius),
                          new Tetragon01(radius),
                          new Hexagon01(radius),
                          new Pentagon01(radius)
                          };
        for(int i=0; i<pg.length; i++)
            System.out.println(pg[i]);

    }
}
```

(2)

```
//    抽象クラス  Polygon02
abstract class Polygon02{
    abstract String getShape();
    abstract double getCircum();
    abstract double getArea();

    public String toString(int radius, String shape,
                                    double circum, double area){
        return String.format(
            "半径が %d の円に内接する %s の外周は %.2f, 面積は %.2f です。",
            radius, shape, circum, area);
    }
}
```

```
//   三角形  Trigon02
class Trigon02 extends Polygon02{
    private int myRadius;
    private String myShape = "三角形";
    public Trigon02(int myRadius){
        this.myRadius = myRadius;
    }
    public String getShape(){
        return myShape;
    }
    public double getCircum(){
        return myRadius * Math.sqrt(3) * 3;
    }
    public double getArea(){
        return 3*Math.sqrt(3)/4*Math.pow(myRadius, 2);
    }
}
```

```
//    四角形   Tetragon02
class Tetragon02 extends Polygon02{
    private int myRadius;
    private String myShape = "四角形";
    public Tetragon02(int myRadius){
        this.myRadius = myRadius;
    }
    public String getShape(){
        return myShape;
    }
    public double getCircum(){
        return myRadius * Math.sqrt(2) * 4;
    }
    public double getArea(){
        return 2*Math.pow(myRadius, 2);
    }
}
```

```
// 五角形 Hexagon02
class Hexagon02 extends Polygon02{
    private int myRadius;
    private String myShape = "五角形";
    public Hexagon02(int myRadius){
        this.myRadius = myRadius;
    }
    public String getShape(){
        return myShape;
    }
    public double getCircum(){
        return   Math.sqrt(10-2*Math.sqrt(5))/2*myRadius * 5;
    }
    public double getArea(){
        return 5 * Math.sqrt(10+2*Math.sqrt(5)) / 8 *
                                            Math.pow(myRadius, 2);
    }
}
```

```
//    六角形    Pentagon02
class Pentagon02 extends Polygon02{
    private int myRadius;
    private String myShape = "六角形";
    public Pentagon02(int myRadius){
        this.myRadius = myRadius;
    }
    public String getShape(){
        return myShape;
    }
    public double getCircum(){
        return myRadius * 6;
    }
    public double getArea(){
        return 3*Math.sqrt(3)*Math.pow(myRadius, 2)/2;
    }
}
```

```
//    テストプログラム    Polygon02Test
import java.util.Scanner;

public class Polygon02Test{
    public static void main(String[] args){
        Scanner scan = new Scanner(System.in);
        System.out.print("図形の外接円の半径を整数で入力してください。:");
        int radius = scan.nextInt();
        Polygon02[] pg = {new Trigon02(radius),
                          new Tetragon02(radius),
                          new Hexagon02(radius),
                          new Pentagon02(radius)
                          };
        for(int i=0; i<pg.length; i++)
            System.out.println(
                pg[i].toString(radius, pg[i].getShape(),
                pg[i].getCircum(), pg[i].getArea()));

    }
}
```

■第 10 章 §1 復習問題 (p.153)

1．コレクション
① データ構造　② ArrayList　③ コレクション　④ import
2．ArrayList のインスタンスの生成
① 要素のデータ型　② <Integer>　③ 参照型　④ ラッパークラス
3．ArrayList のメソッド
① add(e)　② add(i, e)　③ set(i, e)　④ get(i)　⑤ contains(e)
⑥ indexOf(e)　⑦ indexLastOf(e)　⑧ clear()　⑨ remove(i)
⑩ removeRange(i, j)　⑪ size()
4．オートボクシング
① new Integer(i)　② 基本データ型　③ 参照型　④ オートボクシング
⑤ 参照型　⑥ 基本データ型　⑦ オートアンボクシング　⑧ オートボクシング
⑨ オートアンボクシング
5．拡張 for 文
① ループカウンタ　② 拡張 for 文　③ String str : month

■第 10 章 §1 練習問題 (p.155)

1．正しいもの　(3)　(6)　(8)

(1)　char は基本データ型なので、ArrayList の要素の型として指定できない。
(2)　要素は < >（アングルブラケット）で指定する。末尾の ( ) もない。
(4)　ArryList<String> myString = new ArrayList<String>(); が正しい。
(5)　配列のような初期値設定はできない。
(6)　要素数の初期値を仮設定することはできる。
(7)　double は基本データ型なので指定できない。

2．(1)　④　　(2)　③　　(3)　①　　(4)　②　　(5)　④

3．(1)　autoboxing　23 行　　myList.add(i)
int 型の i を Integer 型に autoboxing している。
auto-unboxing　28 行　　for(int e : myList)
参照型の myList の要素を int 型へ auto-unboxing している。
(2)　④　　（myList の要素数が 100 以上で、かつ次の要素が素数のとき）
(3)　②

■第 10 章 §1 実習問題 (p.158)

1．【各種データの ArrayList への記憶】
(1)

```
import java.util.ArrayList;

class ListStringTest{
    public static void main(String[] args){
        ArrayList<String> myFruits = new ArrayList<String>();

        myFruits.add("orange");
        myFruits.add("peach");
        myFruits.add("grape");
        myFruits.add("banana");
        myFruits.add("apple");
        myFruits.add("strawberry");

        for(String f : myFruits)
            System.out.println(f);
    }
}
```

(2)

```
import java.util.ArrayList;
import static java.lang.Math.*;

class ListDoubleTest{
    public static void main(String[] args){
        ArrayList<Double> myRoot = new ArrayList<Double>();

        myRoot.add(sqrt(10));
        myRoot.add(sqrt(20));
        myRoot.add(sqrt(30));
        myRoot.add(sqrt(40));
        myRoot.add(sqrt(50));

        for(Double root : myRoot)
            System.out.printf("%.7f¥n", root);
    }
}
```

(3)

```
import java.util.ArrayList;

class ListCharacterTest{
    public static void main(String[] args){
        ArrayList<Character> myFish = new ArrayList<Character>();

        myFish.add('鰹');   //カツオ
        myFish.add('鰆');   //サワラ
        myFish.add('鱈');   //タラ
        myFish.add('鮃');   //ヒラメ
        myFish.add('鰤');   //ブリ
        myFish.add('鰊');   //ニシン
        myFish.add('鰯');   //イワシ
        myFish.add('鯵');   //アジ
        myFish.add('鱸');   //スズキ
        myFish.add('鱧');   //ハモ

        for(Character fish : myFish)
            System.out.println(fish);
    }
}
```

2.【Java のキーワードの確認テスト】

```
import java.util.Scanner;
import java.util.ArrayList;

class KeywordTest{
    public static void main(String[] args){
        Scanner scan = new Scanner(System.in);
        ArrayList<String> arr = new ArrayList<String>();
        System.out.println(" 知っている Java のキーワードを入力してください。");
        String keyword = scan.next();
        while(!keyword.equals("end")){
            arr.add(keyword);
            keyword = scan.next();
        }
        System.out.println();
        System.out.printf(" 入力されたキーワード数は %d です。¥n", arr.size());
        for(String javaKey : arr)
            System.out.printf("%s  ", javaKey);
    }
}
```

3.【反転してもまだ素数】

```
import java.util.ArrayList;

class PairPrimeTest{
    public static void main(String[] args){
        int upper = 1000;

