
Javaバイブルシリーズ Javaの教科書

別冊

確認問題解答例

演習問題解答例

プログラミング総合演習プログラム例

株式会社 **SCC**

A1 確認問題解答例

ここに掲載する解答例には、別解が存在することがあります。特に、プログラム例においては、その数が計り知れません。解答例以外の答えを見つけ出すことで、プログラミング能力はさらに向上することでしょう。

ソースコードの穴埋め問題の解答では、ソースコード全体ではなく、穴埋め部分の周辺のためのソースコードを抜粋して掲載している場合があります。また理解を助けるために設問中にはないコメントを追加したところもあります。ご注意ください。

第1章

●確認問題1.1

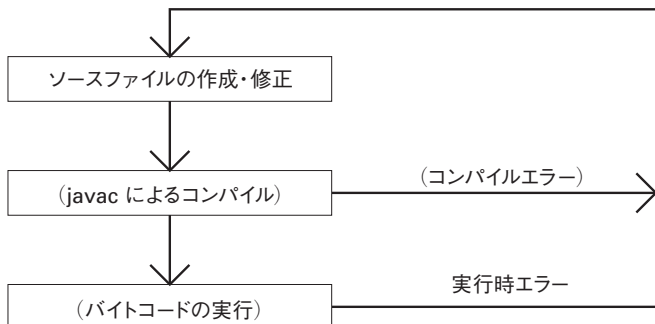
- | | |
|-------------------------------------|----------|
| ① Sun Microsystems 社 (サンマイクロシステムズ社) | |
| ② プラットフォーム | ③ バイトコード |
| ④ ガベージコレクション機能 | ⑤ 例外処理機能 |

●確認問題1.2

- | | |
|----------------------|--------------------------|
| ① Java 2 SDK または JDK | ② Java VM、Java 仮想マシン、JVM |
| ③ ネイティブコード | ④ JIT コンパイラ |

●確認問題1.3

- | | |
|---------|---------|
| ① .java | ② javac |
| ③ java | ④ デバッグ |
| ⑤ | |



第2章

●確認問題 2.1

- | | |
|----------------------|--|
| ① <code>/* */</code> | ② <code>/* */</code> と <code>//</code> |
| ③ <code>//</code> | ④ <code>/** */</code> |
| ⑤ インデント | ⑥ <code>main</code> |

●確認問題 2.2

- | | | | |
|-------|------|------|-----|
| ① 6 | ② 18 | ③ -1 | ④ 8 |
| ⑤ 99 | ⑥ 10 | ⑦ 5 | ⑧ 0 |
| ⑨ -12 | ⑩ 0 | | |

●確認問題 2.3

プログラムの実行結果

```
>java ExeCalc
127 + 128 = 255
100 - 68 = 32
16 * 16 = 256
10 / 100 = 0
2005 % 4 = 1

>
```

第3章

●確認問題 3.1

```
public class VariableExe {
    public static void main(String[] args) {
        int num; // ① int 型変数numの宣言

        num = 100; // ② 変数numに100を代入
        System.out.println( num ); // ③ 変数numの値を表示
    }
}
```

●確認問題 3.2

変数名として使用できるもの

- ② \$int (使用可能であるが、1文字目の記号(\$と_)は避けるべき)
- ③ number
- ④ Address (使用可能であるが、変数名は1文字目を小文字にする習慣がある)
- ⑥ mailAddress
- ⑦ Class (使用可能であるが、予約語のclassと誤解を招くので使用すべきでない)
- ⑧ java (使用可能であるが、使用された例を見たことがない)
- ⑩ longLongLongLongLongLongLongName (長さに制限はないが、限度はあると思う)

変数名として使用できないもの

- ① int (予約語であるため使用できない)
- ⑤ mail-Address (記号のハイフン(-)が使用できない)
- ⑨ red&green&blue (記号のアンパサンド(&)が使用できない)

●確認問題 3.3

- ① int 型 ② long 型 ③ double 型 ④ char 型
- ⑤ char 型 ⑥ String 型 ⑦ String 型 ⑧ boolean 型

第4章

●確認問題 4.1

(順次 または シーケンス) (選択 または 分岐) (繰り返し または ループ)

●確認問題 4.2

- ① トレース ② カウントアップ(インクリメント)
- ③ コードブロック(ブロック)
- ④ true ⑤ false

●確認問題 4.3

- 問題1 ① true ② false ③ true
- ④ true ⑤ false ⑥ false
- 問題2 (5)回表示される

●確認問題 4.4

命令	i	sum	説明
①、②	不定	不定	変数の宣言
③、④	1	0	変数の初期化
⑤	↓	↓	whileの条件式の評価 i <= 10でtrue ⑥へ進む
⑥	↓	1	sum ← sum(0) + i(1)
⑦	2	↓	i ← i + 1 ⑤へ戻る
⑤	↓	↓	条件式の評価でtrue ⑥へ進む
⑥	↓	3	sum ← sum(1) + i(2)
⑦	3	↓	i ← i + 1 ⑤へ戻る
	↓	6	
	4	↓	
	↓	10	
	5	↓	
	↓	15	
	6	↓	
	↓	21	
	7	↓	(中略)
	↓	28	
	8	↓	
	↓	36	
	9	↓	
	↓	45	
	10	↓	
⑤	↓	↓	whileの条件式の評価 i <= 10でtrue ⑥へ進む
⑥	↓	55	sum ← sum(45) + i(10)
⑦	11	↓	i ← i + 1 ⑤へ戻る(これで、iの値が11になったことに注意)
⑤	↓	↓	条件式の評価 i <= 10で false したがって、⑧へ進む
⑧	↓	↓	1から10までの和、「55」を表示する

これでトレースは終了です。このトレースでは途中のトレースを簡略化しています。ただし、繰り返し処理の終わるタイミングでは、丁寧なトレースを行います。これは、ループ型プログラムの終了のタイミングがもっとも重要になるためです。トレースでは、繰り返し処理のパターンを早くつかみ、変数の変化の法則を見つけ出すことが大切です。

● 確認問題 4.5

問題 1 実行結果

```
>java Trace1
iの値は4
numの値は16
>
```

問題 2

- ① 式の終わりがカンマ(,)になっていた。セミコロン(;)が正しい
- while 文に入る前、変数sumの初期化を行う必要がある
- ⑦の前にコードブロックの } (中カッコ閉じ)を忘れている

プログラム (Debug1.java) 修正例

```
public class Debug1 {
    public static void main(String[] args) {
        int i;                                // ① 式の終わりがカンマ(,)になっていた
        int sum;                                // ②

        sum = 0;                                // sumの初期化を忘れている
        i = 1;                                    // ③
        while (i <= 10) {                        // ④
            sum += i;                            // ⑤
            i++;                                // ⑥
        }                                        // コードブロックの中カッコ閉じ(})を忘れている
        System.out.println(sum); // ⑦
    }
}
```

問題 3

正しい結果を得られない理由

- 100 が加算されていないため正しい結果が求められない
- ひとつの解決策は、while 文の条件式を (i <= 100) にする

コーディングし直したプログラム例

```
public class Debug2 {
    public static void main(String[] args) {
        int i;
        int sum;

        sum = 0;
        i = 1;
        while (i <= 100) {                // 条件式「<」を「<=」に修正
            sum += i;
            i = i + 1;
        }
        System.out.println(sum);
    }
}
```



