例題学習と問題演習

Java の手ほどき

学習編

問題解答・解説

■第 1 章 §1 実習問題 (p.15)

(1)

```
class MyFirstProgram{
    public static void main(String[] args){
        System.out.println("my first program.");
    }
}
```

(2)

```
class Case{
    public static void main(String[] args){
        System.out.println("Java は大文字と小文字を区別する。");
    }
}
```

(3)

```
class JavaBirthday{
    public static void main(String[] args){
        System.out.println("Java が生まれたのは 1995 年である。");
    }
}
```

(4)

```
class Extension{
    public static void main(String[] args){
        System.out.println("拡張子は java である。");
    }
}
```

(5)

```
class Terminator{
    public static void main(String[] args){
        System.out.println("終端子はセミコロンである。");
    }
}
```

(6)

```
class YourHello{
    public static void main(String[] args){
        System.out.println("こんにちは、私はマリンバです。");
    }
}
```

クラス定義、main() メソッドの定義および println() の各文は、今後学ぶほとんどすべてのプログラムで出てくる基本的な構文である。
キーボードの前に座ると、無意識のうちに指が動くまで、徹底的に身体で覚える必要がある。

■第 1 章 §2 練習問題 (p.18)

(1) (2) (3)　解答省略

(1) のプログラムを次のように数値計算部分のパーレンを省略してコンパイルしようとすると、次ページのようなエラーが発生する。

```
class Calc01Test{
    public static void main(String[] args){
        System.out.println("10-5-4-3 = " +10-5-4-3);
        System.out.println("10-(5-4)-3 = " +10-(5-4)-3);
        System.out.println("(10-5)-(4-3) = " +(10-5)-(4-3));
    }
}
```

```
C:¥MyJava>javac Calc01Test.java
Calc01Test.java:3: 演算子 - は java.lang.String,int に適用できません。
        System.out.println("10-5-4-3 = " +10-5-4-3);
                                            ^
Calc01Test.java:4: 演算子 - は java.lang.String,int に適用できません。
        System.out.println("10-(5-4)-3 = " +10-(5-4)-3);
                                              ^
Calc01Test.java:5: 演算子 - は java.lang.String,int に適用できません。
        System.out.println("(10-5)-(4-3) = " +(10-5)-(4-3));
                                                    ^
エラー 3 個
```

この原因は、println() 文の 1 行目でいえば、まず「"10-5-4-3 = " +10」で「10-5-4-3 = 10」という文字列を生成し、その文字列に対して -5 という数値演算を行おうとしたことによる（文字列に対して「+」以外の演算子は使用できない）。

次のプログラムの実行結果がどうなるか、考えてみよう。

```
class Calc02Test{
    public static void main(String[] args){
        System.out.println("10+5+4+3 = " +10+5+4+3);
        System.out.println("10+(5-4)+3 = " +10+(5-4)+3);
        System.out.println("(10-5)+(4-3) = " + (10-5)+(4-3));
    }
}
```

■第 1 章 §2 実習問題 (p.18)

1.

```
class Price{
    public static void main(String[] args){
        System.out.print(" 本体価格 2500 円の本の税込価格は ");
        System.out.println(2500 * 1.08 + " 円です。");
    }//end method main
}//end class Price
```

?! 小数を含む数値 1.08 を式のなかで使っているので、計算結果も小数を含む数値で表示される。整数部分のみを表示するには、第 2 章で学ぶキャスト、第 3 章で学ぶ書式付き出力などの方法がある。

2.

```
class Sphere{
    public static void main(String[] args){
        System.out.println(" 半径 5 ㎝ の球の表面積は, " +
                    4 * 3.14 * 5 * 5 + " 平方 ㎝ です。");
    }//end method main
}//end class Sphere
```

?! " で挟まれた文は途中で改行できない。2 行に分けて書く場合は、上記のように + 記号で分割表記する。

3.

```
class Box{
    public static void main(String[] args){
        System.out.println(
            " 上底 3㎝、下底 5㎝、高さ 7㎝ の台形の面積は " +
                        (3 + 5) * 7 / 2 + " 平方 ㎝ です。");
    }//end method main
}//end class Box
```

■第 2 章 §1 練習問題 (p.25)

① 10　　← 11 ÷ 3 × 7 ÷ 2 の計算結果。割り算の余りはそのつど切り捨てられる。
② false　　← a<=b の式の値は false なので、boo の値も false になる。
③実行時エラー　← d=a%b の値が 0 なので、実行時に次の「0 で除算エラー」が発生する。

```
Exception in thread "main" java.lang.ArithmeticException: / by zero
at Op03.main(Op03.java:18)
```

■第 2 章 §1 実習問題 (p.25)

1．(1)

```
class Price01{
    public static void main(String[] args){
        int net = 2500;
        double tax = 0.08;
        double price;

        price = net * (1 + tax);
        System.out.println(
            " 本体価格 2500 円の本の税込価格は、" + price + " 円です。");
    }
}
```

(2)

```
class Sphere01{
    public static void main(String[] args){
        int hankei = 5;
        double menseki;

        menseki = 4 * hankei * hankei * 3.14;
        System.out.println(" 半径 5 cm の球の表面積は, " + menseki +
                    " 平方 cm です。");
    }
}
```

(3)

```
class Box01{
    public static void main(String[] args){
        int jyotei = 3;
        int katei = 5;
        int takasa = 7;
        int menseki;

        menseki = (jyotei + katei) * takasa / 2;

        System.out.println(
          " 上底 3cm、下底 5cm、高さ 7cm の台形の面積は " +
                              menseki + " 平方 cm です。");
    }
}
```

2.

```
class Coral{
    public static void main(String[] args){
        int seatS = 286;
        int seatA = 127;
        int seatB = 139;
        int seatC = 178;
        int sumOfSeat;
        int sumOfSales;

        int salesS = 12000 * seatS;
        int salesA = 10000 * seatA;
        int salesB = 8000 * seatB;
        int salesC = 6000 * seatC;

        sumOfSeat = seatS + seatA + seatB + seatC;
        sumOfSales = salesS + salesA + salesB + salesC;

        System.out.println(
            "本日の来場予定者数　" + sumOfSeat + "人");
        System.out.println(
            "本日のチケット販売総額　" + sumOfSales + "円");
    }
}
```

■実行結果

本日の来場予定者数　735人
本日のチケット販売総額　6912000円

■第2章 §2 練習問題 (p.32)

1．(1)　コンパイルエラーになるもの　①　　③　　④

①浮動小数点数型（float型）は整数型（long型）に代入できない。
③ 16*myByte=112で、short型の範囲は -32768 ～ +32767。　byte型→short型の黙示的拡張変換によって問題なく代入できそうだが……。

Javaでは、整数型を要素とする式の計算結果は、自動的にint型に「昇格」する。したがって、16*myByteの計算結果は黙示的にint型となり、その値をshort型に代入しようとするとエラーが発生する。代入するには、次のようにキャストが必要。

```
class Byte2Short2{
    public static void main(String[] args){
        byte myByte = 7;
        short myShort = (short)(16 * myByte);
        System.out.println("myShort = " + myShort);
    }
}
```

④ char型とshort型はともに2バイトの記憶領域なので代入できそうだが、char型はshort型に代入できない（char型はbyte、short型以外の数値型変数には代入できる……②参照）。

(2)　解答略

2．

① x1 = 10
　 x2 = 14

② x1 = 1
　 x2 = 2

③ a = 9
　 b = -6
　 c = 51

④ a = 7
　 b = 49

xの値はx1の計算・代入段階で5、その直後に6、x2の計算・代入段階で7である。

cは、3 - (8 × (-6)) → 51。

aはa+=4で7になるので、bは7×7になる。
かりに、b = a * (a+=4) * a だったとしたら、bの値は？

⑤ a = 15
b = 60
c = 56
d = 14.0
e = 2

x の値が表示のつど変化していくことに注意する。

■第 2 章 §2 実習問題 (p.34)

1.

```
class LightYear{
    public static void main(String[] args){
        final int LIGHT_SPEED = 299792458;
        final long SECONDS_PER_YEAR = 365*24*60*60;      //1 年の秒数
        long lightYear = LIGHT_SPEED * SECONDS_PER_YEAR;
        System.out.println("1 光年は " + lightYear/1000 + "km です。");
    }
}
```

光年は正確には、1 年を 365.25 日として、299792458m × 24 × 60 × 60 × 365.25 日の計算で 9,460,730,472,580,800m と定められている。

2.

```
class KitchenKnife{
    public static void main(String[] args){
        int genbu = 99720;
        int byakko = 110270;
        int suzaku = 124980;
        int seiryu = 141730;
        int sum = genbu*3 + byakko*5 + suzaku*5 + seiryu*2;
        double rate = 127.882872;

        int pound = (int)(sum/rate);
        int pense = (int)((sum/rate - pound) * 100);

        System.out.println(" 請求金額 ");
        System.out.println(" 日本円  :" + sum + " 円 ");
        System.out.println(" 英ポンド :" + pound + " ポンド " + pense +
                           " ペンス ");
    }
}
```

3.

```
class FarMoon{
    public static void main(String[] args){
        final double DISTANCE = 3.85E5;
        final int SECONDS_PER_HOUR = 60*60;
        double time;
        int hours;
        int minutes;
        int seconds;

        //時間の計算
        time = DISTANCE / (8 * SECONDS_PER_HOUR);    //所要時間＝距離÷時速
        hours = (int)time;                           //端数を切り捨て時間の整数部分だけ取り出す
        time -= hours;                               //少数部分を time に代入
        //分の計算
        time = time * 60;                            //時間を分に変換
        minutes = (int)time;                         //端数を切り捨て分の整数部分だけ取り出す
        //画面表示
        System.out.print(" 月への飛行時間は、");
        System.out.println(hours + " 時間 " + minutes + " 分です。");
    }
}
```

■第 3 章 §1 練習問題 (p.40)

1．(1)

■実行結果

```
1234567890123
-------------
12345
 12345
       12345
      12,345
000000012345
12345
```

(2)

■実行結果

```
1234567890123456
-----------------
12345.678900
    12345.678900
  +12,345.678900
   12345.68
      12345.6789
12345.6789
```

2．
① `import java.util.Scanner;`
② `new Scanner(System.in);`
③ `yen = scan.nextDouble();`
④ `"米ドル　：%8.4f ドル¥n", dollar`
⑤ `"英ポンド：%8.4f ポンド¥n", pound`
⑥ `"欧ユーロ：%8.4f ユーロ¥n", euro`

8.4f は、整数部分 8 けた、小数部分 4 けたではなく、小数部分 4 けた、小数点 1 けた、整数部分 8 － 4 － 1 ＝ 3 けた になることに注意する。

■第 3 章 §1 実習問題 (p.41)

1．

```
import java.util.Scanner;

class SalesTax{
    public static void main(String[] args){
        int netPrice;           //本体価格
        int salesPrice;         //販売価格
        double taxRate = 0.08;  //消費税率
        int tax;                //消費税額

        Scanner scan = new Scanner(System.in);

        System.out.print("販売価格を入力してください。:");
        salesPrice = scan.nextInt();
        netPrice = (int)(salesPrice / (1 + taxRate) + 0.5); //四捨五入
        tax = salesPrice - netPrice;

        System.out.printf("本体価格：%,d 円¥n", netPrice);
        System.out.printf("消費税　：%,d 円¥n", tax);
    }
}
```

四捨五入には、0.5 を加えて小数点以下を切り捨てる方法がよく用いられる。
Java には、四捨五入のためのメソッドも用意されており、次節で学ぶ。

2.

```
import java.util.Scanner;

class FarMoon02{
    public static void main(String[] args){
        final double DISTANCE = 3.85E5;
        final int SECONDS_PER_HOUR = 60 * 60;
        double time;
        int hours;
        int minutes;
        int seconds;
        double speed;

        Scanner scan = new Scanner(System.in);
        System.out.print("平均速度（秒速）を入力してください（km）。: ");
        speed = scan.nextDouble();

        //時間の計算
        time = DISTANCE / (speed * SECONDS_PER_HOUR); //時間＝距離÷時速
        hours = (int)time;       //端数を切り捨て時間の整数部分だけ取り出す
        time -= hours;           //少数部分を time に代入

        //分の計算
        time = time * 60;        //時間を分に変換
        minutes = (int)time;     //端数を切り捨て分の整数部分だけ取り出す
        time -= minutes;         //少数部分を time に代入  time = time - minutes

        //秒の計算
        seconds = (int)(time *= 60);          //分を秒に換算

        //画面表示
        System.out.printf("所要時間は、%d 時間 %d 分 %d 秒です。¥n",
                                              hours, minutes, seconds);
    }
}
```

■第 3 章 §2 練習問題 (p.46)

(1) ■実行結果

```
pow(7,3) = 343.000000
sqrt(5) = 2.236068
abs(-3.45) = 3.45
ceil(34.56) = 35.00
floor(-34.56) = -35.00
round(123.45) = 123
```

(2) ■実行結果

```
toRadians(36) = 0.628319
sin(toRadians(36)) = 0.587785
cos(toRadians(36)) = 0.809017
tan(toRadians(36)) = 0.726543
```

■第 3 章 §2 実習問題 (p.46)

1.

```
class Pow2Byte{
    public static void main(String[] args){
        System.out.printf("1KByte = %,18.0fByte¥n", Math.pow(2, 10));
        System.out.printf("1MByte = %,18.0fByte¥n", Math.pow(2, 20));
        System.out.printf("1GByte = %,18.0fByte¥n", Math.pow(2, 30));
        System.out.printf("1TByte = %,18.0fByte¥n", Math.pow(2, 40));
    }
}
```

2.

```
import java.util.Scanner;

class Radius{
    public static void main(String[] args){
        double radius;          // 円の半径
        double circum;          // 円周
        double area;            // 円の面積
        double surface;         // 球の表面積
        double volume;          // 球の体積

        Scanner scan = new Scanner(System.in);

        System.out.print("半径を入力してください。: ");
        radius = scan.nextDouble();

        circum = 2 * Math.PI * radius;
        area = Math.PI * Math.pow(radius, 2);
        surface = 4 * Math.PI * Math.pow(radius, 2);
        volume = 4 / 3 * Math.PI * Math.pow(radius, 3);

        System.out.printf(
            "半径%.2fの円の円周は、%.2fです。¥n", radius, circum);
        System.out.printf(
            "半径%.2fの円の面積は、%.2fです。¥n", radius, area);
        System.out.printf(
            "半径%.2fの球の表面積は、%.2fです。¥n", radius, surface);
        System.out.printf(
            "半径%.2fの球の体積は、%.2fです。¥n", radius, volume);
    }
}
```

■第4章 §1 練習問題 (p.52)

1. (1)

```
if(point>=60)
    System.out.println("合格です。");
else
    System.out.println("不合格です。");
```