        PrimeList pl = new PrimeList();
        ArrayList<Integer> primes = pl.listUp(upper);
        Reverser rv = new Reverser();
        // 素数ペアの表示
        for(int dt : primes){
            int reverseDt = rv.reverse(dt);
            if(dt>=reverseDt) continue;      //2 数の組み合わせの重複を排除
            if(primes.contains(reverseDt))  // 反転数が素数リストに含まれているか
                System.out.printf("%3d と %3d はともに素数 ¥n",
                                                dt, reverseDt);
        }
    }
}
```

4．【BMI クラスの実行プログラム】

```
import java.util.ArrayList;

public class BMI_Tester{
    public static void main(String[] args){
        ArrayList<BMI> bmiList = new ArrayList<BMI>();

        bmiList.add(new BMI("西本　元", 165.5, 60.9));
        bmiList.add(new BMI("村尾惇平", 170.2, 51.7));
        bmiList.add(new BMI("芦沢典宏", 166.3, 60.1));
        bmiList.add(new BMI("水本壮史", 160.9, 50.5));
        bmiList.add(new BMI("深沢裕重", 174.8, 87.1));
        bmiList.add(new BMI("木場　諭", 176.4, 64.3));
        bmiList.add(new BMI("日向達耶", 164.7, 62.3));
        bmiList.add(new BMI("渡邉弦太", 174.6, 98.2));
        bmiList.add(new BMI("松葉紀希", 173.2, 59.5));
        bmiList.add(new BMI("尾関征宏", 165.1, 53.5));
        bmiList.add(new BMI("本間利之", 181.0, 90.9));
        bmiList.add(new BMI("脇田芳朗", 167.2, 56.1));

        String header = "氏名      身長   体重  BMI  健康体重  判定";
        System.out.println(header);
        for(BMI bdt : bmiList)
            System.out.println(bdt);
    }
}
```

■第 10 章 §2 復習問題 (p.162)

1．データ構造
① 順次構造　② 連鎖構造　③ 木構造　④ 順次構造　⑤ 連鎖構造　⑥ 木構造

2．コレクションの特徴
① データ構造　② メソッド　③ 動的データ構造　④ 参照型　⑤ 基本データ型
⑥ 固定 (生成時に決定)　⑦ 可変（実行時に伸長）　⑧ 基本データ型・参照型
⑨ 参照型のみ　⑩ 演算子　⑪メソッド　⑫ 順次構造　⑬ 順次構造・連鎖構造など

3．コレクションフレームワーク
① コレクションフレームワーク　② List　③ Set　④ Map　⑤ List
⑥ index　⑦ Set　⑧ LinkedList　⑨ 連鎖　⑩ HashSet　⑪ 木
⑫ 連鎖　⑬ ハッシュ　⑭ TreeMap　⑮ LinkedHashMap

4．ユーティリティクラス
① Collections　② Arrays　③ swap　④ rotate　⑤ sort
⑥ reverse　⑦ shuffle　⑧ fill

■第 10 章 §2 練習問題 (p.164)

1．(1) ③　(2) ①　(3) ③　(4) ③　(5) ②　(6) ③　(7) ②　(8) ①
(9) ③　(10) ③　(11) ①　(12) ②　(13) ②　(14) ②

2．(1) TreeSet　(2) LinkedHashSet　(3) HashSet

3．(1) ① TreeSet　② mySet　③ v
(2) ① Arrays.asList(f1)　② Arrays.asList(f2)　③ addAll　④ Double

Arrays クラスの aslist() メソッドによって配列を List 化したコピーを作り、そのコピーを TreeSet のコンストラクタにより TreeSet に記憶している。 このように TreeSet を使えば、配列の重複する要素を整理し、それらを一定の基準で整列させることができる。

■第 10 章 §2 実習問題 (p.166)

1.【野球チームの登録メンバー】

```
import java.util.*;

class Baystars{
    public static void main(String[] args){
       String[] major = {"石川", "梶谷", "グリエル", "ブランコ",
                                    "後藤", "筒香", "山崎", "黒羽根"};
       String[] minor = {"桑原", "下園", "内村", "嶺井",
                                                "松本", "白崎", "荒波"};
       String[] pitcher = {"久保", "モスコーソ", "山口", "加賀",
                                                "藤井", "井納", "三上"};

        LinkedList<String> majorList =
            new LinkedList<String>(Arrays.asList(major));
        LinkedList<String> minorList =
            new LinkedList<String>(Arrays.asList(minor));
        LinkedList<String> pitcherList =
            new LinkedList<String>(Arrays.asList(pitcher));

        System.out.println("(1)majorList を表示する。");
        for(String player : majorList)
            System.out.printf("%s ", player);
        System.out.println("¥n");

        System.out.println(
        "(2)major の 3 番目に minor の 4 番打者を挿入し、以下、打順を 1 つずつ下げる。");
        majorList.remove(majorList.size() - 1);
        majorList.add(2, minorList.get(3));

        for(String player : majorList)
            System.out.printf("%s ", player);
        System.out.println("¥n");

        System.out.println(
        "(3)major の末尾に pitcher リストの先頭の要素を加える。");
        majorList.add(pitcherList.getFirst());
        for(String player : majorList)
            System.out.printf("%s ", player);
        System.out.println("¥n");

        System.out.println(
        "(4)major の「後藤」を削除し、8 番打者として minor の末尾の選手を加える。");
        majorList.remove(majorList.indexOf("後藤"));
        majorList.add(7, minorList.getLast());
        for(String player : majorList)
            System.out.printf("%s ", player);
    }
}
```

2.【Java のキーワードの確認テスト 2】

```
import java.util.*;

class KeywordDrill{
    public static void main(String[] args){
        Scanner scan = new Scanner(System.in);
        String[] keyArray = {
            "abstract", "assert", "boolean", "break", "byte",
            "case", "catch", "char", "class", "const",
            "continue", "default", "do", "double", "else",
            "enum", "extends", "final", "finally", "float",
            "for", "goto", "if", "implements", "import",
            "instanceof", "int", "interface", "long", "native",
            "new", "package", "private", "protected", "public",
            "return", "short", "static", "strictfp", "super",
            "switch", "synchronized", "this", "throw", "throws",
            "transient", "try", "void", "volatile", "while"
            };                                          （次ページに続く）
```

```
        HashSet<String> wordSet =
                new HashSet<String>(Arrays.asList(keyArray));
        ArrayList<String> inputList = new ArrayList<String>();
        System.out.println("知っている Java のキーワードを入力してください。");
        String keyword = scan.next();
        while(wordSet.contains(keyword)){
            inputList.add(keyword);
            keyword = scan.next();
        }
        System.out.println();
        System.out.printf("%s は Java のキーワードではありません。¥n", keyword);
        System.out.printf(
            " 入力された Java のキーワード数は %d です。¥n", inputList.size());
        for(String javaKey : inputList)
            System.out.printf("%s  ", javaKey);
    }
}
```

3．【TreeSet クラスによる直近の素数】

```
import java.util.*;

public class PrimeTreeSetTest{
    public static void main(String[] args){
        int inputNum = 0;
        int inputMax = 9999999;
        Scanner scan = new Scanner(System.in);

        do{
            System.out.printf(
                    " 整数を入力してください (2 ～ %d)。: ", inputMax);
            inputNum = scan.nextInt();
        }while(inputNum<2 || inputNum>inputMax);
        System.out.println();