インデントしないコーディングや1行にいくつも文があるプログラムは、大変読みづらくなります。コーディングするときは、必ずインデントをして可読性を高めましょう。

●確認問題 4.6

- ① キーボードからデータを入力するときには、`readLine` メソッドを使う
- ② 数字(文字)列を整数に変換するには、`Integer` クラスの `parseInt` メソッドを使う
- ③ 入力データが空だったときには、受け取り側の `String` クラスの変数に `null` が設定される

●確認問題 4.7

問題1 (5回)表示される

問題2 for文のコーディング例

```
for ( i = 5; 0 <= i; i-- ) {
    System.out.println(i);
}
```

問題3 実行結果

```
>java Trace2
iの値は10
numの値は1024

>
```

問題4 for文と九九の結果表示のコーディング例

```
for ( i = 0; i < 10; i++ ) {
    System.out.println(num + "*" + i + "=" + (num * i) );
}
```

●確認問題4.8

- ① スコープ ② 無限(永久)ループ ③ 空文

●確認問題4.9

問題1 実行結果

```
>java Trace3
1
2 4
3 6 9

>
```

問題2 ネストしたfor文のコーディング例

```
for (int i = 10; 0 < i ; i-- ) {
    System.out.print("i=" + i + "¥t"); // iの値とタブの表示
    for (int j = i ; 0 < j ; j-- ) {
        System.out.print("o"); // oの表示(繰り返し)
    }
}
```

第5章

●確認問題 5.1

- ① if文 ② if else 文

●確認問題 5.2

合格判定のコーディング例

```
/* 合格判定 */
if (70 <= score) {
    System.out.println( "合格です" );
}
```

●確認問題 5.3

合格判定のコーディング例

```
/* 合格判定 */
if (score < 70) {
    System.out.println( "不合格です" );
} else {
    System.out.println( "合格です" );
}
```

●確認問題 5.4

- ① 評価はBです ② 評価はAです ③ 評価はCです
④ 評価はDです ⑤ 評価はCです ⑥ 評価はBです

●確認問題 5.5

文字列の比較のコーディング例

```
/* 文字列の比較 */
if ( str1.compareTo(str2) < 0 ) {
    // 比較結果表示文字列の作成
    appear += str1;
} else if ( str1.compareTo(str2) > 0 ) {
    // 比較結果表示文字列の作成
    appear += str2;
} else {
    // 比較結果表示文字列の作成
    appear = "一致:" + str1 + "と" + str2;
}
```

●確認問題 5.6

- 問題1 ① true ② false ③ true
 ④ false ⑤ true
- 問題2 ① true ② false ③ false
 ④ true ⑤ false ⑥ false
- 問題3 ① false ② true ③ true
 ④ true ⑤ true

第6章

●確認問題 6.1

問題1

- ① 0 ② 7

③

	[0]	[1]	[2]	[3]	[4]
arrayScore	6	9	0	10	7

④

	[0]	[1]	[2]	[3]	[4]
arrayScore	6	10	0	10	7

問題2

- ① `total ← arrayScore[0] + arrayScore[1] + arrayScore[2]`
- ② `total ← 0`
 `total ← total + arrayScore[0]`
 `total ← total + arrayScore[1]`
 `total ← total + arrayScore[2]`

●確認問題 6.2

- ① (○) 配列の宣言と配列の確保と同時に行い、要素 10 個の int 型配列を確保します。
- ② (×) `score[0]~score[9]` の 10 個を確保します。添字の最大値は 9 になります。
- ③ (×) 添字は 0 からはじまるので、要素番号 5 は先頭から 6 個目になります。
- ④ (○) `int[] score;` を使用している場合が多いですが、後置も正しい記述です。
- ⑤ (×) `new` 演算子で、配列を確保します。

●確認問題 6.3

問題 1 ①

問題 2 ③

●確認問題 6.4

これ以降コーディング例の解答ではソースコード全体ではなく、穴埋め部分の周辺のためのソースコードを抜粋して掲載している場合があります。ご注意ください。

コーディング例 (Q6_4_1.java)

```
int[] arrayTotalTen = new int[ arrayAmScore.length ]; // 合計点数

/* 合計点数の計算と表示 */
for (int i = 0; i < arrayAmScore.length; i++) {
    arrayTotalScore[i] = arrayAmScore[i] + arrayPmScore[i];
    System.out.println((i + 1) + " 人目の合計は "
                        + arrayTotalScore[i] );
}
```

「arrayAmScore.length」は、「arrayPmScore.length」でも可

●確認問題 6.5

問題 1 コーディング例 (Q6_5_1.java)

```
/* 値設定ループ */
for (int i = 0; i < arrayResult.length; i++) { // length属性
    arrayResult[i] = 1;
}

/* 配列要素の表示 */
for (int i = 0; i < arrayResult.length; i++) { // length属性
    System.out.print( arrayResult[i] + " ");
}
```

問題2 コーディング例(Q6_5_2.java)

```
/* 値設定ループ */
for (int i = 0; i < arrayNo.length; i++) {    // length属性
    arrayNo[i] = i + 1;
}

/* 配列要素の表示 */
for (int i = 0; i < arrayNo.length; i++) {    // length属性
    System.out.print(arrayNo[i] + " ");
}
```

問題3 コーディング例(Q6_5_3.java)

```
/* データの入力 */
for (int i = 0; i < arrayName.length; i++) {
    System.out.print((i + 1) + "人目の名前を入力>");
    arrayName[i] = br.readLine();                // 名前の入力
    System.out.print((i + 1) + "人目の点数を入力>");
    arrayScore[i] = Integer.parseInt(br.readLine()); // 点数の入力
}
```

●確認問題6.6

問題1 コーディング例(Q6_6_1.java)

```
int count = 0;                                // 人数集計エリアの宣言と初期化

/* 人数集計ループ */
for (int idx = 0; idx < arrayScore.length; idx++) {
    if (80 <= arrayScore[idx]) {
        count++;                            // 80点以上のとき人数カウントアップ
    }
}
```

問題2 コーディング例(Q6_6_2.java)

```
int count = 0;                                // 人数集計
int total = 0;                                // 点数合計
double average;                               // 平均

/* 集計ループ */
for (int i = 0; i < arrayScore.length; i++) {
```

続く>>>

```

        if ( [-1] < arrayScore[i]) {           // 欠席者を除いて集計
            total += [arrayScore[i] ];
            count++;
        }
    }
    average = (double) [total / count];        // 平均算出
    System.out.println("平均=" + average);

```

問題3 コーディング例(Q6_6_3.java)

```

int max = [0] ;                               // 最大値の初期化

/* 最大値の探索 */
for (int i = 0; i < arrayAdm.length; i++) {
    if ( [max] < arrayAdm[i]) {                // 最大値を超える値のとき
        max = [arrayAdm[i] ];                // 最大値の設定
    }
}

```

問題4 コーディング例(Q6_6_4.java)

```

for (idx = 0; idx < [arrayEreq.length]; idx++) {    // 配列初期化
    arrayEreq[idx] = 0;
}
for (idx = 0; idx < [arrayScore.length]; idx++) {  // 範囲の決定
    int j = [arrayScore[idx] / 10];                 // 位置算出
    arrayEreq[j]++;                                // 度数加算
}
for (idx = 0; idx < [arrayEreq.length - 1]; idx++){ // 表示
    System.out.println((idx * 10) + "点台:" + [arrayEreq[idx]]);
}
System.out.println((idx * 10) + "点:" + [arrayEreq[idx]]); // idx使用