(2)

```
if(x<640)
    x++;
else
    x--;
```

(3)

```
if(year%4 == 0)
    leap++;
```

(4)

```
if((Math.pow(b,2)-4*a*c) >= 0)
    System.out.println("解は実数");
else
    System.out.println("解は虚数");
```

2. (1) n%3==0 && n%5!=0
   (2) n>=1 && n<=31 && n%7==0
   (3) n%2==0 || n%5==0
   (4) c!='y' && c!='n'

■第 4 章 §1 実習問題 (p.52)

1.

```
import java.util.Scanner;

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

        System.out.print("西暦年を入力してください。:");
        year = scan.nextInt();
        if((year % 400 == 0) || (year % 4 == 0) && (year % 100 != 0))
            System.out.printf("%d 年はうるう年です。", year);
        else
            System.out.printf("%d 年はうるう年ではありません。", year);
    }
}
```

2. (1)

```
class ChouHan01{
    public static void main(String[] args){
        int die1;
        int die2;
        int sum;

        System.out.println(" ２つのサイコロを振ります。");
        die1 = (int)(Math.random() * 6) + 1;
        die2 = (int)(Math.random() * 6) + 1;
        sum = die1 + die2;

        System.out.printf(" サイコロ１：%d¥n", die1);
        System.out.printf(" サイコロ２：%d¥n", die2);
        System.out.printf(" 合　　計：%d¥n", sum);
        System.out.println();

        if(sum%2==0)
            System.out.println(" 丁！ ");
        else
            System.out.println(" 半！ ");
    }
}
```

(2)

```
import java.util.Scanner;

class ChouHan02{
    public static void main(String[] args){
        int inputValue;
        int die1;
        int die2;
        int sum;
        boolean inputIsEven = false;
        boolean diceIsEven = false;

        Scanner scan = new Scanner(System.in);
        System.out.print(" 好きな数を入力してください。:");
            inputValue = scan.nextInt();

        if(inputValue%2==0){
            inputIsEven = true;
            System.out.println(" 入力値は、丁！ ");
        }else{
            System.out.println(" 入力値は、半！ ");
        }
```

（次ページへ続く）

```
        die1 = (int)(Math.random() * 6) + 1;
        die2 = (int)(Math.random() * 6) + 1;
        sum = die1 + die2;
        System.out.println();
        System.out.println("サイコロを振ります。");
        System.out.printf("サイコロ１：%d¥n", die1);
        System.out.printf("サイコロ２：%d¥n", die2);
        System.out.printf("合　　計：%d¥n", sum);

        if(sum%2==0){
            diceIsEven = true;
            System.out.println("出目は、丁！ ");
        }else{
            System.out.println("出目は、半！ ");
        }
        System.out.println();
        if(inputIsEven == diceIsEven)    // ともにTrueか、ともにfalseのとき
            System.out.println("あなたの勝ち！ ");
        else
            System.out.println("あなたの負け！ ");
    }
}
```

それぞれ真偽値を持つ２つの命題があり、その一方が true で他方が false のとき true を返し、反対にともに true かともに false のとき false を返す論理演算を排他的論理和（exclusive or ：XOR）といい、Java では「^」(山型記号）であらわす。勝ち負けの判定部分をこの排他的論理和を使って書くと、次のようになる。

```
if(inputIsEven ^ diceIsEven)// 排他的論理和がtrueなら負け、falseなら勝ち
    System.out.println("あなたの負け！ ");
else
    System.out.println("あなたの勝ち！ ");
```

■第４章 §２練習問題 (p.56)

１．(1)

①　ポーションと西の塔の鍵

②

```
if(posX == 7){
    if(posY == 3)
        System.out.println("なんと，ポーションを手に入れた。");
    else{
        System.out.println("なんと，uPadを手に入れた。");
        System.out.println("なんと，錦の御旗を手に入れた。");
        System.out.println("なんと，西の塔の鍵を手に入れた。");
    }
}
```

③

```
if(posX == 7){
    if(posY == 3)
        System.out.println("なんと，ポーションを手に入れた。");
    else
        System.out.println("なんと，uPadを手に入れた。″);
    System.out.println("なんと，錦の御旗を手に入れた。");
    System.out.println("なんと，西の塔の鍵を手に入れた。");
}
```

(2)

```
if(posX == 7){
    if(posY == 3)
        System.out.println("なんと、ポーションを手に入れた。");
    else{
        System.out.println("なんと、uPad を手に入れた。");
        System.out.println("なんと、錦の御旗を手に入れた。");
        System.out.println("なんと、西の塔の鍵を手に入れた。");
    }
}
```

【別解】下記でも同じ結果が得られる。

```
if(posX == 7){
    if(posY == 3)
        System.out.println("なんと，ポーションを手に入れた。");
    else
        System.out.println("なんと，uPad を手に入れた。˝);
    System.out.println("なんと，錦の御旗を手に入れた。");
    System.out.println("なんと，西の塔の鍵を手に入れた。");
}
```

2．「パー」が表示される。

if 文の判定式が boo == false ではなく、boo = false なので、boo の値にかかわらず、式の値は false。またこの式の結果、boo の値も false になるので、else if 文の判定式も false。したがって、表示されるのは「パー」になる。

■第 4 章 §2 実習問題 (p.57)

1．

```
import java.util.Scanner;

class Era{
    public static void main(String[] args){
        int year;
        String era = "";
        Scanner scan = new Scanner(System.in);

        System.out.println("西暦を元号に変換します。" );
        System.out.print("1868 年以降の西暦年を入力してください。:");
        year = scan.nextInt();

        if (year<1868)
            System.out.println("1868 年以降の西暦年を入力してください。");
        else if(year<1912){
            era = "明治";
            year -= 1868 - 1;
        }
        else if(year<1926){
            era = "大正";
            year -= 1912 - 1;
        }
        else if(year<1989){
            era = "昭和";
            year -= 1926 - 1;
        }
        else{
            era = "平成";
            year -= 1989 - 1;
        }
        System.out.printf("%s%d 年です。", era, year);
    }
}
```

各元号の元年は 0 年ではなく 1 年のため、西暦ベースの数式で 1 をマイナスしている。

2.

```
import java.util.Scanner;

class Discomfort{
    public static void main(String[] args){
        double temp;     // 気温
        double hum;      // 湿度
        double di;       // 不快指数
        Scanner scan = new Scanner(System.in);

        // 気温と湿度の入力
        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();

        // 不快指数の計算
        di = 0.81 * temp + 0.01 * hum * (0.99 * temp - 14.3) + 46.3;

        // 結果の表示
        if (di<=55)
            System.out.println("寒いです。");
        else if(di<=60)
            System.out.println("肌寒いです。");
        else if(di<=75)
            System.out.println("快適です。");
        else if(di<=80)
            System.out.println("やや不快です。");
        else if(di<=85)
            System.out.println("不快です。");
        else
            System.out.println("超不快です。");
    }
}
```

3.

```
import java.util.Scanner;

public class TaxiFee{
    public static void main(String[] args){
        double distance;     // 走行距離
        int fee = 0;         // タクシー料金

        // 走行距離の入力
        Scanner scan = new Scanner(System.in);
        System.out.print("距離を入力してください（km）：");
        distance = scan.nextDouble();

        // 料金計算と表示
        if(distance <= 2.0)
            fee = 660;
        else
            fee = 660 +
              (int)(Math.ceil((Math.ceil((distance - 2) / 0.274) * 80) / 10)) * 10;

        System.out.printf(
            "走行距離 %.2fkm のタクシー料金は %d 円です。¥n", distance, fee);
    }
}
```

2km を超えると 274m ごとに 80 円が加算されるのであるから、たとえば、2.2km では 660 円＋ 80 円の 740 円になる。つまり、`(distance - 2) / 0.274` の端数は切り上げる必要がある。料金の計算式の内側の `ceil()` メソッドはこの処理を行っている。また、10 円未満の端数の切り上げは、km 当たり料金を計算した後、それを 10 で割り、その商の小数点未満を外側の `ceil()` メソッドで切り上げ、その後 10 をかけて計算している。

■第 4 章 §3 練習問題 (p.62)

(1) 使用可能なデータ型
　short　　int　　char
(2) ① 100 のとき
　　いかめし（森駅）
　　峠の釜めし（横川駅）
　② 200 のとき
　　峠の釜めし（横川駅）
　③ 300 のとき
　　ひっぱりだこ飯（西明石駅）
　　甲州かつサンド（小淵沢駅）
　④ 400 のとき
　　甲州かつサンド（小淵沢駅）

(1)　switch 文の判定式のデータ型としては、一般的には long 型を除く整数型（byte、short、char、int）が使用できる。しかし、この問題では i の値が byte 型の範囲（-128 ～ +127）を超えるので、byte 型を使用するとコンパイルエラーになる。また、char 型は使用可能だが、case 文のリテラルが文字ではないので、何を入力しても default の「甲州かつサンド」を表示する。
(2)　break 文の有無に注意する。

■第 4 章 §3 実習問題 (p.62)

1.

```
import java.util.Scanner;

class Grade{
    public static void main(String[] args){
        int score;
        String grade;

        Scanner scan = new Scanner(System.in);
        System.out.print("得点を入力してください (0 ～ 100)。:");
        score = scan.nextInt();

        switch (score/10){
            case 10: case 9:
                grade = "AA";
                break;
            case 8:
                grade = "A";
                break;
            case 7:
                grade = "B";
                break;
            case 6:
                grade = "C";
                break;
            case 5:case 4:
                grade = "再試験";
                break;
            default:
                grade = "F";
                break;
        }
        System.out.printf("あなたの成績は，%s です。", grade);
    }
}
```

得点別評価の分類と得点の整数除算の結果の分類が一致していることを利用する。

2.

```
class Omikuji{
    public static void main(String[] args){
        int kuji;
        //0～9の範囲の数を乱数で生成
        kuji = (int)(Math.random()*10);
        System.out.println("*** お告げ ***");

        switch (kuji){
            case 1:
            case 2:
                System.out.println("大吉です。");
                break;
            case 3:
            case 4:
            case 5:
            case 6:
                System.out.println("吉です。");
                break;
            case 7:
            case 8:
            case 9:
                System.out.println("凶です。");
                break;
            default:          //i=0のとき
                System.out.println("大凶です。");
                break;
        }
    }
}
```

■第5章 §1 練習問題 (p.67)

1.

	初期値	継続条件	増分
(1)	int i=2;	i<=20;	i+=3)
(2)	int i=50;	i>=0;	i-=7)
(3)	int i=1;	i<=16;	i+=i)

2. (1) ④　(2) ①

入力された文字列をその末尾の文字から順に取り出し、取り出した順に、前からつなげて新しい文字列を生成している。

■第5章 §1 実習問題 (p.68)

1.

```
import java.util.Scanner;

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

        System.out.print("いくつまでの総和を求めますか。:");
        num = scan.nextInt();

        for (int i=1; i<=num; i++)
            sum += i;

        System.out.printf("¥n 1から%d までの総和： %d ", num, sum);
    }
}
```

2．(1)

```
class Interest01{
    public static void main(String[] args){
        double principal = 1200000.0;  //元金
        double rate = 0.0185;          //年利
        double amount;                 //元利合計

        System.out.printf("%s%12s¥n",″年″, ″元利合計 ");    //ヘッダー
        for(int year=1; year<=15; year++){
            amount = principal * Math.pow(1.0 + rate, year);
            System.out.printf("%2d%,18.2f¥n", year, amount);
        }
    }
}
```

(2)

```
import java.util.Scanner;

class Interest02{
    public static void main(String[] args){
        int  principal;
        double rate;
        double amount;
        int term;           //期間
        Scanner scan = new Scanner(System.in);

        System.out.print(" 元金を入力してください。: ");
        principal = scan.nextInt();
        System.out.print(" 金利を入力してください。: ");
        rate = scan.nextDouble();
        System.out.print(" 期間を入力してください。: ");
        term = scan.nextInt();

        System.out.printf("%s%12s¥n"," 年 ", " 元利合計 ");

        for(int year=1; year<=term; year++){
            amount = principal * Math.pow(1.0 + rate, year);
            System.out.printf("%2d%,18.2f¥n", year, amount);
        }
    }
}
```

3．

```
class Ancestors{
    public static void main(String[] args){
        int ancestors = 0;

        for(int i=1; i<=10; i++){
            // 1 世代ごとに 2 のべき乗で増えていく。全部加算する。
            ancestors += Math.pow(2, i);
            System.out.printf(
             "%2d 世代前までのあなたの直系親族は %5d 人です。¥n", i, ancestors);
        }
    }
}
```

4.

```
class FizzBuzz{
    public static void main(String[] args){
        for(int i=1; i<=100; i++){
            if(i%15==0)
                System.out.println("FizzBuzz");
            else if(i%3==0)
                System.out.println("Fizz");
            else if(i%5==0)
                System.out.println("Buzz");
            else
                System.out.println(i);
        }
    }
}
```

【別解】

```
class FizzBuzzCase{
    public static void main(String[] args){
        for(int i=1; i<=100; i++){
            switch (i%15){
                case 0:
                    System.out.println("FizzBuzz");
                    break;
                case 3: case 6: case 9: case 12:
                    System.out.println("Fizz");
                    break;
                case 5: case 10:
                    System.out.println("Buzz");
                    break;
                default:
                    System.out.println(i);
            }
        }
    }
}
```

■第 5 章 §2 練習問題 (p.73)

(1) ④　　(2) ④

■第 5 章 §2 実習問題 (p.74)

1. (1)

```
class Stars01{
    public static void main(String[] args){
        for(int i=1; i<=10; i++){
            for(int j=1; j<=i; j++)
                System.out.print("*");
            System.out.println();
        }
    }
}
```

(2)

```
class Stars02{
    public static void main(String[] args){
        for(int i=1; i<=10; i++){
            for(int j=1; j<=10-i+1; j++)
                System.out.print("*");
            System.out.println();
        }
    }
}
```

(3)

```
class Stars03{
    public static void main(String[] args){
        int i,j,k;
        for(i=1; i<=10; i++){
            for(j=0; j<=i-2; j++)
                System.out.print(" ");
            for(k=0; k<=10-i; k++)
                System.out.print("*");
            System.out.println();
        }
    }
}
```

(4)

```
class Stars04{
    public static void main(String[] args){
        int i,j,k;
        for(i=1; i<=10; i++){
            for(j=1; j<=10-i; j++)
                System.out.print(" ");
            for(k=1; k<=i; k++)
                System.out.print("*");
            System.out.println();
        }
    }
}
```

2.

```
import java.util.Scanner;

class MultiTable{
    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++){
            for(int j=1; j<=scale; j++)
                System.out.printf("%3d  ", i*j);
            System.out.println();
        }
    }
}
```