        // 素数リストの生成と受け取り
        PrimeTreeSet pts = new PrimeTreeSet();
        TreeSet<Integer> primes = pts.listUp();
        int myFloor = primes.floor(inputNum);
        int myCeiling = primes.ceiling(inputNum);

        System.out.printf("%d 以下の最大の素数は %d です。¥n",
                inputNum, myFloor);
        System.out.printf("%d 以上の最小の素数は %d です。¥n",
                inputNum, myCeiling);
    }
}

class PrimeTreeSet{
    public TreeSet<Integer> listUp(){
        int max = 10000000;
        boolean[] isPrime = new boolean[max];
        TreeSet<Integer> primeSet = new TreeSet<Integer>();

        //isPrime[0]、isPrime[1] は生成段階の初期設定で false
        for(int i=2; i<max; i++)
            isPrime[i] = true;
        // エラトステネスのふるい
        for(int i=2; i*i<max; i++){
            if(isPrime[i])
                for(int j=i; i*j<max; j++)
                    isPrime[i*j] = false;
        }
        //TreeSet にパック
        for(int i=2; i<max; i++){
            if(isPrime[i])
                primeSet.add(i);
        }
        return primeSet;
    }
}
```

■第 11 章 §1 復習問題 (p.168)

1．File クラス

① java.io　② String フルパス名　③ String ファイル名　④ String 子パス名
⑤ exists()　⑥ isFile()　⑦ getName()　⑧ long　⑨ list()

2．再帰呼び出し

① 再帰呼び出し　② (n-1) の総和　③ 1　④ num + sumUp(num-1)
⑤ 基底条件　⑥ num += i

■第 11 章 §1 練習問題 (p.170)

1. (1)

```
n = 10
n =  9
n =  8
n =  7
n =  6
n =  5
n =  4
n =  3
n =  2
n =  1
n =  0
```

```
---- return ----
返却値 =  1 +  0
返却値 =  2 +  1
返却値 =  3 +  3
返却値 =  4 +  6
返却値 =  5 + 10
返却値 =  6 + 15
返却値 =  7 + 21
返却値 =  8 + 28
返却値 =  9 + 36
返却値 = 10 + 45
----  end -----
screw01(10) = 55
```

(2) ① 【Screw02Test クラス】

```
class Screw02Test{
    static int screw02(int m, int n){
        System.out.printf("m = %2d  n = %2d¥n", m, n);
        if(m==n){
            System.out.println("---- return ----");
            return m;
        }
        else{
            int returnValue = n + screw02(m+1, n);
            System.out.printf("返却値 = %2d + %2d¥n", n, returnValue - n);
            return returnValue;
        }
    }
    public static void main(String[] args){
        int value = screw02(5, 9);
        System.out.println("----  end -----");
        System.out.printf("screw02(5, 9) = %2d¥n", value);
    }
}
```

【実行経過確認：省略】

② 【Screw03Test クラス】

```
class Screw03Test{
    static int screw03(int m, int n){
        System.out.printf("m = %2d  n = %2d¥n", m, n);
        if(n==0){
            System.out.println("---- return ----");
            return 1;
        }
        else{
            int returnValue = m * screw03(m, n-1);
            System.out.printf("返却値 = %3d * %8d¥n", m, returnValue/m);
            return returnValue;
        }
    }
    public static void main(String[] args){
        int value = screw03(5, 9);
        System.out.println("----  end -----");
        System.out.printf("screw03(5, 9) = %2d¥n", value);
    }
}
```

【実行経過確認：省略】

2．① File ② File ③ exists() ④ getName() ⑤ isFile()
⑥ getAbsolutePath() ⑦ getInstance() ⑧ lastModified()
⑨ isDirectory() ⑩ list()

3． (1) ② (2) ① (3) ① (4) ③

■第 11 章 §1 実習問題 (p.174)

1．【再帰による段数とピン数の関係】

(1)

```
import java.util.Scanner;

class PinTriangle01{
    public static void main(String[] args){
        Scanner scan = new Scanner(System.in);
        int pin = 0;
        System.out.print("段数を入力してください。：");
        int input = scan.nextInt();

        for(int i=1; i<=input; i++)
            pin += i;

        System.out.printf("ピンの数は %d 本です。", pin);
    }
}
```

```
import java.util.Scanner;

class PinTriangle02{
    public static void main(String[] args){
        Scanner scan = new Scanner(System.in);
        System.out.print("段数を入力してください。：");
        int input = scan.nextInt();
        System.out.printf("ピンの数は %d 本です。", sumUp(input));
    }
    static int sumUp(int pin){
            if (pin == 1)
                return 1;
            else
                return pin + sumUp(pin-1);
    }
}
```

2.【 perrin 数列の再帰処理】

(1)

```
class PerrinTest01{
    static int perrin(int n){
        switch(n){
            case 0:
                return 3;
            case 1:
                return 0;
            case 2:
                return 2;
            default:
                return perrin(n-2) + perrin(n-3);
        }
    }
    public static void main(String[] args){
        for(int i=0; i<=65; i++){
            System.out.printf("%2d 項：%, 10d¥n", i, perrin(i));
        }
    }
}
```

(2)

```
class PerrinTest02{
    static int perrin(int n){
        switch(n){
            case 0:
                return 3;
            case 1:
                return 0;
            case 2:
                return 2;
            default:
                return perrin(n-2) + perrin(n-3);
        }
    }
    public static void main(String[] args){
        for(int i=1; i<=65; i++){
            if(perrin(i)%i==0)
                System.out.printf("%d ", i);
        }
    }
}
```

(1) の数列をペラン数列という。(2) はペラン数列 p(n) 上で、p(n)/n == 0 を満たす n を列挙している。この n の並びはこの範囲で素数の数列と一致する。仮に n が大きくなってもこの数列が素数を生成し続けるとすると、何の規則性もないように見える素数の並び方の背後に、一定の法則性が隠されていることになる。この数列の値 p(n) は急速に増大し、n=150 のときの p(n) の値はほとんど long 値の上限に達するが、この範囲でも n の並びは素数列と一致している。n>150 も調べたいが、Java の基本データ型の範囲では、それ以上の計算はオーバーフローする。

この行く末に興味のある方は、次ページに記載した Java の大域整数演算用の BigInteger クラスを使って計算の続きを行い、その結果と「エラトステネスのふるい」法で出力した正規の素数列と比較してみるとよい。

なお、次ページのプログラムで計算する最大の素数 n（=999983）に対応する p(n) は 122,124「けた」の数値で、これをすべて印字すると A4 用紙びっしりで 30 ページ以上の枚数が必要になる。

◆ 1000000 以下の perrin 疑似素数列◆

```
import java.math.BigInteger;

public class PerrinBig{
    public static void main(String[] args){
        int max = 1000000;
        int counter = 1;     //"2" を素数としてあらかじめカウント
        BigInteger p0 = BigInteger.valueOf(3);    //p0 = 3;
        BigInteger p1 = BigInteger.valueOf(0);    //p1 = 0;
        BigInteger p2 = BigInteger.valueOf(2);    //p2 = 2;

        System.out.printf("%7d", 2);
        for(int i=3; i<=max; i++){
            BigInteger p3 = p1.add(p0);    //p3 = p1 + p0;

            if (p3.remainder(BigInteger.valueOf(i)).equals
                (BigInteger.valueOf(0))){       //p3 % i == 0
                counter++;
                System.out.printf(
                        counter%10==0 ? "%7d¥n" : "%7d", i);
            }
            p0 = p1;
            p1 = p2;
            p2 = p3;
        }
        System.out.printf(
              "¥n¥n1000000 以下のペラン疑似素数は %d 個あります。", counter);
    }
}
```

3.【ファイルの検索】