```

問題5 コーディング例(Q6_6_5.java)

```

double[] arrayRatio = [new double[4]];            // 割合(配列)
int total = 0;

for (int idx = 0; idx < arrayEval.length; idx++) { // 人数累計
    total += [arrayCount[idx] ];
}
for (int idx = 0; idx < arrayEval.length; idx++) { // 割合算出

```

```

        arrayRatio[idx] = (double)arrayCount[idx] * 100 / total;
    }
    for (int idx = 0; idx < arrayEval.length; idx++) { // 表示
        System.out.print(arrayEval[idx] + "¥t");
        System.out.println(arrayRatio[idx] + "%");
    }

```

問題6 コーディング例(Q6_6_6.java)

```

int idx = (classNo - 1) * 10 + lot; // クラスと番号からidx算出
int namae = arrayName[idx]; // 配列から氏名設定

```

問題7 コーディング例(Q6_6_7.java)

```

/* 人数分の名前を設定 */
for (idx = 0; idx < n; idx++) {
    System.out.print("予約者>");
    arrayReserve[idx] = br.readLine(); // 名前の設定
}
/* 残りを空き情報に設定 */
for ( ; idx < arrayReserve.length; idx++) {
    arrayReserve[idx] = VACANT; // 空き情報の設定
}
/* 予約者リストの表示 */
System.out.println("予約者リスト");
for (idx = 0; idx < arrayReserve.length; idx++) {
    System.out.print(arrayReserve[idx] + "¥t");
}

```

初期化は
省略する

【解説】 空き情報の設定では、idxを続きからはじめるために初期設定がありません。

問題8 コーディング例(Q6_8.java)

```
/* arrayScolの要素数回ループ */
for (int idx1 = 0; idx1 < arrayScol.length; idx1++) {
    if (arrayScol[idx1] != -1) { // 欠席者以外るとき
        arrayScol2[idx2] = arrayScol[idx1]; // 配列要素のコピー
        idx2++; // 添字のカウントアップ
    }
}
/* コピーした要素数回ループ */
for (int idx1 = 0; idx1 < idx2; idx1++) {
    System.out.print(arrayScol2[idx1] + " "); // コピーデータの表示
}
```

●確認問題6.7

	[0]	[1]	[2]	[3]	[4]
arrayScore	0	100	100	100	91

●確認問題6.8

問題1 コーディング例(Q6_8_1.java)

```
for (int i = 0; i < arrayScore.length; i++) { // 列ごとの処理
    total[i] = 0;
    for (int j = 0; j < arrayScore[i].length; j++) {
        // 列の合計
        total[i] += arrayScore[i][j];
    }
}
```

問題2 コーディング例(Q6_8_2.java)

```
public class Q6_8_2 {
    public static void main(String[] args) {
        int [][] arrayGoban = {{0,1,1,1,1,1,1,0}, // 基盤データ
                                {0,0,2,2,1,2,0,0},
                                {1,0,2,2,2,2,2,2},
                                {2,1,2,2,1,2,2,0},
                                {2,2,2,1,1,2,2,2},
                                {1,2,2,1,1,2,2,2},
                                {0,0,2,2,1,2,0,0},
                                {0,0,2,2,2,2,0,0,0}};
```

続く>>>

```

int white = 0; // 白カウンタ
int black = 0; // 黒カウンタ
int stone; // 石データ
for (int i = 0; i < arrayGoban.length; i++) { // length属性
    // length属性
    for (int j = 0; j < arrayGoban[i].length; j++) {
        stone = arrayGoban[i][j];
        if (stone == 1) { // 石(白)判定
            white++;
        } else if (stone == 2) { // 石(黒)判定
            black++;
        }
    }
}
System.out.println("白=" + white); // 表示
System.out.println("黒=" + black);
}
}

```

第7章

●確認問題7.1

do while 文のコーディング例(Q7_1_1.java)

```

/* 配列arrayScore1の各要素を配列arrayScore2へコピー */
do {
    arrayScore2[i] = arrayScore1[i];
} while (arrayScore2[i++] != STOPPER); // STOPPERまで
/* 配列arrayScore2の要素を表示 */
i = 0;
do {
    System.out.print(arrayScore2[i] + " ");
} while (arrayScore2[i++] != STOPPER); // STOPPERまで
System.out.println();

```

問題 1 break 文を用いたコーディング例 (Q7_2_1.java)

```

/* 配列arrayScore1の各要素を配列arrayScore2へコピー */
for (int i = 0; i < arrayScore1.length; i++) {
    arrayScore2[i] = arrayScore1[i];
    if (arrayScore2[i] == STOPPER) {
        break;
    }
}
/* 配列arrayScore2の要素を表示 */
for (int i = 0; i < arrayScore2.length; i++) {
    System.out.print(arrayScore2[i] + " ");
    if (arrayScore2[i] == STOPPER) {
        break;
    }
}

```

問題 2 break 文を用いたコーディング例 (Q7_2_2.java)

```

/* 評価の判定と表示 */
for (i = 0; i < arrayCrit.length; i++) {
    if (arrayCrit[i] <= score) {
        break;
    }
}
System.out.println(arrayEvalu[i]);

```

問題 3 continue 文を用いたコーディング例 (Q7_2_3.java)

```

/* arraySco1の要素数回ループ */
int idx2 = 0; // コピー先の指標
for (int idx1 = 0; idx1 < arraySco1.length; idx1++) {
    if (arraySco1[idx1] == -1) { // 欠席者の場合
        continue; // 次のデータへ
    }
    arraySco2[idx2] = arraySco1[idx1]; // 配列要素のコピー
    idx2++; // 添字のカウントアップ
}

```

●確認問題 7.3

正しいものは ①と③

【解説】

- ① 文字列も判定できます。
- ② 文字列を指定するときは、ダブルクォーテーション (") で囲みます。
- ③ 文字リテラルの値をシングルクォート (') で囲んだ正しい記述です。
- ④ case 文に条件式を書くことはできません。

第8章

●確認問題 8.1

- ① ×
- ② ×
- ③ ×
- ④ ○
- ⑤ ×

●確認問題 8.2

- ① public
- ② private
- ③ protected

●確認問題 8.3

```
public class Q8_3 {
    public static void main( String[] args) {
        int total;
        int a1, a2;

        a1 = Integer.parseInt( args[0] );
        a2 = Integer.parseInt( args[1] );
        gokei = a1 + a2;
        System.out.println(a1 + "+" + a2 + "=" + total);
    }
}
```

●確認問題 8.4

```
public class Calc {
    /* 2つの値の加算を行うメソッド */
    public int add( int val1, int val2 ) {
        int answer;
        answer = val1 + val2;
        return answer ;
    }
}
```

続く>>>

```

}

/* 2つの値の減算を行うメソッド */
public int sub( int val1, int val2 ) {
    int answer;
    answer = val1 - val2;
    return answer;
}
}

```

第9章

●確認問題9.1

書式設定のコーディング例(Q9_1.java)

```

/* 税率を出力 */
form.applyPattern ("##.0%"); // 書式の設定
System.out.println("税率は " + form.format (TAX) + "です");

/* 各金額の出力 */
form.applyPattern ("¥¥###,###"); // 書式の設定
buf = " " + form.format (price); // 書式をもとに変換
buf = buf.substring(buf.length() - 8);
System.out.println("金額      :" + buf); // 金額を出力
buf = " " + form.format (taxPrice); // 書式をもとに変換
buf = buf.substring(buf.length() - 8);
System.out.println("税額      :" + buf); // 税額を出力
buf = " " + form.format (totalPrice); // 書式をもとに変換
buf = buf.substring(buf.length() - 8);
System.out.println("税込み金額 :" + buf); // 税込み金額を出力