3.

```
class Triangle{
    public static void main(String args[]){
        int counter = 0;
        int loopCounter = 0;
        int a,b,c;

        for(int i=1; i<=100; i++){
            for(int j=1; j<=i; j++){
                for(int k=1; k<=j; k++){
                    a = (int)Math.pow(i, 2);
                    b = (int)Math.pow(j, 2);
                    c = (int)Math.pow(k, 2);
                    loopCounter++;
                    if (a == b + c){
                        counter++;
                        System.out.printf("%3d :%3d :%3d¥n", i, j, k);
                    }
                }
            }
        }
```

（次ページへ続く）

```
            System.out.printf(
                  "直角三角形となる100以下の3辺の数値の組は%dあります。¥n", counter);
            System.out.printf("総ループ回数は%,d回です。", loopCounter);
        }
    }
```

■第5章 §3 練習問題 (p.78)

(1) ① D ② F ③ B

① a : 0 → 1 → 2 → 3 → 4　　b : 0 → 1 → 3 → 6 → 10

② while文の繰り返し条件が単独の変数の場合、その変数がboolean型以外はコンパイルエラーになる。

③ 内側のwhile文は1度も実行されない。

(2) ① C ② D ③ B

① まず、内側のdo ～ while文でbを0から4まで変化させ、それを表示する。
次に、外側のwhile文を4回繰り返し、そのつど内側のdo ～ while文でbの値を5から8まで変化させ、それを表示する。

② a : 0 → 1 → 2 → 3 → 4 → 5
b : 0 → 1
0 → 2
1 → 4
3 → 7
6 → 11
10

③ 12345は内側のdo ～ while文の繰り返しで表示され、678はそれぞれ外側のwhile文で、aの値が2、3、4のとき、内側のdo ～ while文を1回ずつ通過するたびに表示される。

■第5章 §3 実習問題 (p.80)

1.

```
class DayDream{
    public static void main(String args[]){
        int day = 1;
        int income = 1;

        while (income<=100000000){
            day++;
            income *= 2;
        }
        System.out.printf(
            "小遣いが1億円を超えるのは%d日目で,¥n額は%,d円です。", day, income);
    }
}
```

2.

```
class PlusUp{
    public static void main(String[] args){
        int num = 0;
        int max = 10000;
        int sum = 0;

        do{
            num++;
            sum += num;
        }while(sum<=max);
        System.out.printf("総　　和 = %,d¥n", sum);
        System.out.printf("最後の数 = %d¥n", num);
    }
}
```

3．(1)

```
class CalcPai01{
    public static void main(String[] args){
        double pai = 4.0;          // 第 1 項を初期値設定
        double term;               // 各項の値
        int div;                   // 各項の分母

        // 両辺に 4 をかけ、π = 4 - 4/3 + 4/5 - 4/7 + ··· として第 2 項以下を計算。
        for(int i=2; i<=50; i++){
            div = 2*i-1;           //i 項の分母
            term = 4.0/div;
            if(i%2==0)             // 偶数項は減算、奇数項は加算
                pai -= term;
            else
                pai += term;
            System.out.printf(" 第 %2d 項  %1.10f¥n", i, pai);
        }
    }
}
```

(2)

```
class CalcPai02{
    public static void main(String[] args){
        int counter = 1;
        double pai = 4.0;
        double term;
        int div;

        // 両辺に 4 をかけ、π = 4 - 4/3 + 4/5 -4/7 + ····· として計算
        while(Math.abs(Math.PI-pai) > 1e-2){
            counter++;
            div = 2*counter-1;
            term = 4.0/div;
            if(counter%2==0)
                pai -= term;
            else
                pai += term;
            //System.out.printf("%3d  %1.10f  %1.10f¥n",
            //    counter, pai, Math.abs(Math.PI-pai));     ←確認用
        }
        System.out.printf("¥n 差が 0.01 以下になるのは %d 項以降 ", counter);
    }
}
```

4．

```
class Parabola{
    public static void main(String[] args){
        final double GR = 9.8;
        double firstSpeed = (double)150 * 1000 / (60 * 60);
        double angle = 40.0;
        double x = 0.0;
        double y = 0.0;
        double time = 0.0;

        System.out.printf(" 時間（秒）  距離 (m)      高さ (m)¥n");
        do{
            time += 0.1;
            x = firstSpeed
                    * Math.cos(Math.toRadians(angle)) * time;
            y = firstSpeed
                    * Math.sin(Math.toRadians(angle)) * time
                    - GR * Math.pow(time, 2) / 2;

            System.out.printf("%2.1f%12.2f%12.2f¥n", time, x, y);
        }while(y > 0);
    }
}
```

■第 6 章 §1 練習問題 (p.86)

1．(2) … 配列の要素数のデータ型は long 型を除く整数型。浮動小数点数型は使用できない。
(5) … 配列の初期値を一括設定する方法では、要素数を数値で指定できない。

2．(1) `String[] gengo = {"明治", "大正", "昭和", "平成"};`
(2) `for(int i=0; i<bool.length; i++)`
(3) `int[] pow = new int[6];`
(4) `char[] alfa = new char[26];`

3．(1) ③　(2) ④　(3) ①

■第 6 章 §1 実習問題 (p.89)

1．

```
class AccessRank{
   public static void main(String[] args){
       int max = 0;          //仮の最大値
       int min = 10000;      //仮の最小値
       int sum = 0;
       double average;
       int[] access =
           {3736,2892,3840,2592,3014,2175,3776,2992,2857,2306,2881,3243};

       for(int i=0; i<access.length; i++){
           if(access[i] > max)
               max = access[i];
           if(access[i] < min)
               min = access[i];
           sum += access[i];

           System.out.printf("%2d月    %,d¥n", i+1, access[i]);
       }
       average = (double)sum/access.length;
       System.out.printf("最高アクセス数：%d¥n", max);
       System.out.printf("最低アクセス数：%d¥n", min);
       System.out.printf("年間アクセス数：%d¥n",sum);
       System.out.printf("月別平均アクセス数：%.1f¥n", average);
   }
}
```

2．

```
class FibArray{
    public static void main(String[] args){
        long[] fib = new long[50];
        fib[0] = fib[1] = 1;

        for(int i=2; i<fib.length; i++)
            fib[i] = fib[i-1] + fib[i-2];

        System.out.println("【Fibonacci 数列の最初の 50 項】");        //(1)
        for(int i=0; i<fib.length; i++)
            System.out.printf("第%2d項 %12d¥n", i+1, fib[i]);

        System.out.println("¥n【隣接項の比】");                       //(2)
        double rate;
        for(int i=1; i<fib.length; i+=2){
            rate = (double)fib[i]/fib[i-1];
            System.out.printf("%12d/%12d = %.15f¥n",
                                fib[i], fib[i-1], rate);
        }
    }
}
```

フィボナッチ数列の隣接項の比は、いわゆる「黄金比」
$(1 : (1 + \sqrt{5})/2 = 1 : 1.61803398874989484820 \cdots\cdots)$ に収束する。

■第 6 章 §2 練習問題 (p.94)

1．(1) Chloe と Juliet
(2) Maria と Tristan
(3) Romeo と Isolde

2． (1) ② (2) ⑤ (3) ⑥ (4) ①

■第 6 章 §2 実習問題 (p.95)

```
class Exam{
    public static void main(String[] args){
        // あらかじめ得点合計欄を付加した配列を初期値設定
        int[][] test = {
            {5001,46,58,62,0},
            {5002,72,80,45,0},
            {5003,68,62,56,0},
            {5004,57,47,15,0},
            {5005,86,94,52,0},
            {5006,52,59,25,0},
            {5007,37,55,79,0},
            {5008,63,50,35,0},
            {5009,91,79,80,0},
            {5010,77,61,35,0}
            };
        int sum = 0;
        double average;

        // 3科目得点を test[i][4] に集計
        for(int i=0; i<test.length; i++){
            for(int j=0; j<test[i].length; j++){
                test[i][4] = test[i][1] + test[i][2] + test[i][3];
            }
            sum += test[i][4];       //3 科目得点合計の集計
        }
        //3 科目合計の平均点を求める
        average = (double)sum/test.length;

        // 結果の表示
        System.out.println(" 番号 英語 常識 ﾌﾟﾛｸﾞ 合計  結果 ");
        for(int i=0; i<test.length; i++){
            for(int j=0; j<test[i].length; j++){
                System.out.printf("%4d ", test[i][j]);
            }
            if ((double)test[i][4] >= average)
                System.out.printf(" 1次通過 ¥n");
            else
                System.out.println();
        }
        System.out.println();
        System.out.printf(" 3科目合計の平均点：%3.2f", average);
    }
}
```

■第 6 章 §3 練習問題 (p.101)

		(1)	(2)
1.	①	2	1
	②	26	65
	③	9000	1000
	④	15	-3

(1) は上限値 −下限値 + 1、(2) は下限値になる。

2. (1) ⑤ (2) ④ (3) ① (4) ② (5) ③

■第 6 章 §3 実習問題 (p.102)

1. (1)

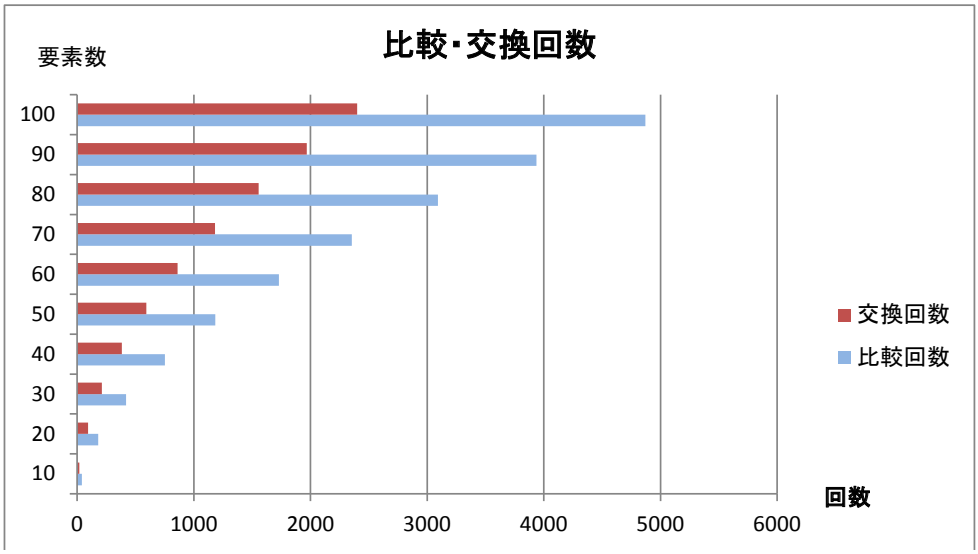
```
import java.util.Random;

class BubbleSort01{
    public static void main(String args[]){
        Random rnd = new Random();
        int[] myArray = new int[10];
        // カウンタの設定・初期化
        int compCounter = 0;     // 比較回数カウンタ
        int swapCounter = 0;     // 交換回数カウンタ

        for(int i=0; i<myArray.length; i++)
             myArray[i] = rnd.nextInt(10) + 1;

        System.out.print(" 整列前 : ");
        for(int i=0; i<myArray.length; i++)
            System.out.printf("%2d ", myArray[i]);
        System.out.print("¥n¥n");

        boolean isSwap;
        for(int line=1; line<myArray.length; line++){
            isSwap = false;
            for(int idx=0; idx<myArray.length-line; idx++){
                compCounter++;       // 比較回数 +1
                if (myArray[idx] > myArray[idx+1]){
                    int temp = myArray[idx];
                    myArray[idx] = myArray[idx+1];
                    myArray[idx+1] = temp;
                    isSwap = true;
                    swapCounter++;  // 交換回数 +1
                }
            }
            if(isSwap){
                System.out.print(" 整列中 : ");
                for(int i=0; i<myArray.length; i++)
                    System.out.printf("%2d ",myArray[i]);
                System.out.println();
            }else{
                break;
            }
        }

        System.out.print("¥n 整列後 : ");
        for(int i=0; i<myArray.length; i++)
            System.out.printf("%2d ", myArray[i]);
        System.out.println();
        System.out.printf(" 比較回数 : %d¥n", compCounter);
        System.out.printf(" 交換回数 : %d¥n", swapCounter);
    }
}
```

(2) 解答略

配列の要素数を 10 から 100 まで 10 きざみで変化させ、各比較・交換回数を計測するプログラムを作成して実行した結果の一例およびそのグラフは、次のとおりである。
交換回数は比較回数のほぼ 1/2 で、要素数が倍になると回数はほぼ 4 倍になっていることがわかる。

要素数	比較回数	交換回数
10	40. 66	19. 87
20	179. 80	92. 78
30	418. 89	211. 90
40	752. 85	382. 82
50	1185. 22	592. 72
60	1728. 25	860. 10
70	2354. 98	1182. 45
80	3093. 57	1554. 66
90	3937. 45	1968. 35
100	4869. 70	2401. 37

（各要素数ごとに 100 回試行の平均値を表示）



2．(1)

```
class IntArraySort01{
    public static void main(String[] args){
        int[] myArray = {33,57,4,11,74,54,93,29,6,82,47,17,61,76,15};
        for(int i=1; i<myArray.length; i++){
            boolean isSwap = false;
            for(int j=0; j<myArray.length-i; j++){
                if(myArray[j] > myArray[j+1]){
                    int temp = myArray[j];
                    myArray[j] = myArray[j+1];
                    myArray[j+1] = temp;
                    isSwap = true;
                }
            }
            if(!isSwap)
                break;
        }
        for(int i=0; i<myArray.length; i++)
            System.out.printf("%d  ", myArray[i]);
    }
}
```

(2)

上記 IntArraySort01.java の 7 行目を、if(myArray[j] < myArray[j+1]) と不等号の向きを反対にするだけでよい。

3．

```
import java.util.Random;

class RandomBar{
    public static void main(String[] args){
        int die1, die2, sumOfDice;
        int face[] = new int[11];        //2 ～ 12 までの要素数 11 の配列
        int arrayIndex;
        int tenCounter, oneCounter;

        // 2 つのさいころの目を乱数で出力し、その出現回数を出目ごとに配列に記憶する
        Random rnd = new Random();
        for(int i=1; i<=1000; i++){
            die1 = 1 + rnd.nextInt(6);
            die2 = 1 + rnd.nextInt(6);
            sumOfDice = die1 + die2;
            arrayIndex = sumOfDice - 2; // 出目 (2 ～ 12) と添字 (0 ～ 10) を調整
            face[arrayIndex]++;
        }
        // 出現回数をグラフ表示
        System.out.println(" 出目 回数 グラフ ----------");
        for(int i=0; i<=10; i++){
            tenCounter = face[i]/10;
            oneCounter = face[i]%10;

            System.out.printf("%2d:%5d  ", i+2, face[i]);
            for(int j=1; j<=tenCounter; j++)
                System.out.print("*");
            for(int k=1; k<=oneCounter; k++)
                System.out.print(".");
            System.out.println();
        }
    }
}
```