```
import java.util.Scanner;
import java.io.File;

class FileFinder{
    static String myFileName;
    static Boolean isFind = false;
    public static void main(String[] args){
        Scanner scan = new Scanner(System.in);

        System.out.print("検索対象のファイル名を入力してください。:");
        myFileName = scan.nextLine();

        System.out.print("検索範囲のフォルダ名を入力してください。:");
        String myDirName = scan.nextLine();
        File myDir = new File(myDirName);

        pathFinder(myDir);
        if(!isFind)
            System.out.println("検索対象ファイルが見あたりません。");
    }
    static void pathFinder(File dir){
        File[] pathList = dir.listFiles();
        for(File f : pathList){
            if(f.isDirectory())
                pathFinder(f);
            else{
                String fName = f.getName();
                if(myFileName.equals(fName)){
                    System.out.println(f.getAbsolutePath());
                    isFind = true;
                }
            }
        }
    }
}
```

4．【ディレクトリの階層表示】

```
import java.io.*;
import java.util.*;

public class IndentedFileList {
    static int maxLevel = 15;           // ファイル階層の深さの最大値
    static String myIndent = "    ";    // 階層ごとにスペース３個
    static String[] myLevel = new String[maxLevel]; // 階層の深さの配列

    public static void main(String[] args){
        IndentedFileList myDir = new IndentedFileList();
        myDir.setDirectory();
    }
    public void setDirectory(){
        myLevel[0] = "";
        for (int i=1; i<maxLevel; i++)
           myLevel[i] = myLevel[i-1] + myIndent;

        System.out.print(
                ″ファイルリストを作成するディレクトリを指定してください。″);
        Scanner input = new Scanner(System.in);
        File root = new File(input.nextLine());

        if (root!=null && root.isDirectory())
            listUp(root, 0);
        else
            System.out.printf("%s はディレクトリではありません。", root);
    }
    // 再帰メソッド
    public void listUp(File fdir, int depth) {
        System.out.println(myLevel[depth] + fdir.getName());
        // それがディレクトリで階層の深さが maxLevel 未満ならば...
        if (fdir.isDirectory() && depth<maxLevel) {
            for (File f : fdir.listFiles()) {  // 階層化の要素を仮引数として
                listUp(f, depth+1);            //listUp() の再帰呼び出し
            }
        }
    }
}
```

■第 11 章 §2 復習問題 (p.176)

1．Scanner クラスにテキストデータの読み込み

① java.lang　② 標準入力　③ File　④ scan.nextLine()　⑤ File
⑥ hasNextLine

2．Split() メソッド

① String　② split()　③ date.split("/")

3．例外処理 1

① 例外　② try 句　③ catch 句　④ try　⑤ catch　⑥ 検査例外
⑦ 非検査例外　⑧ Throwable　⑨ Exception　⑩ IOException
⑪ RuntimeException　⑫ IOException　⑬ try　⑭ finally

4．Map インターフェース

① キー　② HashMap　③ TreeMap　④ LinkedHashMap　⑤ get()
⑥ put()　⑦ keySet()　⑧ clear()　⑨ get　⑩ keySet()　⑪ put
⑫ remove　⑬ size()　⑭ Map<String, String>　⑮ HashMap　⑯ TreeMap
⑰ LinkedHashMap

■第 11 章 §2 練習問題 (p.181)

1．① TreeMap　② Character　③ null　④ put　⑤ keySet()

2．① Collections　② Scanner　③ File　④ FileNotFoundException
⑤ IOException　⑥ NoSuchElementException　⑦ hasNextLine()
⑧ scan.nextLine()

■第 11 章 §2 実習問題 (p.183)

1．【文字列中の単語の取り出しと整列】

```
import java.util.Arrays;

class FruitTest{
    public static void main(String[] args){
        String[] fruit;
        String myFruit =
             "pear lemon avocado citron melon persimmon apricot cherry fig";
        fruit = myFruit.split(" ");
        Arrays.sort(fruit);
        for(String f : fruit)
            System.out.println(f);
    }
}
```

2．【単語ファイルの読み込みと整列１】

```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;
import java.util.ArrayList;
import java.util.Collections;

class JavaKeyword{
    Scanner input;
    ArrayList<String> keywords = new ArrayList<String>();

    public void openFile(String myFile){
        try{
            input = new Scanner(new File(myFile));
        }
        catch(FileNotFoundException e){
            System.out.println(" ファイルが見あたりません。");
            System.exit(1);
        }
    }
    public void readFile(){
        while(input.hasNextLine()){
            String keyString = input.nextLine();
            if(keyString.startsWith("■"))
                continue;
            keywords.add(keyString);
        }
        input.close();
        Collections.sort(keywords);
    }
    public void display(){
        for(String k : keywords)
            System.out.println(k);
    }
}
```

（次ページに続く）

```
public class JavaKeywordTest{
    public static void main(String[] args){
        JavaKeyword jkw = new JavaKeyword();
        String myFile = "JavaKeyword.txt";
        jkw.openFile(myFile);
        jkw.readFile();
        jkw.display();
    }
}
```

3.【モールス符号】

(1)

```
import java.util.Scanner;
import java.io.FileNotFoundException;
import java.util.NoSuchElementException;
import java.util.HashMap;
import java.io.File;

class MorseCode01{
    String fileName;
    Scanner scan;
    HashMap<Character, String> morseCode =
                    new HashMap<Character, String>();

    public MorseCode01(String fileName){
        this.fileName = fileName;
    }
    public void openFile(){
        try{
            scan = new Scanner(new File(fileName));
        }
        catch(FileNotFoundException e){
            System.out.println(" ファイルが見あたりません。");
            System.exit(1);
        }
    }
    public void readFile(){
        String codeSet;
        String[] myCode;
        try{
            while(scan.hasNextLine()){
                codeSet = scan.nextLine();
                myCode = codeSet.split("¥¥t");
                morseCode.put(myCode[0].charAt(0), myCode[1]);
            }
        }
        catch(NoSuchElementException e){
            System.out.println(" 入力ファイルのデータ形式が不正です。");
            System.exit(1);
        }
        finally{
            scan.close();
        }
    }
    public void showFile(){
        for(Character ch : morseCode.keySet())
            System.out.printf("%s¥t%s¥n", ch, morseCode.get(ch));
    }
}

public class MorseCode01Test{
    public static void main(String[] args){
        String morse = "morseTab.txt";
        MorseCode01 mc = new MorseCode01(morse);
        mc.openFile();
        mc.readFile();
        mc.showFile();
    }
}
```

(2)