```

●確認問題9.2

KeyInクラスを用いたコーディング例(Q9_2.java)

```

/* インスタンス生成 */
KeyIn ki = new KeyIn(); // キーボード入力クラス
                        // DecimalFormat
DecimalFormat form = new DecimalFormat();

                        // 金額の入力
price = Integer.parseInt( ki.readString("金額:") );
taxPrice = (int)(price * TAX); // 税額の計算
totalPrice = price + taxPrice; // 税込み金額の計算

```

●確認問題 9.3

修正後のプログラム

```
while (true) {
    String buf = ki.readString("年齢を入力:") ;           // 年齢を入力
    try {
        toshi = Integer.parseInt(buf) ;                   // 数値に変換
        break;
    } catch (NumberFormatException e) {                   // 例外処理
        System.out.println("年齢の入力に誤りがあります:" + buf);
    }
}
```

第 10 章

●確認問題 10.1

ファイルオープン ... ファイルを読み書きするための準備を行う

ファイルクローズ ... ファイルの読み書きを終えたあとの処理を行う

●確認問題 10.2

プログラム (FileOut.java)

```
import java.io.*;
public class FileOut{
    BufferedWriter bw = null;                               // BufferedWriter クラス

    /* ファイルのオープンを行うメソッド */
    public boolean open(String fname) {
        boolean sts = true;
        try {
            bw = new BufferedWriter(new FileWriter(fname)) ;
        } catch (IOException e) {
            System.out.println("ファイル名に誤りがあります¥n" + e);
            sts = false;
        }
        return sts;
    }

    /* ファイルへのデータ書き込みを行うメソッド */
    public boolean writeln(String str) {
```

続く>>>

```

        boolean sts = true;
        try {
            bw.write(str); // 1行分のデータをファイル出力
            bw.newLine(); // 行区切り文字を出力
        } catch (IOException e) {
            System.out.println("書き込みエラー¥n" + e);
            sts = false;
        }
        return sts;
    }

    /* ファイルのクローズを行うメソッド */
    public boolean close(){
        boolean sts = true;
        try {
            bw.close(); // ファイルのクローズ
        } catch (IOException e) {
            System.out.println("ファイルクローズエラー¥n" + e);
            sts = false;
        }
        return sts;
    }
}

```

●確認問題 10.3

プログラム(FileIn.java)

```

import java.io.*;

public class FileIn {
    BufferedReader br = null; // BufferedReaderクラス

    /* ファイルのオープンを行うメソッド */
    public boolean open(String fname) {
        boolean sts = true;
        try {
            br = new BufferedReader(new FileReader(fname));
        } catch (IOException e) {
            System.out.println("ファイル名に誤りがあります¥n" + e);
            sts = false;
        }
    }
}

```

続<>>

```

        return sts;
    }
    /* ファイルからデータの読み込みを行うメソッド */
    public String readLine() {
        String buff;
        try {
            buff = br. readLine();
        } catch ( IOException e ) {
            System.out.println(" 読み込みエラー¥n" + e);
            buff = null;
        }
        return buff;
    }

    /* ファイルのクローズを行うメソッド */
    public boolean close() {
        boolean sts = true;
        try {
            br. close();
        } catch ( IOException e ) {
            System.out.println(" ファイルクローズエラー¥n" + e);
            sts = false;
        }
        return sts;
    }
}

```

● 確認問題 10.4

- | | | |
|---------------------|------------------|-----------------|
| ① exists() | ② lastModified() | ③ setReadOnly() |
| ④ getAbsolutePath() | ⑤ canWrite() | ⑥ delete() |

A2 演習問題解答例

演習問題の解答例のほかにも、たくさんの正解プログラムが存在するでしょう。解答例にとらわれず、自由な発想でたくさんのプログラムを作ることを心がけてください。

第1章

■演習問題1.1

Javaの開発環境の用意については、[付録1 プログラム作成の準備]を参照してください。

ソースファイル名は、「HelloJava.java」として保存します。英大文字小文字の違いとスペルミスに注意してください。次にコンパイルとその実行結果を示します。

コンパイルと実行結果

```
>javac HelloJava.java

>java HelloJava
Hello Java world!

>
```

第2章

■演習問題2.1

誤りを修正したプログラム (DebugTraining1.java)

```
/*
 * デバッグ練習1
 */
public class DebugTraining1 {                // ① 誤 public → 正 public
                                              // ② 誤 nnain → 正 main
    public static void main(String[] args) {
        System.out.println("Hello!");      // ③ 誤 Out → 正 out
                                              // ④ 誤 system → 正 System
                                              // ; がなかった
        System.out.println("How you doing?");
        System.out.println("Okay thanks.");
                                              // ⑤ " がなかった
        System.out.println("How about you?");
                                              // ⑥ } がなかった
    }
}
```

- ① publicのスペルミス
- ② mainのスペルミス (コンパイルではエラーにならず、実行時エラーになる)
- ③ outの大文字小文字の違い
- ④ Systemの大文字小文字の違いとセミコロン(;)を付け忘れた
- ⑤ ダブルクォーテーション (") での文字列の囲みを閉じ忘れた
- ⑥ main() メソッドのコードブロック終了の } を付け忘れた

■演習問題2.2

誤りを修正したプログラム (DebugTraining2.java)

```
public class DebugTraining2 {                // ① クラス名とファイル名の不一致
                                              // ② static キーワードの抜け
    public static void main(String[] args) {
        System.out.println("こんにちは");   // ③ 1(いち)とl(エル)のまちがい
        System.out.println("元気ですか?"); // ④ 誤 : → 正 ;
    }                                         // ⑤ 誤 1 → 正 }
}
```

- ① 保存するファイル名とクラス名が同じでなければならない。英大文字小文字を区別する。
「DebugTraining2」クラス名が問題文では「Debugtraining2」と「t」が英小文字になっていた
- ② static キーワードが抜けていた（コンパイルではエラーにならず、実行時エラーになる）
- ③ println の l(エル)を 1(いち)とまちがえた
- ④ 文の終わりのセミコロン(;)をコロン(:)とまちがえた
- ⑤ コードブロックの終了の } を] とまちがえた

実行結果

```
>java DebugTraining2
こんにちは
元気ですか？
>
```

■演習問題 2.3

修正前のプログラムの実行結果

```
>java Trapez
台形の面積は2075です
>
```

一部の数値が文字列として連結された

台形の面積を求めるプログラム(Trapez.java)の修正例(3行目)

```
System.out.println("台形の面積は" + ((20 + 15) * 10 / 2) + "です");
```

外側の () を省略して $(20 + 15) * 10 / 2$ と記述しても問題はありません。

修正後のプログラムの実行結果

```
>java Trapez
台形の面積は175です
>
```

■演習問題 2.4

実験用のプログラム(Laboratory.java)で、エラーの原因になったコード6行目

```
System.out.println(10 / 0);
```

エラーの原因になったコード

エラーの原因は6行目の「10 / 0」です。整数演算で0を除数として除算を行ったときに**0除算 (Zero divide)**の実行時のエラー(**例外**)が発生します。そして、ArithmeticExceptionのエラーメッセージを表示してプログラムは**異常終了**します。

```
Exception in thread "main" java.lang.ArithmeticException: / by zero
    at Laboratory.main(Laboratory.java:6)
```