■第 7 章 §1 練習問題 (p.110)

1．(1) ③　(2) ④　(3) ③　(4) ①

(1) クラス変数 …… 静的フィールド
インスタンス変数 …… 非静的フィールド
オブジェクト変数 …… インスタンスへの参照を記憶する変数
間違いやすいので注意する。

(2) メソッドの仮引数には変数しか使用できない。

(3) 制御変数の有効範囲はその制御構造のなかだけである。

(4) バブルソートはデータ数が 5 倍になると処理時間は 25 倍になる。
配列の添え字には long 型は使用できない。

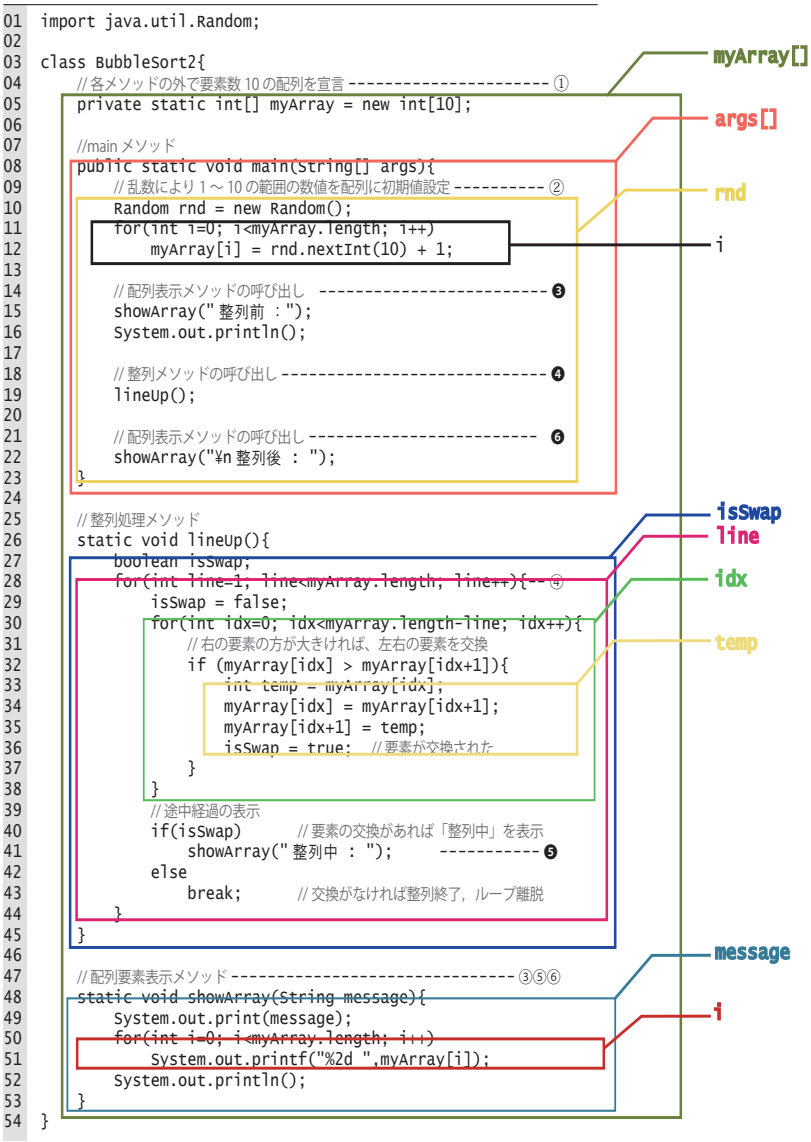
2．　次ページ（p.26）参照

3．②

次の 3 つの変数 x の働きの違いに注意する
静的フィールド x(=20)
printLocal() メソッドのローカル変数 x(=10)
main() メソッドのローカル変数 x(=50)

ローカル変数はその属するメソッドが呼び出されるつど初期化される。これに対して静的フィールドは、プログラムの存続中その値を保持する。

【例題 16 の各変数の有効範囲】




```
01  import java.util.Random;
02
03  class BubbleSort2{
04      //各メソッドの外で要素数 10 の配列を宣言 ---------- ①
05      private static int[] myArray = new int[10];
06
07      //main メソッド
08      public static void main(String[] args){
09          //乱数により 1 ～ 10 の範囲の数値を配列に初期値設定 ---------- ②
10          Random rnd = new Random();
11          for(int i=0; i<myArray.length; i++)
12              myArray[i] = rnd.nextInt(10) + 1;
13
14          //配列表示メソッドの呼び出し  ----------❸
15          showArray("整列前 :");
16          System.out.println();
17
18          //整列メソッドの呼び出し ----------❹
19          lineUp();
20
21          //配列表示メソッドの呼び出し ----------  ❻
22          showArray("¥n 整列後 : ");
23      }
24
25      //整列処理メソッド
26      static void lineUp(){
27          boolean isSwap;
28          for(int line=1; line<myArray.length; line++){ ④
29              isSwap = false;
30              for(int idx=0; idx<myArray.length-line; idx++){
31                  //右の要素の方が大きければ、左右の要素を交換
32                  if (myArray[idx] > myArray[idx+1]){
33                      int temp = myArray[idx];
34                      myArray[idx] = myArray[idx+1];
35                      myArray[idx+1] = temp;
36                      isSwap = true;  //要素が交換された
37                  }
38              }
39              //途中経過の表示
40              if(isSwap)       //要素の交換があれば「整列中」を表示
41                  showArray("整列中 : ");      ----------❺
42              else
43                  break;       //交換がなければ整列終了，ループ離脱
44          }
45      }
46
47      //配列要素表示メソッド ---------- ③⑤⑥
48      static void showArray(String message){
49          System.out.print(message);
50          for(int i=0; i<myArray.length; i++)
51              System.out.printf("%2d ",myArray[i]);
52          System.out.println();
53      }
54  }
```

■第 7 章 §1 実習問題 (p.111)

1.

```
import java.util.Random;

class BubbleSort03{
    private static int[] myArray = new int[10];

    //main メソッド
        （省略 ･･･BubbleSort02 と同じ）
```

（次ページへ続く）

```
    //整列処理メソッド----------------------------------------
    static void lineUp(){
        for(int pass=1; pass<myArray.length; pass++){
            boolean isSwap = false;
            for(int i=0; i<myArray.length-1; i++){
                if (myArray[i] > myArray[i+1]){
                    swap(i, i+1);
                    isSwap = true;
                }
            }
            if(isSwap)
                showArray("整列中 : ");
            else
                break;
        }
    }
    //配列要素交換メソッド----------------------------------------
    static void swap(int left, int right){
        int temp = myArray[left];
        myArray[left] = myArray[right];
        myArray[right] = temp;
    }
    //配列要素表示メソッド----------------------------------------
          (省略･･･BubbleSort2と同じ)
    }
}
```

2.

```
import java.util.Random;

class Shuffle{
    static int myArray[] = new int[10];
    public static void main(String[] args){
        Random rnd = new Random();
        int rndNum;
        //配列に初期値設定
        for(int i=0; i<myArray.length; i++)
            myArray[i] = i;
        System.out.print("シャッフル前    ");
        showArray(myArray);
        System.out.println();
        //シャッフル
        for(int i=0; i<myArray.length; i++){
            rndNum = rnd.nextInt(10);
            swap(i, rndNum);
            System.out.printf("添字%dと%dの交換  ", i, rndNum);
            showArray(myArray);
        }
        //結果の表示
        System.out.println();
        System.out.print("シャッフル後    ");
        showArray(myArray);
    }
    //交換
    static void swap(int left, int right){
        int temp = myArray[left];
        myArray[left] = myArray[right];
        myArray[right] = temp;
    }
    //要素の表示
    static void showArray(int myArray[]){
        for(int i=0; i<myArray.length; i++)
            System.out.printf("%d  ", myArray[i]);
        System.out.println();
    }
}
```

■第 7 章 §2 練習問題 (p.118)

1．(1) ①文字列（または String）
(2) ②ラッパー ③ Character
(3) ④静的フィールド ⑤インスタンス変数
(4) ⑥参照型 ⑦フィールド
(5) ⑧ return

2． (1) C (2) A

(1) のローカル変数 ad は、メソッド func() が呼び出されるつど新しく生成され、初期値 5 が設定される。仮引数 num はつねに 10 なので、func() の返却値もつねに 15 になる。
(2) の静的フィールド ad はプログラムが実行されている間 int 型変数として維持されるので、メソッド func() が呼び出されるつど仮引数 num=10 が加算されていく。

■第 7 章 §2 実習問題 (p.119)

1．(1)

```
class SimpleAdder{
    public static void main(String[] args){
        int sum = 0;
        for(int i=0; i<args.length; i++)
            sum += Integer.parseInt(args[i]);

        System.out.printf("合計 = %d", sum);
    }
}
```

(2)

```
class RightToLeft{
    public static void main(String[] args){
        for(int i=args[0].length()-1; i>=0; i--)
            System.out.print(args[0].charAt(i));
    }
}
```

2．

```
import java.util.Scanner;

class MirrorNumber01{
    public static void main(String[] args){
        long num;
        long mirrorNum;
        Scanner scan = new Scanner(System.in);

        do{
            System.out.print("2 けた以上の整数を入力してください。: ");
            num = scan.nextInt();
        }while(num<10);

        // 1 回目の数字反転メソッドの呼び出し、返却値を mirrorNum に代入
        mirrorNum = mirror01(num);

        // 2 回目以降の反転メソッド呼び出し
        while(num != mirrorNum){
            System.out.printf(
                "%15d + %15d = %15d¥n", num, mirrorNum, num + mirrorNum);
            num += mirrorNum;
            mirrorNum = mirror01(num);
        }
        System.out.printf("¥n%d は回文数です。", num);
    }
```

（次ページへ続く）

```
    // 数値反転メソッド
    static long mirror01(long num) {
        long myReverse = 0;
        while(num != 0) {
            myReverse = (myReverse * 10) + (num % 10);
            num /= 10;
        }
        return myReverse;
    }
}
```

検証については、たとえば、次のように上記数値反転メソッドに printf 文を挿入して実行すれば、num と myReverse の変化のプロセスを画面表示することができる。

```
    // 数値反転メソッド
    static long mirror01(long num){
        long myReverse = 0;
        String space = "                                          ";
        System.out.printf(
              "%smyReverse=%9d  num=%9d¥n", space, myReverse, num);
        while(num != 0) {
            myReverse = (myReverse * 10) + (num % 10);
            num /= 10;
            System.out.printf(
              "%smyReverse=%9d  num=%9d¥n", space, myReverse, num);
        }
        return myReverse;
    }
```

例題 16 の方法で 2 けたの数値はすべて回文数になる。3 けたの数値のうち、 196、295、394、689、788、790、879 の 7 つの数値およびその反転数値である 691、592、493、986、887、978 の 6 つ、合計 13 個の数値については、現在のコンピュータの計算能力をもってしても回文数になるかどうか判明していない。

3．(1)
```
static double area(double a, double b, double c){
    double s, area;
    s = (a + b + c) / 2;
    area = Math.sqrt(s * (s-a) * (s-b) * (s-c));
    return area;
}
```

(2)
```
static boolean isTriangle(double a, double b, double c){
     if((a<b+c) && (b<c+a) && (c<a+b))
        return true;
     else
        return false;
}
```

あるいは、
```
static boolean isTriangle(double a, double b, double c){
     if((a>=b+c) || (b>=c+a) || (c>=a+b))
         return false;
     else
         return true;
}
```

(3)

```
import java.util.Scanner;

class Trigon{
    public static void main(String[] args){
        double a, b, c;
        Scanner scan = new Scanner(System.in);

        System.out.print("三角形の第１辺の長さを入力してください。:");
        a = scan.nextDouble();
        System.out.print("三角形の第２辺の長さを入力してください。:");
        b = scan.nextDouble();
        System.out.print("三角形の第３辺の長さを入力してください。:");
        c = scan.nextDouble();

        if(isTriangle(a, b, 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(a, b, c));
        else
            System.out.printf(
                "%.2f, %.2f, %.2f は三角形の３辺を構成しません。",
                 a, b, c);
    }
    static double area(double a, double b, double c){
        double s, area;
        s = (a + b + c) / 2;
        area = Math.sqrt(s * (s-a) * (s-b) * (s-c));
        return area;
    }
    static boolean isTriangle(double a, double b, double c){
        if((a<b+c) && (b<a+c) && (c<b+a))
            return true;
        else
            return false;
    }
}
```

4.

```
class AmicableNum{
    public static void main(String[] args){
        int max = 20000;
        int n1, n2;

        for(int i=2; i<max; i++){
            n1 = calcSum(i);     //i の約数の総和
            n2 = calcSum(n1);    //(i の約数の総和）の約数の総和

            if ((i == n2) && (i < n1))
            //(i < n1) は、完全数 (i = n1) と友愛数同士の２重表記の排除

                System.out.printf("%5d と %5d は友愛数¥n", i, n1);
        }
    }
    // 約数の総和を求める
    static int calcSum(int n){
        int num = n;
        int max = num/2;         // ある数の 1/2 を超える約数はない
        int sum = 1;

        for(int i=2; i<=max; i++){
            if(num % i == 0)         // 余りが 0 なら約数
                sum += i;
        }
        return sum;
    }
}
```

5.

```
import java.util.Scanner;

class IsbnChecker{
    public static void main(String[] args){
        String inputBn;
        String isbn;
        int num;
        int sum = 0;
        int checkDigit;
        Scanner scan = new Scanner(System.in);

        do{    // 数字のけた数が 12 でなければ再入力
            System.out.print(
              "ISBN をハイフン付きで正しく入力してください。:");
            inputBn = scan.next();
            isbn = toDigits(inputBn);
        }while(isbn.length() != 12);

        for(int i=0; i<isbn.length(); i++){
            num = Integer.parseInt(isbn.substring(i, i+1));
            if(i%2 == 0)
                sum += num;
            else
                sum += num * 3;
        }
        checkDigit = (10 - sum%10) % 10;
        System.out.printf(
              "チェックデジット付き ISBN：%s-%d", inputBn, checkDigit);
    }
    // 数字以外を除去して数字列を返却
    static String toDigits(String str){
        StringBuilder sb = new StringBuilder();
        for(int i=0; i<str.length(); i++)
            if(Character.isDigit(str.charAt(i)))     // 文字が数字なら…
                sb.append(str.charAt(i));
        return sb.toString();
    }
}
```

■第 8 章 §1 練習問題 (p.128)