```
import java.util.Scanner;
import java.io.FileNotFoundException;
import java.util.NoSuchElementException;
import java.util.HashMap;
import java.io.File;

class MorseCode02{
    String fileName;
    Scanner scan;
    HashMap<Character, String> morseCode =
                        new HashMap<Character, String>();

    public MorseCode02(String fileName){
        this.fileName = fileName;
    }
    public void openFile(){
        try{
            scan = new Scanner(new File(fileName));
        }
        catch(FileNotFoundException e){
            System.out.println(" ファイルが見あたりません。");
            System.exit(1);
        }
    }
    public void readFile(){
        String codeSet;
        String[] myCode;
        try{
            while(scan.hasNextLine()){
                codeSet = scan.nextLine();
                myCode = codeSet.split("¥¥t");
                morseCode.put(myCode[0].charAt(0), myCode[1]);
            }
        }
        catch(NoSuchElementException e){
            System.out.println(" 入力ファイルのデータ形式が不正です。");
            System.exit(1);
        }
        finally{
            scan.close();
        }
    }
    public void enCode(String myString){
        myString = myString.toUpperCase();
        char myChar;
        StringBuilder sb = new StringBuilder();

        for(int i=0; i<myString.length(); i++){
            myChar = myString.charAt(i);
            if(myChar == ' ')
                sb.append("   "); // 空白 2 個追加、合計 3 個に
            else{
                sb.append(morseCode.get(myChar));
                sb.append(' ');
            }
        }
        String myCode = sb.toString();
        System.out.println(myCode);
    }
}

public class MorseCode02Test{
    public static void main(String[] args){
        String morse = "morseTab.txt";
        String jobs = "stay hungry, stay foolish.";
        MorseCode02 mc = new MorseCode02(morse);
        mc.openFile();
        mc.readFile();
        mc.enCode(jobs);
    }
}
```

(3)

```
import java.io.*;
import java.util.*;

class MorseCode03{
    String codeName;
    String myTextName;
    Scanner codeFile;
    Scanner myTextFile;
    HashMap<Character, String> morseCode =
          new HashMap<Character, String>();

    public MorseCode03(String codeName, String myTextName){
        this.codeName = codeName;
        this.myTextName = myTextName;
    }
    public void openFile(){
        try{        // モールスコード表を開く
            codeFile = new Scanner(new File(codeName));
        }
        catch(FileNotFoundException e){
            System.out.println("コード表が見あたりません。");
            System.exit(1);
        }
        try{        // 変換対象ファイルを開く
            myTextFile = new Scanner(new File(myTextName));
        }
        catch(FileNotFoundException e){
            System.out.println("対象ファイルが見あたりません。");
            System.exit(1);
        }
    }
    public void readCode(){
        String codeSet;
        String[] myCode;
        try{        // コード表を HashMap に読み込み、
            while(codeFile.hasNextLine()){
                codeSet = codeFile.nextLine();
                myCode = codeSet.split("¥¥t");
                morseCode.put(myCode[0].charAt(0), myCode[1]);
            }
        }
        catch(NoSuchElementException e){
            System.out.println("コード表のデータ形式が不正です。");
            System.exit(1);
        }
        finally{
            codeFile.close();
        }
    }
    public void enCode(){
        String myTextLine;
        char myChar;
        StringBuilder sb = new StringBuilder();
        System.out.println("【原 文】");
        try{
            while(myTextFile.hasNextLine()){
                myTextLine = myTextFile.nextLine().toUpperCase();
                System.out.println(myTextLine);
                for(int i=0; i<myTextLine.length(); i++){
                    myChar = myTextLine.charAt(i);
                    if(myChar == ' ')
                        sb.append("  ");    // 空白 2 個追加、合計 3 個に
                    // 対応する符号のない文字は無視
                    else if(morseCode.get(myChar) == null)
                        sb.append("");
                    else{
                        sb.append(morseCode.get(myChar));
                        sb.append(" ");
                    }
                }
                sb.append("¥n");
            }
        }
```

（次ページに続く）

```
        catch(NoSuchElementException e){
            System.out.println("対象ファイルのデータ形式が不正です。");
            System.exit(1);
        }
        finally{
            myTextFile.close();
        }
        System.out.println("【翻訳文】");
        System.out.print(sb.toString());
    }
}
public class MorseCode03Test{
    public static void main(String[] args){
        MorseCode03 mc =
                  new MorseCode03("morseTab.txt", "DannyBoy.txt");
        mc.openFile();
        mc.readCode();
        mc.enCode();
    }
}
```

(4)

```
import java.io.*;
import java.util.*;

class MorseCode04{
    String codeName;
    String myMorseName;
    Scanner codeFile;
    Scanner myMorseFile;
    HashMap<String, Character> morseCode =
                        new HashMap<String, Character>();

    public MorseCode04(String codeName, String myMorseName){
        this.codeName = codeName;
        this.myMorseName = myMorseName;
    }
    public void openFile(){
        try{
            codeFile = new Scanner(new File(codeName));
        }
        catch(FileNotFoundException e){
            System.out.println("コード表が見あたりません。");
            System.exit(1);
        }
        try{          // 変換対象ファイルを開く
            myMorseFile = new Scanner(new File(myMorseName));
        }
        catch(FileNotFoundException e){
            System.out.println("対象ファイルが見あたりません。");
            System.exit(1);
        }
    }
    public void readCode(){
        String codeSet;
        String[] myCode;
        try{          // コード表を HashMap に読み込み、
            while(codeFile.hasNextLine()){
                codeSet = codeFile.nextLine();
                myCode = codeSet.split("¥¥t");
                morseCode.put(myCode[1], myCode[0].charAt(0));
            }
        }
        catch(NoSuchElementException e){
            System.out.println("コード表のデータ形式が不正です。");
            System.exit(1);
        }
        finally{
            codeFile.close();
        }
    }
```

（次ページに続く）

```
    public void deCode(){
        String myMorseLine;
        String[] myCode;
        StringBuilder sb = new StringBuilder();

        System.out.println("【コード文】");
        try{
            while(myMorseFile.hasNextLine()){
                myMorseLine = myMorseFile.nextLine();
                System.out.println(myMorseLine);
                String[] word = myMorseLine.split("    ");
                for(int j=0; j<word.length; j++){
                    String[] ch = word[j].split(" ");
                    for(int k=0; k<ch.length; k++){
                        // 変換不能記号は"?"で処理
                        if(morseCode.get(ch[k]) == null)
                            sb.append("?");
                        else
                            sb.append(morseCode.get(ch[k]));
                    }
                    sb.append(" ");
                }
                sb.append("¥n");
            }
        }
        catch(NoSuchElementException e){
            System.out.println("入力ファイルのデータ形式が不正です。");
            System.exit(1);
        }
        finally{
            myMorseFile.close();
        }
        System.out.println("【復号文】");
        System.out.print(sb.toString());
    }
}

public class MorseCode04Test{
    public static void main(String[] args){
        MorseCode04 mc =
                new MorseCode04("morseTab.txt", "DannyBoyMrs.txt");
        mc.openFile();
        mc.readCode();
        mc.deCode();
    }
}
```

4.【深海生物の DNA 分析】

```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;
import java.util.TreeMap;

class DeepCreature{
    Scanner scan;
    String sequence;

    public void openFile(){
        try{
            scan = new Scanner(new File("DNA.txt"));
        }
        catch(FileNotFoundException e){
            System.out.println("ファイルが見あたりません。");
            System.exit(1);
        }
    }
```

（次ページに続く）

```
    public void readFile(){
        while(scan.hasNextLine())
            sequence = scan.nextLine();
        scan.close();
    }
    public void codonCounter(){
        final int span = 3;             //3 塩基で 1 組
        int counter = 0;                // 出現頻度計測カウンタ
        int eachCodonTotal = 0;         // 総コドン数のカウンタ
        int grandTotal = 0;             // 総コドン数
        String codon = null;

        // キーを文字列型、値を Integer 型の TreeMap を生成
        TreeMap<String, Integer> treeMap
                        = new TreeMap<String, Integer>();
        System.out.printf(" 塩基配列のサイズは %,d です。¥n",
                                              sequence.length());