ただし、0.0を除数にするとこのエラーは発生しません。このときの実行結果は、**無限大 (Infinity)**になります。

実験用のプログラム (Laboratory2.java)

コーディング例

```
public class Laboratory2 {
    public static void main(String[] args) {
        System.out.println(10 / 0.0);           // 結果は無限大
    }
}
```

プログラムの実行結果

```
>java Laboratory2
Infinity
>
```

無限大 (Infinity)

■ 演習問題 2.5

コーディング例 (NameDisp1.java)

```
/*
 * 氏名を表示するプログラム1 (NameDisp1.java)
 */
public class NameDisp1 {
    public static void main(String[] args) {
        System.out.println("Seto");           // 名字の表示
        System.out.println("Marino");         // 名前の表示
    }
}
```

■演習問題 2.6

氏名の表示のコーディング例

```
System.out.println( "Seto " + "Marino" );
```

名字のあとに1文字分のスペースを入れています。

```
"名字△" + "名前" (△はスペース)
```

■演習問題 2.7

コーディング例 (ProfileDisplay.java)

```
/*
 * プロフィール表示するプログラム (ProfileDisplay.java)
 */
public class ProfileDisplay {
    public static void main(String[] args) {
        /* A Profile is displayed. */
        System.out.println("姓   : " + "瀬戸");
        System.out.println("名   : " + "雅彦");
        System.out.println("住所 : " + "札幌市中央区");
        System.out.println("TEL  : " + "011-111-8888");
    }
}
```

プログラムの実行例

```
>java ProfileDisplay
姓   : 瀬戸
名   : 雅彦
住所 : 札幌市中央区
TEL  : 011-111-8888

>
```

第3章

■ 演習問題 3.1

トレースの結果

実行される命令	変数 ans の値	変数 valA の値	変数 valB の値
変数の宣言	不定	不定	不定
①	↓	5	↓
②	↓	↓	5
③	10	↓	↓
④	↓	20	↓
⑤	25	↓	↓
⑥	50	↓	↓

プログラムの実行結果

```
>java Calc3
  ans = 50
>
```

■ 演習問題 3.2

プログラム (RectArea.java)

変数の宣言

```
int height; // 変数の宣言(縦)
int width;  // 変数の宣言(横)
int area;   // 変数の宣言(面積)
```

面積を求めるコード

```
area = height * width; // 面積を求める
```

■演習問題 3.3

コーディング例 (TriArea.java)

```
/*
 * 三角形の面積を求めるプログラム
 */
import java.io.*;

public class TriArea {
    public static void main(String[] args) throws IOException {
        int base;                // 底辺
        int height;              // 高さ
        int area;                // 面積

        /* 入力の準備 */
        BufferedReader br =
            new BufferedReader(new InputStreamReader(System.in));
        /* 底辺の入力 */
        System.out.println("底辺の長さを入力してください");
        base = Integer.parseInt(br.readLine());    // 底辺を入力
        /* 高さの入力 */
        System.out.println("高さを入力してください");
        height = Integer.parseInt(br.readLine()); // 高さを入力
        /* 面積を求めて表示 */
        area = base * height / 2;                // 面積を求める
        System.out.println("面積は " + area );    // 面積を表示する
    }
}
```

プログラムの実行例 (太字の部分が入力したデータ)

```
>java TriArea
底辺の長さを入力してください
38
高さを入力してください
24
面積は 456

>
```

■演習問題 3.4

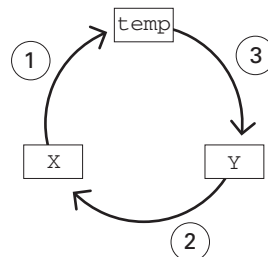
プログラムの解答例(Swap.java)

変数の宣言

```
int temp; // 作業エリア temp の宣言
```

データの入れ替えを行うコード

```
/* データの入れ替え */
temp = valX; // ①
valX = valY; // ②
valY = temp; // ③
```



■演習問題 3.5

プログラム(ConsTax.java)

税込み金額の算出を行うコード

```
money = (int)(money * 1.08); // 税込み金額を求める
System.out.println(" 税込み¥¥" + money); // 税込み金額を表示する
```

別解

```
money *= 1.08; // 税込み金額を求める
```

複合代入演算子の詳細は[第4章 繰り返し型のプログラム]を参照してください。

■演習問題 3.6

変数の宣言

```
double circ; // 円周を求める変数
double area; // 面積を求める変数
```

円周と面積を求めるコード

```
/* 円周を求めて表示する */
circ = 2 * 3.141592 * r; // 円周の算出

/* 面積を求めて表示する */
area = 3.141592 * r * r; // 面積の算出
```

【参考】

Mathクラスを利用したコーディング例

```
circ = 2 * Math.PI * r; // 円周の算出
```

Math.PIは定数で、3.14159265358979323846です。

```
area = Math.PI * Math.pow(r, 2); // 面積を求める
```

Math.pow(n, m)は、nのm乗を求めるメソッドです。Math.pow(r, 2)では、半径(r)の2乗を求めています。

第4章

■演習問題4.1

① 1から10までの和を求めるプログラム(Sum1.java)

コーディング例(プログラムの一部)

```
sum = 0; // 初期化
i = 1;
while (i < 11) {
    sum += i;
    i++;
}
System.out.println(sum);
```

② 1から10までの和を求めるプログラム(Sum2.java)

コーディング例(プログラムの一部)

```
sum = 0; // 初期化
i = 10;
while (i > 0) {
    sum = sum + i;
    i--;
}
System.out.println(sum);
```

③ 1から10までの和を求めるプログラム(Sum3.java)

コーディング例(プログラムの一部)

```
sum = 0; // 初期化
i = 0;
while (i < 10) {
    i++;
    sum += i;
}
```

続<>>>

```

    }
    System.out.println(sum);
}

```

ほかにもたくさん解答例がありそうです。いろいろなパターンでコーディングして確認しましょう。

実際には、次のようにループ制御を使用しなくてもコーディングすることが可能です。しかし、累計処理のプログラムがループ制御を理解するためには重要です。

1から10までの和を求めるプログラム (Sum4.java)

ループ制御を用いないコーディング例

```

public class Sum4 {
    public static void main(String[] args) {
        int i = 10;
        int sum = i * (i + 1) / 2;    // 一式で総和を算出
        System.out.println(sum);
    }
}

```

■演習問題 4.2

プログラム名 (EvenTotal.java)

コーディング例 (プログラムの一部)

```

int sum = 0;                                // 総和の累計エリア (sum) の初期化
for (int i = 2; i <= 100; i += 2) {
    sum += i;
}
System.out.println(sum);

```

Question の答え 「2500」

■演習問題 4.3

プログラム名 (SumFrom1ToN.java)

コーディング例 (プログラムの一部)

```

/* 総和の累計エリア (sum) の宣言と初期化 */
int sum = 0;
/* 1からnまでの総和を求める */
for (int i = 1; i <= n; i++) {
    sum += i;
}
System.out.println(sum);

```

Questionの答え 「65535」

【解説】

これは難問でした。int型データが保持できる値の範囲は-2147483648 ~ +2147483647です。つまり、最大値2147483647になるまでは正しく計算できます。1から65535までの和は2147450880ですから(1からnまでの和 = $n \times (n + 1) \div 2$)正しく求められます。しかし、65536までの和を求めたときにint型の整数の範囲を超えた結果(2147516416)になります。これを**オーバーフロー**といいます。しかし、オーバーフローしたからといってエラーでプログラムが異常終了することはありません。