1. (1) (5) (7)

2. (1)

```
class MyCircle{
    // フィールド
    private int radius;
    // コンストラクタ
    public MyCircle(int radius){
        this.radius = radius;
    }
    // メソッド
    public int getRadius(){
        return radius;
    }
    public double getCircum(){
       return 2 * Math.PI * radius;
    }
    public double getArea(){
        return Math.PI * radius * radius;
    }
}
```

(2)

```
class Applicant{
    //フィールド
    int number;
    String name;
    int eng;
    int spi;
    int prog;
    int total;

    //コンストラクタ
    public Applicant(
        int number, String name, int eng, int spi, int prog){
        this.number = number;
        this.name = name;
        this.eng = eng;
        this.spi = spi;
        this.prog = prog;
        total = eng + spi + prog;
    }

    //メソッド
    public int getNumber(){
        return number;
    }
    public String getName(){
        return name;
    }
    public int getEng(){
        return eng;
    }
    public int getSpi(){
        return spi;
    }
    public int getSpi(){
        return spi;
    }
    public int getTotal(){
        return total;
    }
    public boolean isPassed(int ave){
        if(total>=ave)
            return true;
        else
            return false;
    }
}
```

■第 8 章 §1 実習問題 (p.129)

1.

```
public class MyCircleTest{
    public static void main(String[] args){
        Scanner scan = new Scanner(System.in);
        System.out.print("円の半径を入力してください。:");
        int myRadius = scan.nextInt();

        MyCircle circle = new MyCircle(myRadius);
        System.out.printf("半径%dの円の円周は%.2f、面積は%.2fです。",
            circle.getRadius(), circle.getCircum(),
            circle.getArea());
    }
}
```

2．(1)

```
public class SquareTest{
    public static void main(String[] args){
        Square sq1 = new Square('+', 8);
        Square sq2 = new Square('@', 5);
        sq1.display();
        System.out.println();
        sq2.display();
    }
}
```

(2)

```
class Square01{
    String str;
    int width;
    public Square01(String a, int b){
        str = a;
        width = b;
    }
    public void display(){
        for(int i=1; i<=width; i++){
            for(int j=1; j<=width; j++)
                System.out.print(str);
            System.out.println();
        }
    }
}
public class Square01Test{
    public static void main(String[] args){
        Square01 sq1 = new Square01("(~o~) ", 8);
        Square01 sq2 = new Square01("_(._.)_ ", 5);
        System.out.println(" 第九交響曲 ");
        sq1.display();
        System.out.println();
        System.out.println(" お詫びの言葉もございません。");
        sq2.display();
    }
}
```

■第 8 章 §2 練習問題 (p.137)

① java.util.Scanner クラスの nextDouble() メソッド
入力の次のトークンを double としてスキャンする。

② java.util.Random クラスの nextDouble() メソッド
乱数ジェネレータのシーケンスに基づいて 0.0 から 1.0 の範囲の一様分布の double 型の疑似乱数値を返す。

■第 8 章 §2 実習問題 (p.137)

1．

```
class MeijiMakers{
    private String name;        // 名前
    private int birthYear;      // 生年
    private int heaven;         // 没年
    private String cause;       // 死因
    private int lifeTime;       // 享年
```

（次ページへ続く）

```
    public MeijiMakers(
              String name, int birthYear, int heaven, String cause){
        this.name = name;
        this.birthYear = birthYear;
        this.heaven = heaven;
        this.cause = cause;
        setLifeTime();
    }
    public String getName(){
        return name;
    }
    public int getBirthYear(){
        return birthYear;
    }
    public int getHeaven(){
        return heaven;
    }
    public int getLifeTime(){
        return lifeTime;
    }
    public String getCause(){
        return cause;
    }
    public void setLifeTime(){
        lifeTime = heaven - birthYear + 1;
    }
}
```

2.

```
class TopBatter{
    private String name;          //選手名
    private int plate;            //打席数
    private int hit;              //安打数
    private int deadFour;         //四死球数
    private int bunt;             //犠打数
    private double onBase;        //出塁率

    public TopBatter(
           String name, int plate, int hit, int deadFour, int bunt){
        this.name = name;
        this.plate = plate;
        this.hit = hit;
        this.deadFour = deadFour;
        this.bunt = bunt;
    }

    public void setOnBase(){
        onBase = (double)(hit + deadFour)/(plate - bunt);
    }
    public String getName(){
        return name;
    }
    public int getPlate(){
        return plate;
    }
    public int getHit(){
        return hit;
    }
    public double getOnBase(){
        return onBase;
    }
}
```

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

        TopBatter[] batter = new TopBatter[12];

        batter[0] = new TopBatter("長野", 676, 171, 50, 10);
        batter[1] = new TopBatter("坂　", 668, 214, 43, 0);
        batter[2] = new TopBatter("大島", 625, 170, 40, 5);
        batter[3] = new TopBatter("山田", 667, 209, 81, 0);
        batter[4] = new TopBatter("堂林", 659, 172, 57, 36);
        batter[5] = new TopBatter("石川", 584, 153, 24, 38);
        batter[6] = new TopBatter("陽　", 662, 193, 74, 7);
        batter[7] = new TopBatter("聖澤", 577, 150, 62, 24);
        batter[8] = new TopBatter("中村", 662, 167, 50, 10);
        batter[9] = new TopBatter("金子", 643, 170, 48, 14);
        batter[10] = new TopBatter("荻野", 692, 206, 83, 8);
        batter[11] = new TopBatter("坂口", 622, 172, 56, 7);

        double maxOnBase = 0.00;
        String maxName = "";

        System.out.println("選手名　打席数　安打数　出塁率");

        for(int i=0; i<batter.length; i++){
            TopBatter player = batter[i];

            player.setOnBase();

            if(maxOnBase < player.getOnBase()){
                maxOnBase = player.getOnBase();
                maxName = player.getName();
            }
            System.out.printf(""%s      %3d     %3d     %.3f¥n",
                player.getName(), player.getPlate(),
                player.getHit(), player.getOnBase());
        }
        System.out.println();
        System.out.printf("出塁率が一番の選手は、%s 選手で、%.3f です。",
                maxName, maxOnBase);
    }
}
```

3.

(1) 「点」クラス

```
class MyPoint{
    private int x, y;    //x 座標、y 座標を記憶するフィールド
    // コンストラクタ …… 仮引数をフィールドに初期値設定
    public MyPoint(int x, int y){
        this.x = x;
        this.y = y;
    }
    //x 座標を返却するメソッド
    public int getX(){
        return x;
    }
    //y 座標を返却するメソッド
    public int getY(){
        return y;
    }
}
```

(2) 「線」クラス

```
class MyLine{
    private MyPoint p1, p2;      //Point 型のフィールド
    // コンストラクタ …… 引数をフィールドに初期値設定
    public MyLine(MyPoint p1, MyPoint p2){
        this.p1 = p1;
        this.p2 = p2;
    }
    // ２点間の距離を計算・返却 …… MyPoint クラスの getX()、getY() メソッド呼び出し
    public double getLen(){
        double len = Math.sqrt(
            Math.pow(p1.getX() - p2.getX(),2) +
            Math.pow(p1.getY() - p2.getY(),2));
        return len;
    }
}
```

(3) 「三角形」クラス

```
class MyTriangle{
    private MyLine ln1, ln2, ln3;    //MyLine 型のフィールド
    // コンストラクタ …… 引数をフィールドに初期値設定
    public MyTriangle(MyLine ln1, MyLine ln2, MyLine ln3){
        this.ln1 = ln1;
        this.ln2 = ln2;
        this.ln3 = ln3;
    }
    // ヘロンの公式により三角形の面積を計算・返却
    public double getArea(){
        //MyLine クラスの getLen() メソッドの呼び出し
        double a = ln1.getLen();
        double b = ln2.getLen();
        double c = ln3.getLen();

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

(4) テストドライバ

```
public class TriangleAreaTest{
    public static void main(String[] args){
        MyPoint[] pt = new MyPoint[3];  //MyPoint 型の配列定義
        MyLine[] ln = new MyLine[3];    //MyLine 型の配列定義

        // ３点の座標を引数として、MyPoint 型のインスタンスを生成
        pt[0] = new MyPoint(0,0);
        pt[1] = new MyPoint(3,2);
        pt[2] = new MyPoint(1,4);
        // ３点のインスタンスを引数として、MyLine 型のインスタンスを生成
        ln[0] = new MyLine(pt[0], pt[1]);
        ln[1] = new MyLine(pt[1], pt[2]);
        ln[2] = new MyLine(pt[2], pt[0]);
        // ３線のインスタンスを引数として、MyTriangle 型のインスタンスを生成
        MyTriangle tri = new MyTriangle(ln[0], ln[1], ln[2]);
        // 三角形のインスタンスの getArea() メソッドを呼び出して面積を表示
        System.out.printf(" ３点 A, B, C を頂点とする三角形の面積は %.2f です。",
                                tri.getArea());
    }
}
```

■第 8 章 §3 練習問題 (p.147)

1．(1) オブジェクト変数 this はその属するインスタンスの**非**静的メンバーにアクセスすることができる。
(2) Java のクラスには、複数のコンストラクタを定義することができる。これをコンストラクタのオーバー**ロード**と呼ぶ。
(3) 必要な**メソッド**だけを外部に公開し、外部から直接、**フィールド**にアクセスできないようにする考え方をカプセル化という。
(4) フィールド値の照会や更新のためのメソッドを**アクセサ**メソッドという。
(5) 仮引数のデータ型は同じだが、その並び順が異なるメソッドは多重定義でき**る**。
(6) Calendar クラスのコンストラクタは **protected** 修飾されており、他のパッケージの一般クラスから new 演算子でインスタンス化できない。
(7) Calendar クラスの getTimeInMillis() メソッドは、Calendar にセットされた時刻値を 1/1000 秒単位の **long** 型で返す。

2．(1) `Calendar cal = Calendar.getInstance();`
(2) `int month = cal.get(Calendar.MONTH) + 1;`
(3) `long millis = cal.getTimeInMillis();`

■第 8 章 §3 実習問題 (p.148)

1．

```
import java.util.Scanner;

public class DaysBetween{
    public static void main(String[] args){
        int year1, year2;
        int month1, month2;
        int day1, day2;
        int daysBetween;
        final long MIL_PER_DAY = 24*60*60*1000;
        Scanner scan = new Scanner(System.in);

        System.out.println("2 つの年月日間の日数を計算・表示します。");
        System.out.print(" 初めの年を入力してください。:");
        year1 = scan.nextInt();
        System.out.print(" 初めの月を入力してください。:");
        month1 = scan.nextInt();
        System.out.print(" 初めの日を入力してください。:");
        day1= scan.nextInt();

        // 例題 20 の MyDate クラスをそのまま再利用するので同一フォルダに入れておく
        MyDate dt1 = new MyDate(year1, month1, day1);
        if(dt1.getYear()==0 || dt1.getMonth()==0 || dt1.getDay()==0){
            System.out.println(" 年月日のいずれかの数値が範囲外です。");
            System.exit(1);
        }
        System.out.println();

        System.out.print(" 次の年を入力してください。:");
        year2 = scan.nextInt();
        System.out.print(" 次の月を入力してください。:");
        month2 = scan.nextInt();
        System.out.print(" 次の日を入力してください。:");
        day2 = scan.nextInt();

        MyDate dt2 = new MyDate(year2, month2, day2);
        if(dt2.getYear()==0 || dt2.getMonth()==0 || dt2.getDay()==0){
            System.out.println(" 年月日のいずれかの数値が範囲外です。");
            System.exit(1);
        }
```

（次ページへ続く）

```
        daysBetween =
             (int) ((dt2.getMillis() - dt1.getMillis()) / MIL_PER_DAY);
        System.out.println();
        System.out.printf("%d年%d月%d日と%d年%d月%d日の間の日数は%d日です。",
            year1, month1, day1, year2, month2, day2, daysBetween);
    }
}
```

2.

```
class FiveTriangle{
    private double base;
    private double height;
    private double a, b, c;

    //コンストラクタのオーバーロード①～⑤
    //①デフォルトコンストラクタ
    public Triangle(){}
    //②正三角形　高さは　底辺×√3/2　仮引数int型
    public Triangle(int a){
        this.base = a;
        this.height = a * Math.sqrt(3) / 2;
    }
    //③正三角形　高さは　底辺×√3/2　仮引数double型
    public Triangle(double a){
        this.base = a;
        this.height = a * Math.sqrt(3) / 2;
    }
    //④普通の三角形 底辺×高さ
    public Triangle(double b, double h){
        setSize(b, h);
    }
    //⑤３辺の長さ
    public Triangle(double a, double b, double c){
        setSize(a, b, c);
    }

    //底辺と高さのset()メソッド
    public void setSize(double b, double h) {
        base = b;
        height = h;
    }
    //３辺のset()メソッド
    public void setSize(double a, double b, double c){
        this.a = a;
        this.b = b;
        this.c = c;
    }

    //フィールド値の返却
    public double getBase(){
        return base;
    }
    public double getHeight(){
        return height;
    }
    public double getA(){
        return a;
    }
    public double getB(){
        return b;
    }
    public double getC(){
        return c;
    }
```

（次ページへ続く）

```
        // 底辺と高さによる面積計算
        public double getArea1(){
            return base * height / 2;
        }
        // 3辺の長さによる面積計算
        public double getArea2(){
            double s = (a + b + c) / 2;
            return Math.sqrt(s * (s-a) * (s-b) * (s-c));
        }
    }
```

■第 9 章 §1 練習問題 (p.156)

1． 誤っているもの (1) (3) (4) (5)

(1) 一般にサブクラスは、スーパークラスより多くの機能を持つ。
(3) サブクラスは 1 つのスーパークラスしか持つことができない。一方、スーパークラスは、複数のサブクラスで継承できる。
(4) 継承が想定されるスーパークラスのメンバーのアクセス修飾子は、一般に protected とする。
(5) 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には、super を用いる。

2． (3)

Parent と Child の 2 つのクラスで、それぞれ同名のフィールド x と y を定義し、それらに異なる初期値を設定している。そして、Child クラスのコンストラクタで、Child クラスの x フィールドと Parent クラスの y フィールドの値を変更している。
Child クラスの y フィールドには、初期値が設定されていないが、ローカル変数は明示的に初期値を設定しないとエラーになるのに対して、フィールドはその属するインスタンスが生成されたとき、整数型については 0、浮動小数点数型については 0.0、boolean 型については false、参照型については null が暗黙の内に設定されるので、明示的に初期値設定しなくてもエラーにならない。

■第 9 章 §1 実習問題 (p.157)