        // 塩基配列データを３文字ずつスキップしてアクセス
        for(int i=0; i<sequence.length(); i+=span){
            // 塩基配列を３文字ずつのコドンに区切る
            codon = sequence.substring(i, i+span);
            //treeMap のキーとして未登録なら登録し、その値として 1 を設定
            if(treeMap.get(codon) == null)
                treeMap.put(codon, 1);
            else{    // すでに登録されていれば、その値に 1 を加算
                counter = treeMap.get(codon);
                treeMap.put(codon, ++counter);
            }
        }
        // コドン数を keySet() メソッドを使って取り出し、総コドン数に集計して表示
        for(String cdn : treeMap.keySet()){
            eachCodonTotal = treeMap.get(cdn);  // キーごとの値の取り出し
            grandTotal += eachCodonTotal;
            System.out.printf("%s : %3d¥n",cdn, eachCodonTotal);
         }
          System.out.printf("¥n 総コドン数は %,d です。", grandTotal);

    }
}

public class DeepCreatureTest{
    public static void main(String[] args){
        DeepCreature dc = new DeepCreature();
        dc.openFile();
        dc.readFile();
        dc.codonCounter();
    }
}
```

■第 11 章 §3 復習問題 (p.186)

1．Formatter クラスの利用

① util　② format()　③ Formatter　④ format　⑤ System
⑥ close()

2．オブジェクトの整列

① Comparator　② compare()　③ Collections　④ toString()
⑤ SortByHeight()　⑥ implements　⑦ compare　⑧ compareTo
⑨ Integer　⑩ A>B　⑪ A<B　⑫ return 1　⑬ return -1

3．例外処理 2

① throws

■第 11 章 §3 練習問題 (p.189)

1 ① Throwable ② Exception ③ IOException ④ FileNotFoundException
⑤ split() ⑥ 検査例外 ⑦ catch ⑧ カンマ

2. (1) ① throws ② Formatter
(2) ③ Comparator ④ compare ⑤ return ⑥ compareTo
(3) ⑦ util ⑧ split ⑨ asList ⑩ cards

3. ① IOException ② TreeSet ③ Formatter ④ format ⑤ close()
⑥ try

■第 11 章 §3 実習問題 (p.191)

1.【単語ファイルの読み込みと整列】

```
import java.io.File;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Formatter;
import java.util.NoSuchElementException;
import java.util.Scanner;

class JavaKeyFile{
    Scanner input;
    Formatter output;
    ArrayList<String> keywords = new ArrayList<String>();

    public void openFile(String inFile, String outFile)
                                    throws FileNotFoundException{
        input = new Scanner(new File(inFile));
        output = new Formatter(new File(outFile));
    }
    public void readFile() throws NoSuchElementException{
        while(input.hasNextLine()){
            String keyString = input.nextLine();

            if(keyString.startsWith("■"))
                continue;
            keywords.add(keyString);
        }
        input.close();
        Collections.sort(keywords);
    }
    public void writeFile(String outFile) throws IOException{
        for(String ks : keywords)
            output.format("%s¥n", ks);
        output.close();
    }
}

public class JavaKeyFileTest{
    public static void main(String[] args){
        JavaKeyFile jkw = new JavaKeyFile();
        String inFile = "JavaKeyword.txt";
        String outFile = "SortedKeyword.txt";
        try{
            jkw.openFile(inFile, outFile);
        }catch(FileNotFoundException e){
            System.out.println(" ファイルが見あたりません。");
            System.exit(1);
        }
```

(次ページに続く)

```
        try{
            jkw.readFile();
        }catch(NoSuchElementException e){
            System.out.println(" 入力ファイルのデータ形式が不正です。");
            System.exit(1);
        }
        try{
            jkw.writeFile(outFile);
        }catch(IOException e){
            System.out.println(" ファイルに書き込みができません。");
            System.exit(1);
        }
    }
}
```

2.【政令指定都市のファイル処理】

(1)

```
import java.io.File;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.ArrayList;
import java.util.NoSuchElementException;
import java.util.Scanner;

class BigCityFileIO_01{
    Scanner input;
    ArrayList<BigCity> city = new ArrayList<BigCity>();

    public void openFile(String inFileName)
                                throws FileNotFoundException{
        input = new Scanner(new File(inFileName));
    }
    public void readFile() throws NoSuchElementException{
        String cityLine;
        String[] cityItem;
        while(input.hasNextLine()){
            cityLine = input.nextLine();
            cityItem = cityLine.split("¥¥t");
            city.add(new BigCity(cityItem[0], cityItem[1], cityItem[2],
               Integer.parseInt(cityItem[3]), Integer.parseInt(cityItem[4]),
               Integer.parseInt(cityItem[5])));
        }
    }
    public void showFile(){
        String header = " コード  都市    都道府県  政令指定年  区の数   人口 ";
        System.out.println(header);
        for(BigCity bc : city)
            System.out.printf("%s  %s  %s  %d     %2d  %,10d¥n",
                        bc.getCode(), bc.getCity(), bc.getPref(),
                        bc.getYear(), bc.getWard(), bc.getPop());
    }
}

public class BigCityFileIO_01Test{
    public static void main(String[] args){
        String inFile = "BigCity.txt";
        BigCityFileIO_01 bc01 = new BigCityFileIO_01();
        try{
            bc01.openFile(inFile);
        }catch(FileNotFoundException e){
            System.out.printf("%s が見つかりません。", inFile);
            System.exit(1);
        }
        try{
            bc01.readFile();
        }catch(NoSuchElementException e){
            System.out.println(" 入力ファイルのデータ形式が不正です。");
            System.exit(1);
        }
        bc01.showFile();
    }
}
```

(2)

```
import java.io.File;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.NoSuchElementException;
import java.util.Scanner;

class BigCityFileIO_02{
    Scanner input;
    ArrayList<BigCity> city = new ArrayList<BigCity>();

    public void openFile(String inFileName)
                                throws FileNotFoundException{
        input = new Scanner(new File(inFileName));
    }
    public void readFile() throws NoSuchElementException{
        String cityLine;
        String[] cityItem;
        while(input.hasNextLine()){
            cityLine = input.nextLine();
            cityItem = cityLine.split("¥¥t");
            city.add(new BigCity(cityItem[0], cityItem[1], cityItem[2],
               Integer.parseInt(cityItem[3]), Integer.parseInt(cityItem[4]),
               Integer.parseInt(cityItem[5])));
        }
        Collections.sort(city, new SortByPop());
    }
    public void showFile(){
        String header = "コード 都市    都道府県 政令指定年 区の数   人口";
        System.out.println(header);
        for(BigCity bc : city)
            System.out.printf("%s %s %s %d     %2d %,10d¥n",
                        bc.getCode(), bc.getCity(), bc.getPref(),
                        bc.getYear(), bc.getWard(), bc.getPop());
    }
}

public class BigCityFileIO_02Test{
    public static void main(String[] args){
        String inFile = "BigCity.txt";
        BigCityFileIO_02 bc01 = new BigCityFileIO_02();
        try{
            bc01.openFile(inFile);
        }catch(FileNotFoundException e){
            System.out.printf("%s が見つかりません。", inFile);
            System.exit(1);
        }
        try{
            bc01.readFile();
        }catch(NoSuchElementException e){
            System.out.println("入力ファイルのデータ形式が不正です。");
            System.exit(1);
        }
        bc01.showFile();
    }
}

class SortByPop implements Comparator<BigCity>{
    public int compare(BigCity bc1, BigCity bc2){
        return bc2.getPop().compareTo(bc1.getPop());
    }
}
```

(3)