ですから、プログラムは常にどのような範囲で値が格納されるかを想定してプログラミングする必要があります。この場合、変数sumをlong型で定義しておくと、とりあえず正しい結果が得られます。

■演習問題 4.4

プログラム名(InputAverage.java)

コーディング例(プログラムの一部)

```
String buf; // 入力バッファ
int sum = 0; // 合計エリア
int i = 0; // 件数カウンタ
/* 入力の繰り返し */
while ((buf = br.readLine()) != null) { // 入力があれば繰り返し
    sum += Integer.parseInt(buf); // 点数の累計
    i++; // データ件数のカウント
}
System.out.println("合計:" + sum); // 合計の表示
System.out.println("件数:" + i); // データ件数の表示
System.out.println("平均:" + (sum / i)); // 平均の表示
```

■演習問題 4.5

プログラム名(Multiple1.java)

while文を使用したコーディング例

```
public class Multiple1 {
    public static void main(String[] args) {
        int i = 1; // 縦方向のカウンタ
        while (i < 10) { // 9まで繰り返し
            int j = 1; // 横方向のカウンタ
            while (j < 10) { // 9まで繰り返し
                System.out.print(" " + i * j); // 九九の表示
                j++;
            }
        }
    }
}
```

続<>>>

```

        System.out.println();           // 改行
        i++;
    }
}

```

プログラム名 (Multiple2.java)

for 文を使用したコーディング例

```

public class Multiple2 {
    public static void main(String[] args) {
        for (int i = 1; i < 10; i++) {           // 変数 i で縦方向を制御
            for (int j = 1; j < 10; j++) {       // 変数 j で横方向を制御
                System.out.print(" " + (i * j)); // 九九の表示
            }
            System.out.println();               // 改行
        }
    }
}

```

第5章

■演習問題 5.1

プログラム名 (OddEvenJudge.java)

奇数・偶数判定のコーディング例

```

/* 奇偶判定 */
if ( (num % 2) == 0 ) {
    System.out.println(num + " は偶数です");
} else {
    System.out.println(num + " は奇数です");
}

```

【解説】

偶数 (2 の倍数) の場合、2 で剰余を求めて余りが 0 だったときに偶数であると判断できます。

同様に x の倍数であればその値で剰余を求めれば、その値の倍数か否かを判断できます。

◆応用課題5.1.1

プログラム名(MultipleOfX.java)

【プログラムの変更箇所】

```
if ( (num % x) == 0 ) {  
    System.out.println(num + "は" + x + "の倍数です");  
} else {  
    System.out.println(num + "は" + x + "の倍数ではありません");  
}
```



コンピュータの内部では、数値を2進数で取り扱っています。その2進数の性質を利用して奇偶数の判定を
すると、次のようにコーディングできます。

```
if ((num & 0x00000001) == 0) {
```

これは、最下位ビットだけを抽出するために1と論理積(&)をとっています。論理演算子の論理積(&)は
ビットの操作にも使えます。2つの数値の論理積(&)は両方のビットが1のとき1、それ以外は0にします。
0x00000001は16進数で表現しましたが1と記述しても同じです。

一般的な「論理演算」、および「ビット反転」のしくみを次の表に示します。

表 B6.1 ビット操作のしくみ

	論理積(&)	論理和()	排他的論理和(^)	ビット反転(~)
演算例	0011	0011	0011	~ 01
	& 0101	0101	^ 0101	10
	0001	0111	0110	

ビット操作には、ビットの論理演算のほかにも、ビットを右や左にずらす「シフト演算」があります。Javaの
シフト演算には、表に示した演算子が用意されています。シフト演算子は、整数型の値に使用されます。
また、Javaでは、byte 型などのint 型より小さなサイズのデータの演算を行う際に、int 型に変換してから
演算される特性を持つため、想定していた結果が得られないことがあるので注意してください。

表 B6.2 シフト演算子

演算子	使用例	説明
<<	x << n	xのビット列をnビットだけ「左にシフト」させる。 右端から空いた部分には「0」を埋める。これを左シフトと呼ぶ (一般に論理左シフトと呼ばれるものと同じ操作)
>>>	x >>> n	xのビット列をnビットだけ「右にシフト」させる。 左端から空いた部分には「0」を埋める。 これを論理右シフトと呼ぶ
>>	x >> n	xのビット列をnビットだけ「右にシフト」させる。 左端から空いた部分には「元の最上位ビット」と 同じ符号を埋める。これを算術右シフトと呼ぶ

シフト演算の例

式	10進数	2進数
x = 10;	10	0000 0000 0000 0000 0000 0000 0000 1010
x << 2;	40	00 0000 0000 0000 0000 0000 0000 0010 1000
		あふれたビット 空きに「0」
x = 10;	10	0000 0000 0000 0000 0000 0000 0000 1010
x >> 2;	2	0000 0000 0000 0000 0000 0000 0000 0010 10
		空きに「0」 あふれたビット
x = -10;	-10	①111 1111 1111 1111 1111 1111 1111 0110
x >> 2;	-3	1111 1111 1111 1111 1111 1111 1111 1101 10
		最上位ビット 空きに「元の最上位ビット」 あふれたビット

■演習問題5.2

プログラム名 (PassJudge.java)

合格判定のコーディング例

```

final int PASSING_MARK = 50;           // 定数 合格点(50)
.
.
.

String message = "";                   // 表示メッセージ
/* 合格判定 */
if (score < PASSING_MARK) {             // 合否判定
    int shortage = PASSING_MARK - score; // 不足分の計算
                                           // メッセージの設定
    message = "あと " + shortage + "点で合格です";
} else {
    message = "合格です";               // 合格メッセージの設定
}
System.out.println(message);           // メッセージの表示

```

【解説】

点数が50点未満の場合は、合格の基準点から点数を引いて不足分の点数を求めています。

「final int PASSING_MARK = 50;」は、int型の定数です。定数の宣言は[第3章 データの記憶 コラム]を参照してください。

■ 演習問題 5.3

プログラム名 (Maximum.java)

最大値の判定のコーディング例

```
/* 最大値の判定 */
if ( (x < y) ) {
    if ( (y < z) ) {
        max = MAX_MESSAGE + z;
    } else {
        max = MAX_MESSAGE + y;
    }
} else {
    if ( (x < z) ) {
        max = MAX_MESSAGE + z;
    } else {
        max = MAX_MESSAGE + x;
    }
}
System.out.println(max);
```

◆ 応用課題 5.3.1

最大値の判定のコーディング例

まず、最初のデータを最大値としてスタートします。

```
/* 最大値の判定 */
int maxvalue = x;

if ( (maxvalue < y) ) {
    maxvalue = y;
}
if ( (maxvalue < z) ) {
    maxvalue = z;
}
System.out.println(MAX_MESSAGE + maxvalue);
```

◆ 応用課題 5.3.2

プログラム名 (Minimum.java)

最小値の判定のコーディング例

```
/* 最小値の判定 */
```

続<>>>

```
int minValue = x;

if ( y < minValue ) {
    minValue = y;
}
if ( z < minValue ) {
    minValue = z;
}

System.out.println(MIN_MESSAGE + minValue);
```

■演習問題 5.4

プログラム名(Multiple.java)