(1) 管理職報酬計算プログラム（Manager クラス）

```
class Manager extends Staff{
    private final int SECTION_RATE = 2250;
    private int sectionPoint;
    private int earnings;

    // コンストラクタ
    public Manager(String code, String name,
                int baseSalary, int careerPoint, int sectionPoint){
        // スーパークラス（staff クラス）のコンストラクタ呼び出し
        super(code, name, baseSalary, careerPoint);
        this.sectionPoint = sectionPoint;
    }
    public String getCode(){
        return code;
    }
    public String getName(){
        return name;
    }
    public int getSectionPoint(){
        return sectionPoint;
    }
    // 月例給与（一般社員給与＋部門別評価点×部門別評価基準金額）の照会
    public int getEarnings(){
        earnings= super.getEarnings() + (sectionPoint * SECTION_RATE);
        return earnings;
    }
}
```

(2) パートタイマー報酬計算プログラム（PartTimer クラス）

```
class PartTimer extends Staff{
    private double hour;
    private int wagePerHour;
    private int earnings;

    public PartTimer(String name,
              String code, double hour, int wagePerHour){
        super(name, code, 0, 0);
        this.hour = hour;
        this.wagePerHour = wagePerHour;
    }
    public String getCode(){
        return code;
    }
    public String getName(){
        return name;
    }
    public double getHour(){
        return hour;
    }
    public int getWagePerHour(){
        return wagePerHour;
    }
    public int getEarnings(){
        earnings =(int)(hour * wagePerHour);
        return earnings;
    }
}
```

■第 9 章 §2 練習問題 (p.165)

1．誤っているもの：(1)　(3)　(4)　(5)
2．　①a　②a　③c　④b

■第 9 章 §2 実習問題 (p.166)

1．

```
public class Sorter2Test{
    public static void main(String[] args){
        int arraySize = 50000;

        Timer timer = new Timer();
        // スーパークラスの配列に各ソートプログラムインスタンスを生成・格納
        Sorter[] sorter = {
            new BubbleSorter(arraySize),
            new SelectSorter(arraySize),
            new InsertSorter(arraySize)
        };

        for(int i=0; i<sorter.length; i++){
            sorter[i].setRandom();

            timer.start();          // 計測開始
            sorter[i].lineUp();     // 整列実行
            timer.stop();           // 計測終了

            // 所要時間を整数部 4 けた、小数部 2 けたで表示
            System.out.printf("所要時間 :%4.2f 秒 ¥n",timer.getTime());
            // 確認用表示メソッドを呼び出し
            sorter[i].showArray();
        }
    }
}
```

2.【Staff クラス】

```
class Staff extends Employee{
    protected int baseSalary;            // 基本給
    protected int careerPoint;           // 個人別評価点
    protected int earnings;              // 月例所得
    protected final int ID_RATE=1500;    // 個人別評価基準金額

    // コンストラクタ（名前、社員番号、基本給、個人別評価点）
    public Staff(String code,
                 String name, int baseSalary, int careerPoint){
        super(code, name);
        this.baseSalary = baseSalary;
        this.careerPoint = careerPoint;
    }
    public String getCode(){
        return code;
    }
    public String getName(){
        return name;
    }
    public int getBaseSalary(){
        return baseSalary;
    }
    public int getCareerPoint(){
        return careerPoint;
    }
    // 月例給与（基本給＋個人別評価点× ID_RATE）
    public int getEarnings(){
        earnings = baseSalary + (careerPoint * ID_RATE);
        return earnings;
    }
}
```

【Manager クラス】

```
class Manager extends Staff{
    private static final int SECTION_RATE = 2250;
    private int sectionPoint;
    private int earnings;

    // コンストラクタ
    public Manager(String code, String name, int baseSalary,
                           int careerPoint, int sectionPoint){
        super(code, name, baseSalary, careerPoint);
        this.sectionPoint = sectionPoint;
    }
    public String getCode(){
        return code;
    }
    public String getName(){
        return name;
    }
    public int getSectionPoint(){
        return sectionPoint;
    }
    // 月例給与（一般社員給与＋部門別評価点×部門別評価基準金額）
    public int getEarnings(){
        earnings= super.getEarnings() + (sectionPoint * SECTION_RATE);
        return earnings;
    }
}
```

【PartTimer クラス】

```
class PartTimer extends Employee{
    private double hour;
    private int wagePerHour;
    private int earnings;

    public PartTimer(String code,
              String name, double hour, int wagePerHour){
        super(code, name);
        this.hour = hour;
        this.wagePerHour = wagePerHour;
    }
    public String getCode(){
        return code;
    }
    public String getName(){
        return name;
    }
    public double getHour(){
        return hour;
    }
    public int getWagePerHour(){
        return wagePerHour;
    }
    public int getEarnings(){
        earnings =(int)(hour * wagePerHour);
        return earnings;
    }
}
```

■第 9 章 §3 練習問題 (p.175)

1． 誤っているもの： (3) (6) (7)

2． a イ　b イ　c ウ　d エ　e オ

■第 9 章 §3 実習問題 (p.178)

1．【プログラム 1】（変更なし）

```
public class AreaTest{
    public static void main(String args[]) {
        Figure[] figures = {
            new Triangle(2, 3, 3),
            new Rectangle(5, 8),
            new Square(5)};
        for (int i=0; i<figures.length; i++) {
            System.out.println(figures[i] +
                          "area = " + figures[i].getArea());
        }
    }
}
```

【プログラム 2】

```
interface  Figure {
   public abstract double getArea();
}
```

【プログラム３】

```
public class Triangle implements Figure{
    double la;
    double lb;
    double lc;
    public Triangle(double la, double lb, double lc) {
        this.la = la;
        this.lb = lb;
        this.lc = lc;
    }
    public String toString() {
        return "Triangle : sides = "
                 + la + ", " + lb + ", " + lc + " : ";
    }
    public double getArea() {
        double s = (la + lb + lc) / 2.0;
            return Math.sqrt(s * (s - la) * (s - lb) * (s - lc));
    }
}
```

【プログラム４】

```
public class Rectangle implements Figure{
   double height;
   double width;
   public Rectangle(double height, double width) {
      this.height = height;
      this.width = width;
   }
   public String toString() {
      return "Rectangle : height = " + height +
             ", width = " + width + " : ";
   }
   public double getArea() {
      return height * width;
   }
}
```

【プログラム５】（変更なし）

```
public class Square extends Rectangle{
   public Square(double width) {
       super(width, width);
   }
   public String toString() {
      return "Square : width = " + width + " : ";
   }
}
```

2.

```
//     Employee.java

interface Employee{
    String getName();
    String getCode();
    String toString();
}
```

```
//     Payment.java

abstract class Payment implements Employee{
    protected String code;
    protected String name;
    protected static final int ID_RATE = 1500;
    protected static final int SECTION_RATE = 2250;

    public Payment(String code, String name){
        this.code = code;
        this.name = name;
    }
    public String getCode(){
        return code;
    }
    public String getName(){
        return name;
    }
    public String toString(){
       return String.format("%s¥t%s¥t%s¥t%,d 円 ¥n",
              getCode(), getName(), getJobType(), getEarnings());
    }
    public abstract String getJobType();
    public abstract int getEarnings();
}
```

```
//     Staff.java

class Staff extends Payment{
    protected int careerPoint;
    protected int baseSalary;
    protected int earnings;
    private String jobType = " 一般社員 ";

    public Staff(String code,
                      String name, int baseSalary, int careerPoint){
        super(code, name);
        this.baseSalary = baseSalary;
        this.careerPoint = careerPoint;
    }
    public int getBaseSalary(){
        return baseSalary;
    }
    public int getCareerPoint(){
        return careerPoint;
    }
    public String getJobType(){
       return jobType;
    }
    // 月例給与（基本給＋個人別評価点× ID_RATE)
    public int getEarnings(){
        earnings = baseSalary + (careerPoint * ID_RATE);
        return earnings;
    }
}
```

```
//     Manager.java

class Manager extends Staff{
    private int sectionPoint;
    private int earnings;
       private String jobType = "管理職  ";

    public Manager(String code, String name, int baseSalary,
                              int careerPoint, int sectionPoint){
        super(code, name, baseSalary, careerPoint);
        this.sectionPoint = sectionPoint;
    }
    public int getSectionPoint(){
        return sectionPoint;
    }
    public String getJobType(){
       return jobType;
    }
    // 月例給与（一般社員給与＋部門別評価点×部門別評価基準金額）の照会
    public int getEarnings(){
        earnings= super.getEarnings() +
                  (sectionPoint * SECTION_RATE);
        return earnings;
    }
}
```

```
//     PartTimer.java

class PartTimer extends Payment{
    private double hour;
    private int wagePerHour;
    private int earnings;
       private String jobType = "ﾊﾟｰﾄﾀｲﾏｰ";

    public PartTimer(String code,
                        String name, double hour, int wagePerHour){
        super(code, name);
        this.hour = hour;
        this.wagePerHour = wagePerHour;
    }
    public double getHour(){
        return hour;
    }
    public int getWagePerHour(){
        return wagePerHour;
    }
    public String getJobType(){
       return jobType;
    }
    // 賃金（時給×勤務時間）
    public int getEarnings(){
        earnings =(int)(hour * wagePerHour);
        return earnings;
    }
}
```

■第 10 章 §1 練習問題 (p.188)

1．(1)　②　　(2)　①　　(3)　④

2．(1)　ArrayList<Integer> lst = new ArrayList<Integer>();
(2)　lst.add(j);
(3)　sum += j;
(4)　for(Integer divisor : lst)

■第 10 章 §1 実習問題 (p.189)

1．

```
import java.util.ArrayList;

public class SolarSystem{
    public static void main(String[] args){
        String[]  planets = {
            "Sun", "Mercury", "Venus", "Mars", "Jupiter",
            "Saturn", "Uranus", "Neptune", "Pluto"};

        ArrayList<String> solar = new ArrayList<String>();

        for(String p : planets)
            solar.add(p);

        solar.remove("Pluto");
        solar.add(solar.indexOf("Mars"), "Earth");

        System.out.println(solar.toString());
    }
}
```

2．

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

public class EigoTest{
    public static void main(String[] args){
        ArrayList<Integer> eigoTest = new ArrayList<Integer>();
        Scanner scan = new Scanner(System.in);
        int num = 2;              // 入力番号
        int tokuten = 0;          // 得点
        int top = 0;              // 最高点の仮の初期値
        int bottom = 100;         // 最低点の仮の初期値
        int total = 0;            // 得点合計
        double avarage;           // 平均点

        //1 番目の得点の入力
        System.out.println(" 英語の得点を入力してください。");
        System.out.print(" 番号 1:");
        tokuten = scan.nextInt();
        //2 番目以下の得点の記憶と入力
        while(tokuten>=0){
            eigoTest.add(tokuten);
            System.out.printf(" 番号 %d:", num++);
            tokuten = scan.nextInt();
        }
        // 最高点・最低点・合計の計算
        for(int tensu : eigoTest){
            if(top < tensu)
                top = tensu;
            if(bottom > tensu)
                bottom = tensu;
            total += tensu;
        }
        // 平均の計算
        avarage = (double)total/eigoTest.size();
```

（次ページへ続く）

```
        // 計算結果の表示
        System.out.println();
        System.out.printf("最高点：%d¥n", top);
        System.out.printf("最低点：%d¥n", bottom);
        System.out.printf("平均点：%2.2f¥n", avarage);
    }
}
```

3.

```
import java.util.ArrayList;

class BigCity{
    private String code;
    private String city;
    private String pref;
    private int year;
    private int ward;
    private int pop;

    public BigCity(String code, String city, String pref,
                        int year, int ward, int pop){
        this.code = code;
        this.city = city;
        this.pref = pref;
        this.year = year;
        this.ward = ward;
        this.pop = pop;
    }
    public String getCode(){
        return code;
    }
    public String getCity(){
        return city;
    }
    public String getPref(){
        return pref;
    }
    public Integer getYear(){
        return year;
    }
    public Integer getWard(){
        return ward;
    }
    public Integer getPop(){
        return pop;
    }
}

public class BigCityDriver{
    public static void main(String[] args){
        ArrayList<BigCity> bigCity = new ArrayList<BigCity>();

        bigCity.add(new BigCity("01100","札幌市　　","北海道　",1972,10,1906141));
        bigCity.add(new BigCity("04100","仙台市　　","宮城県　",1989,5,1057659));
        bigCity.add(new BigCity("11100","さいたま市","埼玉県　",2003,10,1233943));
        bigCity.add(new BigCity("14100","横浜市　　","神奈川県",1956,18,3697894));
        bigCity.add(new BigCity("14130","川崎市　　","神奈川県",1972,7,1437266));
        bigCity.add(new BigCity("23100","名古屋市　","愛知県　",1956,16,2261377));
        bigCity.add(new BigCity("26100","京都市　　","京都府　",1956,11,1470587));
        bigCity.add(new BigCity("27100","大阪市　　","大阪府　",1956,24,2670992));
        bigCity.add(new BigCity("28100","神戸市　　","兵庫県　",1956,9,1541596));
        bigCity.add(new BigCity("34100","広島市　　","広島県　",1980,8,1175275));
        bigCity.add(new BigCity("40130","福岡市　　","福岡県　",1972,7,1483052));

        for(BigCity bc : bigCity)
            System.out.printf("%5s  %-6s  %-6s  %4d  %2d  %,d¥n",
                bc.getCode(), bc.getCity(), bc.getPref(),
                bc.getYear(), bc.getWard(), bc.getPop());
    }
}
```

■第 10 章 §2 練習問題 (p.197)

1．(1) ウ　(2) イ　(3) エ　(4) ア

【確認用テストプログラム】

```
import java.util.LinkedList;
import java.util.HashSet;
import java.util.LinkedHashSet;
import java.util.TreeSet;
import java.util.Collection;
import java.util.Arrays;

public class SugakuTest{
    static Integer[] sugaku =
            {78,86,62,100,83,78,85,91,67,100,62,75};
    public static void main(String[] args){
        Collection<Integer> linkedList =
                    new LinkedList<Integer>();
        showElement(linkedList);

        Collection<Integer> hashSet =
                    new HashSet<Integer>();
        showElement(hashSet);

        Collection<Integer> linkedHashSet =
                    new LinkedHashSet<Integer>();
        showElement(linkedHashSet);

        Collection<Integer> treeSet =
                    new TreeSet<Integer>();
        showElement(treeSet);
    }
    static void showElement(Collection<Integer> col){
        col.addAll(Arrays.asList(sugaku));
        System.out.println(col.getClass().getName());
        System.out.println(col);
    }
}
```

2．(1) basil　(2) sage　(3) oregano　(4) mint　(5) laurel　(6) chicory

3．(1) `LinkedList<String> pairList = new LinkedList<String>();`
(2) `while((num % 2) != 0);`
(3) `right++;`
(4) `for(String prim : pairList)`

■第 10 章 §2 実習問題 (p.199)

1．(1)

```
import java.util.ArrayList;
import java.util.Arrays;

class SushiOfWorld01{
    public static void main(String[] args){
        String[] sushi = {"カリフォルニアロール", "牛タンロール",
                          "キャタピラーロール", "なるとロール",
                          "レインボーロール"};
        ArrayList<String>  newSushi
                = new ArrayList<String>(Arrays.asList(sushi));
        for(String kan : newSushi)
            System.out.println(kan);
    }
}
```

(2)

```
import java.util.ArrayList;
import java.util.Arrays;

class SushiOfWorld02{
    public static void main(String[] args){
        String[] sushi = {"カリフォルニアロール", "牛タンロール",
                          "キャタピラーロール", "なるとロール",
                          "レインボーロール"};
        ArrayList<String>  newSushi
                = new ArrayList<String>(Arrays.asList(sushi));

        newSushi.set(1, "モンキーロール");
        newSushi.set(3, "スパイダーロール");

        for(String kan : newSushi)
            System.out.println(kan);
    }
}
```

2.