```
import java.io.File;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.Formatter;
import java.util.IllegalFormatException;
import java.util.NoSuchElementException;
import java.util.Scanner;

class BigCityFileIO_03{
    Scanner input;
    Formatter output;
    ArrayList<BigCity> city = new ArrayList<BigCity>();

    public void openFile(String inFileName, String outFileName)
                throws FileNotFoundException, IOException{
        input = new Scanner(new File(inFileName));
        output = new Formatter(new File(outFileName));
    }
    public void readFile() throws NoSuchElementException{
        String cityLine;
        String[] cityItem;
        while(input.hasNextLine()){
            cityLine = input.nextLine();
            cityItem = cityLine.split("¥¥t");
            city.add(new BigCity(cityItem[0], cityItem[1], cityItem[2],
               Integer.parseInt(cityItem[3]), Integer.parseInt(cityItem[4]),
               Integer.parseInt(cityItem[5])));
        }
        Collections.sort(city, new SortByPop());
    }
    public void writeFile(){
        String header = "コード 都市   都道府県 政令指定年 区の数   人口";
        output.format("%s¥n", header);
        for(BigCity bc : city)
            output.format("%s  %s  %s  %d      %2d  %,10d¥n",
                          bc.getCode(), bc.getCity(), bc.getPref(),
                          bc.getYear(), bc.getWard(), bc.getPop());
        output.close();
    }
}

public class BigCityFileIO_03Test{
    public static void main(String[] args){
        String inFile = "BigCity.txt";
        String outFile ="sortedBigCity.txt";
        BigCityFileIO_03 bc01 = new BigCityFileIO_03();
        try{
            bc01.openFile(inFile, outFile);
        }catch(FileNotFoundException e){
            System.out.printf("%s が見つかりません。", inFile);
            System.exit(1);
        }
        catch(IOException e){
            System.out.println("ファイルが開けません。");
            System.exit(1);
        }
        try{
            bc01.readFile();
        }catch(NoSuchElementException e){
            System.out.println("入力ファイルのデータ形式が不正です。");
            System.exit(1);
        }
        try{
            bc01.writeFile();
        }catch(IllegalFormatException e){
            System.out.println("出力ファイルのデータ形式が不正です。");
            System.exit(1);
        }
    }
}
```

（次ページに続く）

```
class SortByPop implements Comparator<BigCity>{
    public int compare(BigCity bc1, BigCity bc2){
        return bc2.getPop().compareTo(bc1.getPop());
    }
}
```

3．【各国の領土面積と森林面積】

(1)

```
import java.io.File;
import java.io.IOException;
import java.io.FileNotFoundException;
import java.util.TreeSet;
import java.util.Comparator;
import java.util.Formatter;
import java.util.Scanner;
import java.util.IllegalFormatException;
import java.util.NoSuchElementException;

class GreenState{
    String code;
    String nation;
    String nationE;
    int area;
    int forest;
    double ratio;

    public GreenState(String code, String nation,
                      String nationE, int area, int forest){
        this.code = code;
        this.nation = nation;
        this.nationE = nationE;
        this.area = area;
        this.forest = forest;
    }
    public String getCode(){
        return code;
    }
    public String getNation(){
        return nation;
    }
    public String getNationE(){
        return nationE;
    }
    public Integer getArea(){
        return area;
    }
    public Integer getForest(){
        return forest;
    }
    public Double getRatio(){
        return (double)forest/area;
    }
    public String toString(){
        return String.format(
            "%-10s%-18s%,12d%,10d%8.2f%%  %-10s¥n",
             getCode(), getNationE(), getArea(),
             getForest(), getRatio()*100, getNation());
    }
}

class GreenStateFileIO_01{
    Scanner input;
    Formatter output;
    String header =
          "国コード  国名（英名）         領土面積  森林面積  森林率  国名（日本名）";
    TreeSet<GreenState> forestSet =
            new TreeSet<GreenState>(new SortByRatio());
```

（次ページに続く）

```
    public void openFile(String inFile, String outFile) throws
                FileNotFoundException, IOException{
            input = new Scanner(new File(inFile));
            output = new Formatter(new File(outFile));
    }
    public void readFile() throws NoSuchElementException, IOException{
        String recordLine;
        String[] item;
        String sep = "¥¥t";
        while(input.hasNextLine()){
            recordLine = input.nextLine();
            item = recordLine.split(sep);
            forestSet.add(new GreenState(item[0], item[1], item[2],
                                Integer.parseInt(item[3]),
                                Integer.parseInt(item[4])));
        }
    }
    public void show(){
        System.out.println(header);
        for(GreenState gs : forestSet)
            System.out.print(gs);
    }
    public void writeFile() throws IllegalFormatException, IOException{
        output.format("%s¥n", header);
        for(GreenState gs : forestSet)
            output.format("%s", gs);
        output.close();
    }
}

class SortByRatio implements Comparator<GreenState>{
    public int compare(GreenState g1, GreenState g2){
        return g2.getRatio().compareTo(g1.getRatio());
    }
}

public class GreenStateFileIO_01Test{
    public static void main(String[] args){
        String inFile = "forestArea.txt";
        String outFile = "forestOut.txt";

        GreenStateFileIO_01 gsf = new GreenStateFileIO_01();
        try{
            gsf.openFile(inFile, outFile);
        }catch(FileNotFoundException e){
            System.out.println("入力ファイルが見あたりません。");
            System.exit(1);
        }catch(IOException e){
            System.out.println("入出力例外が発生しました。");
            System.exit(1);
        }
        try{
            gsf.readFile();
        }catch(NoSuchElementException e){
            System.out.println("入力ファイルのデータ形式が不正です。");
            System.exit(1);
        }catch(IOException e){
            System.out.println("入出力例外が発生しました。");
            System.exit(1);
        }
        gsf.show();
        try{
            gsf.writeFile();
        }catch(IllegalFormatException e){
            System.out.println("出力ファイルの形式が不正です。");
            System.exit(1);
        }catch(IOException e){
            System.out.println("入出力例外が発生しました。");
            System.exit(1);
        }
    }
}
```

(2)

```
import java.io.*;
import java.util.*;

class GreenState{

       （省略　(1) と同じ）

}

class GreenStateFileIO_02{
    Scanner input;
    Formatter output;
    TreeSet<GreenState> forestSet;
    String header =
        "国コード　国名（英名）          領土面積　森林面積　森林率　国名（日本名）";

    public void openFile(String inFile, String outFile) throws
                FileNotFoundException, IOException{
            input = new Scanner(new File(inFile));
            output = new Formatter(new File(outFile));
    }
    public void readFile(int choice) throws
                            NoSuchElementException, IOException{
        String recordLine;
        String[] item;
        String sep = "¥¥t";

        switch(choice){
            case 1:
                forestSet = new TreeSet<GreenState>(new SortByCodeUp());
                break;
            case 2:
                forestSet = new TreeSet<GreenState>(new SortByCodeDw());
                break;
            case 3:
                forestSet = new TreeSet<GreenState>(new SortByRatioUp());
                break;
            case 4:
                forestSet = new TreeSet<GreenState>(new SortByRatioDw());
                break;
        }
        while(input.hasNextLine()){
            recordLine = input.nextLine();
            item = recordLine.split(sep);
            forestSet.add(new GreenState(item[0], item[1], item[2],
                            Integer.parseInt(item[3]),
                            Integer.parseInt(item[4])));
        }