コーディング例

```
public class Multiple {
    public static void main(String[] args) {
        for (int i= 1; i < 10; i++) {           // 変数iで縦方向を制御
            for (int j = 1; j < 10; j++) {       // 変数jで横方向を制御
                String space = " ";            // 余白の設定
                int ans = i * j;                 // 九九の計算
                if (ans < 10) {                  // 10以下のとき
                    space += " ";               // 余白の追加
                }
                System.out.print(space + ans);   // 九九の表示
            }
            System.out.println();               // 改行
        }
    }
}
```

【解説】

かけ算の結果が10未満のときに余白を1文字分追加しています。これによって表示する際の桁合わせを行っています。ただし、このほかにもさまざまな方法が存在します。

■演習問題 5.5

問題が発生するケース

「年齢の判定」と「性別の判定」をそれぞれに分けているため、「18歳未満の女性だったとき」は料金が半額の半額、すなわち、「料金が1/4」になってしまう。

論理エラーの解消方法

2つに分けていたif文をOR条件でひとつにまとめるのが、訂正のひとつの方法です。以下にその訂正例を示します。

で囲った部分の修正例

```
/* 条件を判定 年齢と性別 */
if ((age < 18) || gender.equals(FEMALE)) {
    price /= 2; // 半額計算
}
```

■演習問題5.6

問題が発生するケース

「0未満」や「100を超える」データが入力されても、エラーメッセージが表示されない。

論理エラーの解消方法

点数の範囲をチェックしているif文のOR条件をAND条件に訂正します。このように、データの値が指定された値を超えているかどうかの検査することをリミットチェックといいます。

問題のあるコード

```
if (0 <= score || score <= 100) { // 点数の範囲チェック
```

コードの修正例

```
if (0 <= score && score <= 100) { // 点数の範囲チェック
```

第6章

■演習問題6.1

プログラム名(Practice6_1.java)

金種計算のコーディング例

```
/* 金種の計算 */
for (int i = 0; i < arrayDenomi.length; i++) {
    arraySheet[i] = price / arrayDenomi[i];
    price %= arrayDenomi[i];
}
```

続く>>>

```
/* 金種の表示 */
for (int i = 0; i < arrayDenomi.length; i++) {
    System.out.println(arrayDenomi[i] + "¥t" + arrayMaisu[i]);
}
```

■ 演習問題 6.2

プログラム名 (Practice6_2.java)

科目ごとの平均計算のコーディング例

```
/* 科目平均の計算 */
for (int j = 0; j < arrayScore[0].length; j++) {
    subjectTotal = 0;
    for (int i = 0; i < arrayScore.length; i++) {
        subjectTotal += arrayScore[i][j];
    }
    subjectAverage[j] =
        (double)subjectTotal / arrayScore.length;
}

/* 科目平均の表示 */
for (int i = 0; i < arrayScore[0].length; i++) {
    System.out.println(subjectAverage[i]);
}
```

■ 演習問題 6.3

プログラム名 (Practice6_3.java)

最小値を探すコーディング例

```
int min = arrayAdm[0]; // 配列0番目を最小値とする
for (int i = 1; i < arrayAdm.length; i++) {
    if (arrayAdm[i] < min) { // 最小値との比較
        min = arrayAdm[i]; // 最小値の入れ替え
    }
}
```

第7章

■演習問題7.1

プログラム名 (Practice7_1.java)

数当てプログラムのコーディング例

```
/* 当たるまでループを繰り返す */
while ( true ) {
    /* 数の入力 */
    System.out.print("数:>");
    number = Integer.parseInt(br.readLine());

    /* 当たりの判定 */
    if ( hitNumber < number ) {                // 大き過ぎたとき
        System.out.println(number + "より小さい値です");
    } else if ( number < hitNumber ) {         // 小さ過ぎたとき
        System.out.println(number + "より大きい値です");
    } else {                                  // 一致したとき
        System.out.println(number + "で当たります");
        break;
    }
}
```

演習問題

第8章

■演習問題8.1

Arithmetic クラス (Arithmetic.java) のコーディング例

```
public class Arithmetic {
    /* double 値の絶対値 */
    public double absolute(double value) {
        if (value < 0) {                // 正負の判定
            value *= -1.0;              // 符号の反転
        }
        return value;                  // 絶対値 (double 値) の返却
    }

    /* int 値の絶対値 */
    public int absolute(int value) {
        return (int)absolute((double)value); // 絶対値 (int 値) の返却
    }
}
```

続>>>

```

/* 四捨五入 */
public int round(double value) {
    int sign = 1;                                // 符号
    if (value < 0) {                              // 正負の判定
        sign = -1;                                // 負の設定
    }
    value += (0.5 * sign);                        // 符号を考慮し四捨五入する
    return (int)value;                            // int 値で返却
}
}

```

■演習問題 8.2

Search クラス(Search.java)のコーディング例

```

public class Search {
    /* 最小値の探索 */
    public int minimum(int[] arrayData) {
        int minValue = Integer.MAX_VALUE;        // 最小値の初期値設定

        for (int i = 0; i < arrayData.length; i++) {
            if (arrayData[i] < minValue) {        // 最小値を超える値のとき
                minValue = arrayData[i];          // 最小値の設定
            }
        }
        return minValue;                          // 最小値の返却
    }

    /* 最大値の探索 */
    public int maximum(int[] arrayData) {
        int maxValue = Integer.MIN_VALUE;        // 最大値の初期値設定

        for (int i = 0; i < arrayData.length; i++) {
            if (maxValue < arrayData[i]) {        // 最大値を超える値のとき
                maxValue = arrayData[i];          // 最大値の設定
            }
        }
        return maxValue;                          // 最大値の返却
    }
}
}

```

第9章

■演習問題 9.1

プログラム名(KeyIn.Java)

KeyInクラスに追加するメソッド

```
/* 整数値を入力するメソッド*/
public int readInt() {
    int inputIntValue;
    while ( true ) {
        buf = readString();
        try {
            // 数値に変換
            inputIntValue = Integer.parseInt(buf);
            break; // ループ終了
        } catch (NumberFormatException e) { // 数値変換のエラー
            System.out.println(" 整数値を入力してください:" + buf);
            System.out.print(" 再入力>");
        }
    }
    return inputIntValue; // 変換した値の返却
}

/* 入力プロンプトを表示して整数値を入力するメソッド */
public int readInt(String msg) {
    System.out.print(msg + ">"); // プロンプトを表示
    return readInt(); // readInt() の呼び出し
}
```

■演習問題9.2

プログラム名(Practice9_2.java)

入力した値とそれぞれの平均を画面に表示するプログラムのコーディング例

```
/* 最大10人分の年齢・給与をキーボード入力する処理 */
for ( cnt = 0; cnt < MAX_NUMBER; cnt++ ) {
    age[cnt] =
        Integer.parseInt( ki.readString ((cnt + 1) + "人目の年齢:"));
    if (age[cnt] == 999) break; // ストップ入力で終了
    sal[cnt] =
        Integer.parseInt( ki.readString ((cnt + 1) + "人目の給与:"));
    ageTotal += age[cnt]; // 年齢合計
    salTotal += sal[cnt]; // 給与合計
}
if ( cnt != 0 ) { // 入力データ件数の判定
    /* 人数分のデータを書式編集し画面に出力する処理 */
    form.applyPattern ("¥¥###,###");
    System.out.println("番号 年齢 給与"); // タイトルを画面出力
    for ( int i = 0; i < cnt; i++ ) { // 人数分のループ
        // スペース3個と連結

        buf = " " + Integer.toString(i + 1);
        buf = buf.substring(buf.length() - 4);
        System.out.print(buf);

        // スペース4個と連結

        buf = " " + Integer.toString(age[i]);
        buf = buf.substring(buf.length() - 5);
        System.out.print(buf);

        // スペース10個と連結

        buf = " " + form.format (sal[i]);
        buf = buf.substring(buf.length() - 11);
        System.out.println(buf);
    }
    /* 平均算出の処理 */
    ageAvg = ageTotal / cnt; // 平均年齢の算出
    salAvg = salTotal / cnt; // 平均給与の算出

    /* 平均年齢・平均給与を書式編集し画面に出力する処理 */
    System.out.println("-----");
    System.out.print(" 平均");
    form.applyPattern ("###");
```