```
import java.util.*;

class OceanBreeze{
    public static void main(String[] args){
        List<Integer> candy =                    //応募者
                    new ArrayList<Integer>();
        List<Integer> breezer =                  //当選者
                    new ArrayList<Integer>();
        List<Integer> newCandy =                 //補欠候補者
                    new ArrayList<Integer>();
        List<Integer> waitingList =              //補欠当選者
                    new ArrayList<Integer>();

        for(int i=1; i<=1831; i++)               //(1)
            candy.add(i);

        Collections.shuffle(candy);              //(2)
        breezer = candy.subList(0, 277);         //(3)
        newCandy = candy.subList(277,1831);      //当選者を除外したList
        Collections.shuffle(newCandy);           //(4)
        waitingList = newCandy.subList(0, 30); //(5)

        System.out.println("【当選者】");
        Collections.sort(breezer);
        int counter = 0;
        for(Integer br : breezer)
            System.out.printf(
                    ++counter%10==0? "%4d  ¥n" : "%4d  ", br);
        System.out.println();

        System.out.println("【補欠当選者】");
        counter = 0;
        for(Integer wl : waitingList)
            System.out.printf(
                    ++counter%10==0? "%4d  ¥n" : "%4d  ", wl);
    }
}
```

subList(fromIndex, toIndex) は、List インターフェースのメソッドで、特定の List から fromIndex（を含む）から toIndex（を含まない）の範囲のデータを切り出してその部分集合を作成する。List の Index は配列の添字と同様に 0 から始まる。したがって、この問題では List の 0 ～ 1830 の Index 領域に、要素として index+1 の応募者番号を記憶していることになる。補欠当選者の決定（newCandy）の subList() の toIndex が 1831 で、List の最終 Index1830 を超えているが、subList() の toIndex は、

最終 index+1 の数値まで指定できる（IndexOutOfBoundsException は発生しない）。
この問題では、以上を前提に、subList() のパラメータ、List の Index、その要素である応募者番号の関係をていねいに考えて、部分集合の切り出し範囲を指定する必要がある。

3．p.179 の実行用テストプログラム PaymentTest を次のように変更する。その他のプログラムは変更の必要はない。

```
//    PaymentListTest.java

import java.util.LinkedList;

public class PaymentListTest{
    public static void main(String[] args){
        LinkedList<Payment> pmt = new LinkedList<Payment>();

        // 一般社員（名前、社員番号、基本給、個人別評価点）
        pmt.add(new Staff("010-0234", "渋沢　航", 200000, 12));
        pmt.add(new Staff("010-0522", "矢継弓香", 200000, 9));
        pmt.add(new Staff("010-0883", "利根川渉", 200000, 8));
        pmt.add(new Staff("010-0521", "高見　崇", 200000, 10));

        // 管理職（名前、社員番号、基本給、個人別評価点、部門別評価点）
        pmt.add(new Manager("030-0255", "浅井忠保", 250000, 17, 9));
        pmt.add(new Manager("030-0392", "樽井怜治", 250000, 16, 11));

        // パートタイマー（名前、社員番号、勤務時間、時給）
        pmt.add(new PartTimer("000-0431", "鹿島紀子", 145.25, 1065));
        pmt.add(new PartTimer("000-0676", "財部きん", 138.50, 980));
        pmt.add(new PartTimer("000-1505", "菅野葉月", 125.50, 950));
        pmt.add(new PartTimer("000-0184", "西倉豊蔵", 140.75, 920));

        for(Payment emp : pmt)
            System.out.print(emp);
    }
}
```

■第 11 章 §1 練習問題 (p.208)

1．(2)

2．(1)　return 1;
　(2)　return line + countStars(line-1);
　(3)　countStars(number);

3．

	誤		正
02	import java.util.File;	→	import java.io.File;
06	Calendar cal = new Calendar(getInstance());	→	Calendar cal = Calendar.getInstance();
07	File myDir = new myDir("E:¥¥myJavaBook");	→	File myDir = new File("E:¥¥MyJavaBook");
09	if(myDir.exists()){	→	if(!myDir.exists()){
20	String[] pathList = myDir.listFiles();	→	String[] pathList = myDir.list();

■第 11 章 §1 実習問題 (p.209)

1.

```
import java.util.Scanner;

public class RecursiveTest{
    public static void main(String[] args){
        int num;
        Scanner scan = new Scanner(System.in);
        System.out.print(" 総和と階乗を求める数を入力してください。:");
        num = scan.nextInt();
        System.out.println();

        System.out.printf(
            " 1 から %d までの総和は %d です。¥n", num, sumTotal(num));
        System.out.printf(
            "%d の階乗は %, d です。", num, factorial(num));
    }
    // 総和
    static int sumTotal(int num){
        if(num == 1)
            return 1;
        else
            return (num + sumTotal(num-1));
    }
    // 階乗
    static long factorial(int num){
        if(num == 1)
            return 1;
        else
            return (num * factorial(num-1));
    }
}
```

2.

```
import java.util.Scanner;

class ToggledCollatz{
    static int counter = 0;
    public static void main(String[] arags){
        int input;
        Scanner scan = new Scanner(System.in);

        do{
            System.out.print("2 以上の整数を入力してください。: ");
            input = scan.nextInt();
        }while(input<2);
        counter = myToggle(input);
        System.out.printf("¥n%d 回の計算で 1 になりました。", counter);
    }
    static int myToggle(int num){
        if(num==1){
            System.out.printf("%4d", num);
            return counter;
        }else{
            counter++;
            System.out.printf(
                counter%10 == 0? "%4d¥n" : "%4d ", num);
            if(num%2 == 0)
                return myToggle(num/2);
            else
                return myToggle(num*3+1);
        }
    }
}
```

3.

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

public class Dir2ArrayList{
    static int fileCounter = 0;
    static long byteCounter = 0;
    static ArrayList<MyFiles> myFileList = new ArrayList<MyFiles>();

    public static void main(String args[]){
        // ディレクトリ名の入力処理
        Scanner scan = new Scanner(System.in);
        System.out.println(" ディレクトリ名を入力してください。");
        String inputDir = scan.nextLine();
        // 入力ディレクトリのファイルオブジェクトを生成
        File dirName = new File(inputDir);
        // ファイルが見つかれば再帰メソッド呼び出し
        if(dirName.exists())
            listUpPath(dirName);
        else{
            System.out.printf("%s が見つかりません。", dirName);
            System.exit(1);
        }
        // ファイルリストを表示
        for(MyFiles mf : myFileList)
            System.out.printf("%s", mf);
        System.out.printf(" このディレクトリの総ファイル数：%, d¥n",
                                                    fileCounter);
        System.out.printf(" このディレクトリの総バイト数：%, d¥n",
                                                   byteCounter) ;
    }
    static void listUpPath(File dirName){
        String[] pathList = dirName.list();

        for(int i=0; i<pathList.length; i++){
            String path = pathList[i];

            // ディレクトリのファイルオブジェクトを生成
            File myDir = new File(dirName, path);
            // ディレクトリなら自分自身を呼び出し、ファイルなら arrayList に記録
            if (myDir.isDirectory()){
                listUpPath(myDir);
            }else{
                myFileList.add(
                     new MyFiles(myDir.getName(), myDir.length(),
                                                 myDir.getParent()));
            fileCounter++;
            byteCounter += myDir.length();
            }
        }
    }
}
```

■第 11 章 §2 練習問題 (p.219)

1. (1) × finally ブロックは例外の有無にかかわらず必ず実行される。
   (2) × 「キー」と「値」がセットで昇順に記憶されるデータ構造は、TreeMap である。
   (3) × finally ブロックは省略できる。
   (4) ○
   (5) × 重大で一般的な例外処理で対応できない例外は、java では Error という。
   (6) × catch ブロックは複数の例外を捕捉できる。
   (7) × 「キー」は重複できないが、「値」は重複できる。
   (8) ○

2. (1) Throwable (2) Exception
(3) Error (4) RuntimeException
(5) IOException

3. (1) try (2) Scanner
(3) hasNextLine() (4) FileNotFoundException

■第 11 章 §2 実習問題 (p.220)

1.
```
import java.io.*;
import java.util.*;

public class Greeting{
    Scanner lang, greet;
    Map<String, String> myMap = new TreeMap<String, String>();

    public void readFile(String file1, String file2){
        try{
            lang = new Scanner(new File(file1));
            greet = new Scanner(new File(file2));
        }
        catch(FileNotFoundException e){
            System.out.println("入力ファイルが見あたりません。");
            System.exit(1);
        }
        while(lang.hasNextLine() && greet.hasNextLine())
                myMap.put(lang.nextLine(), greet.nextLine());
    }
    public void showElement(){
        System.out.println(myMap.getClass());
        System.out.println();
        for(String element : myMap.keySet()){
            System.out.printf(
                  "%s:%s¥n", element, myMap.get(element));
        }
    }

    public static void main(String[] args){
        Greeting grt = new Greeting();
        grt.readFile("language.txt","greeting.txt");
        grt.showElement();
    }
}
```

2.
```
import java.util.TreeMap;

public class MapOfDivisor{
    public static void main(String[] args){
        TreeMap<Integer, Integer> divisorMap
            = new TreeMap<Integer, Integer>();
        for(int i=100; i<=200; i++){
            int sum = 0;
            for(int j=1; j<=i/2; j++){
                if(i%j == 0)
                    sum += j;
            }
            divisorMap.put(i, sum);
        }
        for(Integer num : divisorMap.keySet())
            System.out.printf(
                 "%3d : %5d¥n", num, divisorMap.get(num));
    }
}
```

3.

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

class Kana2Roman{
    HashMap<Character, String> kanaMap =
                        new HashMap<Character, String>();
    Scanner myFile, myString;

    public void readKanaTable(){
        String line;
        String[] kanaRoman;

        try{
            myFile = new Scanner(new File("roman.txt"));
        }
        catch(FileNotFoundException e){
            System.out.println("ローマ字ファイルが見つかりません。");
            System.exit(1);
        }
        try{
            while(myFile.hasNextLine()){
                line = myFile.nextLine();
                kanaRoman = line.split(",");
                kanaMap.put(kanaRoman[0].charAt(0), kanaRoman[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{
            myFile.close();
        }
    }
    public void stringConverter(){
        String inputStr;
        char inputChar;
        StringBuilder sb = new StringBuilder();

        System.out.println("和文をひらがなで入力してください。");
        myString = new Scanner(System.in);
        inputStr = myString.nextLine();
        for(int i=0; i<inputStr.length(); i++){
            inputChar = inputStr.charAt(i);
            if(inputChar == ' ' || inputChar == '　')
                sb.append(" ");
            else if(kanaMap.get(inputChar) == null)
                sb.append("?");
            else
                sb.append(kanaMap.get(inputChar));
        }
        sb.append("¥n");

        System.out.println("変換文");
        System.out.print(sb.toString());
    }
}
public class Kana2RomanTest{
    public static void main(String[] args){
        Kana2Roman k2r = new Kana2Roman();
        k2r.readKanaTable();
        k2r.stringConverter();
    }
}
```

■第 11 章 §3 練習問題 (p.230)

1.（解答例）

① (java.lang.)ArithemeticException
非検査例外
ゼロで除算するなど算術計算で例外的条件が生じたとき発生する例外。

② (java.lang.)IllegalArgumentException
非検査例外
不正または不適切な引数をメソッドに渡したときに発生する例外

③ (java.lang.)IndexOutOfBoundsException
非検査例外
インデックス（個々のデータを示す指標）を持つデータ集合（配列、コレクション、文字列など）でそのデータ集合の範囲外のインデックスを指示したときに発生する例外。

④ (java.lang.)ArrayIndexOutOfBoundsException
非検査例外
マイナスの数値または要素数を超える不正なインデックスを使って配列がアクセスされたとき発生する例外。

⑤ (java.util.)NoSuchElementException
非検査例外
データ集合に指定した型のデータがないとき発生する例外。

⑥ (java.lang.)NumberFormatException
非検査例外
非数字を数値に変換しようとするとき発生する例外。

⑦ (java.io.)FileNotFoundException
検査例外
指定されたパス名のファイルが開けなかったときに発生する例外。

⑧ (java.io.)EOFException
検査例外
入力の途中で、予想外の「ファイル終了」（EOF:End Of File）に達したときに発生する例外。

2.
(1) d2f.getFiles(dirName)
(2) new Formatter(new File(outFile))
(3) fileOut.format
(4) IllegalFormatException e
(5) new File(f, str)
(6) getFiles(file)

■第 11 章 §3 実習問題 (p.231)

1.