    }
    public void show(){
        System.out.println(header);
        for(GreenState gs : forestSet)
            System.out.print(gs);
    }
    public void writeFile() throws
                    IllegalFormatException, IOException{
        output.format("%s¥n", header);
        for(GreenState gs : forestSet)
            output.format("%s", gs);
        output.close();
    }
}
class SortByCodeUp implements Comparator<GreenState>{
    public int compare(GreenState g1, GreenState g2){
        return g1.getCode().compareTo(g2.getCode());
    }
}
class SortByCodeDw implements Comparator<GreenState>{
    public int compare(GreenState g1, GreenState g2){
        return g2.getCode().compareTo(g1.getCode());
    }
}
```

（次ページに続く）

```
class SortByRatioUp implements Comparator<GreenState>{
    public int compare(GreenState g1, GreenState g2){
        return g1.getRatio().compareTo(g2.getRatio());
    }
}
class SortByRatioDw implements Comparator<GreenState>{
    public int compare(GreenState g1, GreenState g2){
        return g2.getRatio().compareTo(g1.getRatio());
    }
}

public class GreenStateFileIO_02Test{
    public static void main(String[] args){
        String inFile = "forestArea.txt";
        String outFile = "forestOut.txt";
        Scanner scan = new Scanner(System.in);
        int choice;
        GreenStateFileIO_02 gsf = new GreenStateFileIO_02();

        System.out.println("■Menu■");
        System.out.println(
        "1. 国コード（昇順）¥n2. 国コード（降順）¥n3. 森林比率（昇順）¥n4. 森林比率（降順）");
        do{
            System.out.print("整列方法を１～４の番号で指定してください。:");
            choice = scan.nextInt();
            System.out.println();
        }while(choice<1 || choice>4);

        try{
            gsf.openFile(inFile, outFile);
        }catch(FileNotFoundException e){
            System.out.println("入力ファイルが見あたりません。");
            System.exit(1);
        }catch(IOException e){
            System.out.println("入出力例外が発生しました。");
            System.exit(1);
        }
        try{
            gsf.readFile(choice);
        }catch(NoSuchElementException e){
            System.out.println("入力ファイルのデータ形式が不正です。");
            System.exit(1);
        }catch(IOException e){
            System.out.println("入出力例外が発生しました。");
            System.exit(1);
        }
        gsf.show();
        try{
            gsf.writeFile();
        }catch(IllegalFormatException e){
            System.out.println("出力ファイルの形式が不正です。");
            System.exit(1);
        }catch(IOException e){
            System.out.println("入出力例外が発生しました。");
            System.exit(1);
        }
    }
}
```

4．【点、線クラスを使った三角形の面積計算】

◆ MyTriangle クラス◆

```
class MyTriangle{
    private String triCode;
    private MyPoint[] pt = new MyPoint[3];
    private MyLine[] ln = new MyLine[3];
    private double[] length = new double[3];

    public MyTriangle(String triCode, MyPoint p1,
                                      MyPoint p2, MyPoint p3){
        this.triCode = triCode;
        pt[0] = p1;
        pt[1] = p2;
        pt[2] = p3;
        ln[0] = new MyLine(pt[0], pt[1]);
        ln[1] = new MyLine(pt[1], pt[2]);
        ln[2] = new MyLine(pt[2], pt[0]);
        setLength();
    }
    private void setLength(){
        length[0] = ln[0].getLen();
        length[1] = ln[1].getLen();
        length[2] = ln[2].getLen();
    }
    public String getCode(){
        return triCode;
    }
    public String getP0(){
        return String.format("(%2d,%2d)", pt[0].getX(), pt[0].getY());
    }
    public String getP1(){
        return String.format("(%2d,%2d)", pt[1].getX(), pt[1].getY());
    }
    public String getP2(){
        return String.format("(%2d,%2d)", pt[2].getX(), pt[2].getY());
    }
    public Double getLn0(){
        return length[0];
    }
    public Double getLn1(){
        return length[1];
    }
    public Double getLn2(){
        return length[2];
    }
    public Double getArea(){
        double a = length[0];
        double b = length[1];
        double c = length[2];

        double s = (a + b + c) / 2;
        double area =
            Math.sqrt((s - a) * (s - b) * (s -c));
        return area;
    }
}
```

◆ MyTriangleFileIO クラス◆

```
import java.io.*;
import java.util.*;

class MyTriangleFileIO{
    Scanner input;
    Formatter output;
    TreeSet<MyTriangle> triSet
        = new TreeSet<MyTriangle>(new SortByArea());

    public void openFile(String inFile, String outFile)
            throws FileNotFoundException,IOException{
        input = new Scanner(new File(inFile));
        output = new Formatter(new File(outFile));
    }
```

（次ページに続く）

```
    public void readFile() throws NoSuchElementException{
        String dataLine;
        String triCode;
        String[] pos;
        int p0x, p0y, p1x, p1y, p2x, p2y;

        while(input.hasNextLine()){
            dataLine = input.nextLine();
            pos = dataLine.split(",");
            triCode = pos[0];
            p0x = Integer.parseInt(pos[1]);
            p0y = Integer.parseInt(pos[2]);
            p1x = Integer.parseInt(pos[3]);
            p1y = Integer.parseInt(pos[4]);
            p2x = Integer.parseInt(pos[5]);
            p2y = Integer.parseInt(pos[6]);
            triSet.add(new MyTriangle(triCode, new MyPoint(p0x, p0y),
                        new MyPoint(p1x, p1y), new MyPoint(p2x, p2y)));
        }
    }
    public void writeFile() throws IllegalFormatException, IOException{
        String header
            = "型名   点A      点B      点C    線ab  線bc  線ca  面積";
        output.format("%s¥n",header);
        for(MyTriangle mt : triSet)
            output.format("%s  %s  %s  %s  %.2f  %.2f  %.2f  %.2f¥n",
                mt.getCode(), mt.getP0(), mt.getP1(), mt.getP2(),
                mt.getLn0(), mt.getLn1(), mt.getLn2(), mt.getArea());
        output.close();
    }
}

class SortByArea implements Comparator<MyTriangle>{
    public int compare(MyTriangle mt1, MyTriangle mt2){
        return mt1.getArea().compareTo(mt2.getArea());
    }
}
```

◆MyTriangleAreaTest クラス◆

```
import java.io.*;
import java.util.*;

public class MyTriangleAreaTest{
    public static void main(String[] args){
        String inFile = "TriData.txt";
        String outFile = "TriOut.txt";
        MyTriangleFileIO mtf = new MyTriangleFileIO();
        try{
            mtf.openFile(inFile, outFile);
        }catch(FileNotFoundException e){
            System.out.println("入力ファイルが見あたりません。");
            System.exit(1);
        }catch(IOException e){
            System.out.println("入力例外が発生しました。");
            System.exit(1);
        }
        try{
            mtf.readFile();
        }catch(NoSuchElementException e){
            System.out.println("入力ファイルのデータ形式が不正です。");
            System.exit(1);
        }
        try{
            mtf.writeFile();
        }catch(IllegalFormatException e){
            System.out.println("出力ファイルの形式が不正です。");
            System.exit(1);
        }catch(IOException e){
            System.out.println("出力例外が発生しました。");
            System.exit(1);
        }
    }
}
```