```
import java.util.Scanner;
import java.util.Formatter;
import java.util.TreeMap;
import java.io.File;
import java.io.FileNotFoundException;
import java.util.IllegalFormatException;
import java.util.NoSuchElementException;
```

（次ページへ続く）

```
class WordPile02{
    Scanner input;
    Formatter output;
    String inputFile, outputFile;
    int wordCounter = 0;
    TreeMap<String, Integer> wordMap
                = new TreeMap<String, Integer>();

    public WordPile02(String inputFile, String outputFile){
        this.inputFile = inputFile;
        this.outputFile = outputFile;
    }
    public void openFile() throws FileNotFoundException{
        input = new Scanner(new File(inputFile));
        output = new Formatter(new File(outputFile));
    }
    public void readFile() throws NoSuchElementException{
        String sequence;         //1 行の文字列
        String[] word;           //1 行ごとの単語の記憶用配列
        int frequency = 0;       // 単語の出現頻度

        while(input.hasNextLine()){
            sequence = input.nextLine().toLowerCase();
            word = sequence.split(" ");
            wordCounter += word.length;
            for(String we : word){
                if(wordMap.get(we) == null)
                    wordMap.put(we, 1);
                else{
                    frequency = wordMap.get(we);
                    wordMap.put(we, ++frequency);
                }
            }
        }
    }
    public void writeFile() throws IllegalFormatException{
        for(String we : wordMap.keySet())
            output.format("%-12s : %2d¥n", we, wordMap.get(we));
        output.format(" 単語総数は %d です。", wordCounter);
        // ファイルを close して、データをバッファからファイルへ転送
        output.close();
    }
}

public class WordPile02Test{
    public static void main(String[] args){
        WordPile02 wp = new WordPile02(
                        "DannyBoy.txt", "DannyBoyPile.txt");
        try{
            wp.openFile();
        }catch(FileNotFoundException e){
            System.out.println(" 入力ファイルが見あたりません。");
            System.exit(1);
        }
        try{
            wp.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{
            wp.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);
        }
    }
}
```

2．　(1)

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

//     PaymentFileIO.java

class PaymentFileIO{
    private Scanner fileIn;
    private Formatter fileOut;
    LinkedList<Payment> pmt = new LinkedList<Payment>();
    // ファイルオープン
    public void openFile() throws
                FileNotFoundException, IOException{
        fileIn = new Scanner(new File("DreamWorks.csv"));
        fileOut = new Formatter("DreamOut.txt");
    }
    // ファイル入力
    public void readFile() throws
                NoSuchElementException, IOException{
        String personData;
        String[] item;
        String jobCode;
        while(fileIn.hasNextLine()){
            personData = fileIn.nextLine();
            item = personData.split(",");
            jobCode = item[0].substring(0,3); // 職種コードを読み取る
            // 一般社員（名前、社員番号、基本給、個人別評価点）
            if(jobCode.equals("010"))
                pmt.add(new Staff(item[0], item[1],
                    Integer.parseInt(item[2]),
                    Integer.parseInt(item[3])));
            // 管理職（名前、社員番号、基本給、個人別評価点、部門別評価点）
            else if(jobCode.equals("030"))
                pmt.add(new Manager(item[0], item[1],
                    Integer.parseInt(item[2]),
                    Integer.parseInt(item[3]),
                    Integer.parseInt(item[4])));
            // パートタイマー（名前、社員番号、勤務時間、時給）
            else
                pmt.add(new PartTimer(item[0], item[1],
                    Double.parseDouble(item[2]),
                    Integer.parseInt(item[3])));
        }
    }
    // ファイル出力
    public void writeFile() throws
                IllegalFormatException, IOException{
       for(Payment emp : pmt)
           fileOut.format("%s", emp);
       fileOut.close();
    }
}
```

```
//     PaymentFileIO_Test.java

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

public class PaymentFileIO_Test{
    public static void main(String[] args){
        PaymentFileIO pfio = new PaymentFileIO();

        try{
            pfio.openFile();
        }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);
        }
```

（次ページへ続く）

```
        try{
            pfio.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);
        }
        //pfo.show();

        try{
            pfio.writeFile();
        }catch(IllegalFormatException e){
            System.out.println("出力ファイの形式が不正です。");
            System.exit(1);
        }catch(IOException e){
            System.out.println("入出力例外が発生しました。");
            System.exit(1);
        }
    }
}
```

（注）コンパイル、実行には同一フォルダ内に次のファイルが必要（第9章第3節実習問題2で作成）。

Employee.java （インターフェース）
Payment.java （抽象クラス）
Staff.java
Manager.java
PartTimer.java
DreamWorks.csv （入力ファイル）

(2)

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

//   PaymentFileSortTest.java

public class PaymentFileSortTest{
    public static void main(String[] args){
        PaymentFileSort pfs = new PaymentFileSort();
        try{
            pfs.openFile();
        }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{
            pfs.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);
        }
        //pfs.show();  確認用
        try{
            pfs.writeFile();
        }catch(IllegalFormatException e){
            System.out.println("出力ファイの形式が不正です。");
            System.exit(1);
        }catch(IOException e){
            System.out.println("入出力例外が発生しました。");
            System.exit(1);
        }
    }
}
```

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

//    PaymentFileSort クラス

class PaymentFileSort{
    private Scanner fileIn;
    private Formatter fileOut;
    LinkedList<Payment> pmt = new LinkedList<Payment>();

    public void openFile() throws
                FileNotFoundException, IOException{
        fileIn = new Scanner(new File("DreamWorks.csv"));
        fileOut = new Formatter("DreamOut.txt");
    }
    public void readFile() throws
                NoSuchElementException, IOException{
        String personData;
        String[] item;
        String jobCode;
        while(fileIn.hasNextLine()){
            personData = fileIn.nextLine();
            item = personData.split(",");
            jobCode = item[0].substring(0,3);
            // 一般社員（名前、社員番号、基本給、個人別評価点）
            if(jobCode.equals("010"))
                pmt.add(new Staff(item[0], item[1],
                    Integer.parseInt(item[2]),
                    Integer.parseInt(item[3])));
            // 管理職（名前、社員番号、基本給、個人別評価点、部門別評価点）
            else if(jobCode.equals("030"))
                pmt.add(new Manager(item[0], item[1],
                    Integer.parseInt(item[2]),
                    Integer.parseInt(item[3]),
                    Integer.parseInt(item[4])));
            // パートタイマー（名前、社員番号、勤務時間、時給）
            else
                pmt.add(new PartTimer(item[0], item[1],
                    Double.parseDouble(item[2]),
                    Integer.parseInt(item[3])));
        }
        //LinkedList 中のデータをコード順に整列
        Collections.sort(pmt, new SortByCode());
    }
    /* 確認用
    public void show(){
        for(Payment pm : pmt)
            System.out.print(pm);
    }
    */
    public void writeFile()
                throws IllegalFormatException, IOException{
        for(Payment pm : pmt)
            fileOut.format("%s", pm);
        fileOut.close();
    }
}


//    SortByCode クラス    整列用クラス（コード順）

class SortByCode implements Comparator<Payment>{
    public int compare(Payment p1, Payment p2){
        return p1.getCode().compareTo(p2.getCode());
    }
}
```

3.

(1)

```
//    TowerList クラス

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

class TowerList{
    Scanner scan;
    Formatter output;
    ArrayList<Tower> towerList = new ArrayList<Tower>();

    public void openFile(String inFile, String outFile){
        try{
            scan = new Scanner(new File(inFile));
            output = new Formatter(new File(outFile));
        }
        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 towerLine;
        String[] towerItem;
        String name;
        int height;
        String city;
        String nation;
        int year;

        try{
            while(scan.hasNextLine()){
                towerLine = scan.nextLine();
                towerItem = towerLine.split("¥t");
                name = towerItem[0];
                height = Integer.parseInt(towerItem[1]);
                city = towerItem[2];
                nation = towerItem[3];
                year = Integer.parseInt(towerItem[4]);
                towerList.add(new Tower(name,
                                height, city, nation, year));
            }
        }
        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();
        }
        Collections.sort(towerList, new SortByHeight());
    }
    public void showFile(){
        for(Tower tw : towerList){
            System.out.printf(
                " 塔　名：%s¥n 高　さ：%dm¥n 所在地：%s¥n 国　名：%s¥n 竣工年：%d 年 ¥n¥n",
                tw.getName(), tw.getHeight(), tw.getCity(),
                tw.getNation(), tw.getYear());
        }
    }
    public void writeFile(){
        try{
            for(Tower tw : towerList)
                output.format(
                " 塔　名：%s¥n 高　さ：%dm¥n 所在地：%s¥n 国　名：%s¥n 竣工年：%d 年 ¥n¥n",
                tw.getName(), tw.getHeight(), tw.getCity(),
                tw.getNation(), tw.getYear());
        }catch(IllegalFormatException e){
            System.out.println(" 出力データ形式が不正です。");
        }finally{
            output.close();
        }
    }
}
```

```
//    SortByHeight クラス

import java.util.Comparator;

class SortByHeight implements Comparator<Tower>{
    public int compare(Tower t1, Tower t2){
        return t2.getHeight().compareTo(t1.getHeight());
    }
}
```

```
//     Tower クラス

class Tower{
    String name;
    int height;
    String city;
    String nation;
    int year;
    public Tower(String name,
              int height, String city, String nation, int year){
        this.name = name;
        this.height = height;
        this.city = city;
        this.nation = nation;
        this.year = year;
    }
    public String getName(){
       return name;
    }
    public Integer getHeight(){
       return height;
    }
    public String getCity(){
       return city;
    }
    public String getNation(){
       return nation;
    }
    public Integer getYear(){
       return year;
    }
}
```

```
//     TowerListTest クラス

public class TowerListTest{
    public static void main(String[] args){
        TowerList tl = new TowerList();
        String inFile = "世界の塔 tab. txt";
        String outFile = "世界の塔 out. txt";
        tl.openFile(inFile, outFile);
        tl.readFile();
        tl.showFile();
        tl.writeFile();
    }
}
```

(2)

（注）Tower クラスと SortByHeight クラスは (1) と同じ。テストプログム TowerSetTest クラスは、上記 (1) の TowerListTest クラスのクラス名を TowerSetTest に変更するとともに、TowerList オブジェクト生成部分を次のように変更して作成する。

`TowerList tl = new TowerList();` → `TowerSet ts = new TowerSet();`

`tl.openFile(inFile, outFile);` → `ts.openFile(inFile, outFile);`

以下、`tl.readFile()`、`tl.showFile()`、`tl.writeFile()` も同じ。

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

//    TowerSet クラス

class TowerSet{
    Scanner scan;
    Formatter output;
    TreeSet<Tower> towerSet =
            new TreeSet<Tower>(new SortByHeight());
    public void openFile(String inFile, String outFile){
        try{
            scan = new Scanner(new File(inFile));
            output = new Formatter(new File(outFile));
        }
        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 towerLine;
        String[] towerItem;
        String name;
        int height;
        String city;
        String nation;
        int year;

        try{
            while(scan.hasNextLine()){
                towerLine = scan.nextLine();
                towerItem = towerLine.split("\t");
                name = towerItem[0];
                height = Integer.parseInt(towerItem[1]);
                city = towerItem[2];
                nation = towerItem[3];
                year = Integer.parseInt(towerItem[4]);
                towerSet.add(new Tower(name, height, city, nation, year));
              }
        }
        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(Tower tw : towerSet){
            System.out.printf(
                " 塔　名：%s¥n 高　さ：%dm¥n 所在地：%s¥n 国　名：%s¥n 竣工年 :%d 年 ¥n¥n",
                tw.getName(), tw.getHeight(), tw.getCity(),
                tw.getNation(), tw.getYear());
        }
    }
    public void writeFile(){
        try{
            for(Tower tw : towerSet)
                output.format(
                " 塔　名：%s¥n 高　さ：%dm¥n 所在地：%s¥n 国　名：%s¥n 竣工年 :%d 年 ¥n¥n",
                tw.getName(), tw.getHeight(), tw.getCity(),
                tw.getNation(), tw.getYear());
        }catch(IllegalFormatException e){
            System.out.println(" 出力データ形式が不正です。");
        }finally{
            output.close();
        }
    }
}
```

(3)

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

//TowerMenu クラス

class TowerMenu{
    Scanner scan;
    Formatter output;
    TreeSet<Tower> towerSet;

    public void openFile(String inFile, String outFile){
        try{
            scan = new Scanner(new File(inFile));
            output = new Formatter(new File(outFile));
        }
        catch(FileNotFoundException e){
            System.out.println(" ファイルが見あたりません。");
            System.exit(1);
        }
    }
    public void readFile(int choice){
        String towerLine;
        String[] towerItem;
        String name;
        int height;
        String city;
        String nation;
        int year;

        switch(choice){
            case 1:
                towerSet = new TreeSet<Tower>(new SortByHeightUp());
                break;
            case 2:
                towerSet = new TreeSet<Tower>(new SortByHeightDw());
                break;
            case 3:
                towerSet = new TreeSet<Tower>(new SortByYearUp());
                break;
            case 4:
                towerSet = new TreeSet<Tower>(new SortByYearDw());
                break;
        }
        try{
            while(scan.hasNextLine()){
                towerLine = scan.nextLine();
                towerItem = towerLine.split("¥t");
                name = towerItem[0];
                height = Integer.parseInt(towerItem[1]);
                city = towerItem[2];
                nation = towerItem[3];
                year = Integer.parseInt(towerItem[4]);
                towerSet.add(new Tower(name,
                                       height, city, nation, year));
            }
        }
        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(Tower tw : towerSet){
            System.out.printf(
                " 塔　名：%s¥n 高　さ：%dm¥n 所在地：%s¥n 国　名：%s¥n 竣工年 :%d 年 ¥n¥n",
                tw.getName(), tw.getHeight(), tw.getCity(),
                tw.getNation(), tw.getYear());
        }
    }
```

（次ページへ続く）

```
    public void writeFile(){
        try{
            for(Tower tw : towerSet)
                output.format(
                "塔　名：%s¥n高　さ：%dm¥n所在地：%s¥n国　名：%s¥n竣工年：%d年¥n¥n",
                tw.getName(), tw.getHeight(), tw.getCity(),
                tw.getNation(), tw.getYear());
        }catch(IllegalFormatException e){
            System.out.println("出力データ形式が不正です。");
        }finally{
            output.close();
        }
    }
}
//SortByHeightDwクラス … 高さ（降順）
class SortByHeightDw implements Comparator<Tower>{
    public int compare(Tower t1, Tower t2){
        return t2.getHeight().compareTo(t1.getHeight());
    }
}
//SortByHeightUpクラス … 高さ（昇順）
class SortByHeightUp implements Comparator<Tower>{
    public int compare(Tower t1, Tower t2){
        return t1.getHeight().compareTo(t2.getHeight());
    }
}
//SortByYearDWクラス … 竣工年（降順）
class SortByYearDw implements Comparator<Tower>{
    public int compare(Tower t1, Tower t2){
        return t2.getYear().compareTo(t1.getYear());
    }
}
//SortByYearUpクラス … 竣工年（昇順）
class SortByYearUp implements Comparator<Tower>{
    public int compare(Tower t1, Tower t2){
        return t1.getYear().compareTo(t2.getYear());
    }
}
```

```
import java.util.Scanner;

//TowerMenuTestクラス

public class TowerMenuTest{
    public static void main(String[] args){
        TowerMenu tm = new TowerMenu();
        String inFile = "世界の塔tab.txt";
        String outFile = "世界の塔out.txt";
        Scanner scan = new Scanner(System.in);
        int choice;

        System.out.print(
            " 1. 高さ（昇順）¥n 2. 高さ（降順）¥n 3. 竣工年（昇順）¥n 4. 竣工年（降順）¥n");
        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();
        }while(choice<0 || choice>5);

        tm.openFile(inFile, outFile);
        tm.readFile(choice);
        tm.showFile();
        tm.writeFile();
    }
}
```