続く>>>

```

        buf = "        " + form.format (ageAvg);           // スペース4個と連結
        buf = buf.substring(buf.length() - 5);
        System.out.print (buf);
        form.applyPattern ("¥¥###,###");

                                                                    // スペース10個と連結
        buf = "        " + form.format (salAvg);
        buf = buf.substring(buf.length() - 11);
        System.out.print (buf);
    } else {
        System.out.println("-----");
        System.out.println("入力データがありません");
    }
}

```

第10章

■演習問題10.1

プログラム名(Q10_1.java)

入力データの組をcsv形式のファイルへ出力するプログラムのコーディング例

```

/*
 * ファイル名をキーボード入力し、そのファイル名でファイルオープンを行う
 * ファイルオープンが正常に行われるまでループする
 */
ret = false;
while (ret != true) {
    fname = ki.readString("出力ファイル名:"); // ファイル名入力
    ret = fo.open(fname);                     // ファイルオープン
}
/* 複数の年齢と給与の組をキーボード入力する処理 */
while (true) {
                                                                    // 年齢入力
    age = ki.readInt((count + 1) + "人目の年齢:");
    if (age == 999) break;

                                                                    // 給与入力
    salary = ki.readInt((count + 1) + "人目の給与:");
    /* 自動採番した番号と入力文字列(年齢、給与)をファイルへ出力する */
    buf = Integer.toString(count + 1) + STR_COMMA +
           Integer.toString(age) + STR_COMMA +
           Integer.toString(salary);
}

```

続く>>>

```

        ret = fo. writeln(buf) ;                // 1レコード出力
        if (ret == false) {                      // エラー判定
            fo.close();
            System.out.println("プログラムを異常終了します");
            System.exit(1);
        }
        count++;                                // カウントアップ
    }
    /* ファイルのクローズ処理 */
    ret = fo. close() ;                        // ファイルクローズ
    if (ret == false) {
        System.out.println("プログラムを異常終了します");
        System.exit(1);
    }
    System.out.print( count + "件のデータをファイルに出力しました");

```

■演習問題10.2

プログラム名(Q10_2.java)

csv形式のファイルから入力したデータを編集出力するプログラムのコーディング例

```

/*
 * ファイル名をキーボード入力し、そのファイル名でファイルオープンを行う処理
 * ファイルオープンが正常に行われるまでループ
 */
ret =false;
while (ret != true) {
    fname = ki.readString("入力ファイル名:"); // ファイル名入力
    ret = fi. open(fname) ;                // ファイルオープン
}

/*
 * ファイルから複数のレコードを入力し、年齢と給与を切り出して数値に変換
 * 変換した年齢と給与をそれぞれ年齢合計および給与合計に加算
 * すべてのレコードを処理したらループから抜ける
 */
while (true) {
    buf = fi. readLine() ;                // 1レコード入力
    if( buf == null ) break;                // 入力終了
    try {

```

続く>>>

```

/*
 * StringTokenizerクラスを用いてトークンを指定して
 * カンマごとのデータを切り出す
 */
StringTokenizer tkn =                // 番号
    new StringTokenizer(buf, STR_COMMA);
String dummy = tkn.nextToken();

                                // 年齢
ageTotal += Integer.parseInt(tkn.nextToken());

                                // 給与
salTotal += Integer.parseInt(tkn.nextToken());
/* ファイル中のデータに誤りがあった場合の例外処理 */
} catch (NoSuchElementException e) {    // カンマ区切りの誤り
    System.out.println("データに誤りがあります:" + buf);
    System.out.println("プログラムを異常終了します:" + e);
    fi.close();
    System.exit(1);
} catch (NumberFormatException e) {      // 数字列の誤り
    System.out.println("データに誤りがあります:" + buf);
    System.out.println("プログラムを異常終了します:" + e);
    fi.close();
    System.exit(1);
} catch (Exception e) {                // 予期せぬエラー
    System.out.println("予期せぬエラーでプログラムを異常終了します:" + e);
    fi.close();
    System.exit(1);
}

    count++;                        // カウントアップ
}
/* ファイルのクローズ処理 */
ret = fi.close();                    // ファイルクローズ
if (ret == false) {
    System.out.print("プログラムを異常終了します");
    System.exit(1);
}
/* 平均年齢と平均給与を書式編集して画面に出力 */
System.out.println(count + "件のデータを入力しました");
if (count != 0) {
    ageAvg = ageTotal / count;        // 平均年齢の算出

```

続く>>>

```

        salAvg = salTotal / count;           // 平均給与の算出
    System.out.println("平均年齢:" + ageAvg + " 歳");
    form.applyPattern("###,###,###円");
    System.out.println("平均給与:" + form.format(salAvg));
}

```

A3 プログラミング総合演習プログラム例

プログラミング総合演習の解答には独自に定義した入出力クラス (KeyIn, FileIn, FileOut クラス) を使用しています。解答例を提示する前にそれらのクラスを紹介します。解答例は A-53 ページから掲載しています。

(0) 入出力クラス

プログラムで使用する KeyIn クラス、FileIn クラスおよび FileOut クラスを紹介します。

【KeyIn クラス KeyIn.java】

```
import java.io.*;

/*
 * キーボードから入力されたデータを読み取ります。
 * 読み取り可能なデータは文字列 (String 型データ)、および整数 (int 型データ) です。
 */

public class KeyIn {
    String buf = null;                // 入力バッファの初期化
    BufferedReader br =               // BufferedReader クラスのインスタンス
        new BufferedReader(new InputStreamReader(System.in));

    /*
     * 文字列を入力するメソッドです。
     * 入力された文字列を返します。
     * 行の内容を含む文字列、ただし行の終端文字は含めない
     * ストリームの終わりに達している場合 ([Ctrl]+[Z]) は、null を返す
     * [Enter] キーのみが入力された場合は、"" を返す
     */
    public String readString() {
        try {
            buf = br.readLine();      // キーボード入力
        } catch (IOException e) {     // キーボード入力致命的エラー
            e.printStackTrace();      // エラー情報の表示
            System.exit(1);           // プログラムの異常終了
        }
        return buf;                  // 文字列を返却
    }
}
```

続く>>>

```

/*
 * 入力プロンプトを表示して文字列を入力するメソッドです。
 * 入力された文字列を返します。
 * 指定された文字列msgに続き ">" を連結しプロンプト表示する。
 */
public String readString(String msg) {
    System.out.print(msg + ">");    // プロンプト表示
    return readString();           // キーボード入力
}

/*

 * 整数値を入力するメソッドです。
 * 入力された数字列を整数値に変換して返します。
 * 入力された文字列に数字以外のデータが入力されたり、int 型の範囲を超えるデータが
 * 入力されたりした場合は、" 再入力>" プロンプトを表示して再入力を要求します。
 * これは、正しく入力されるまで繰り返されます。
 */
public int readInt() {
    int inputIntValue;
    while (true) {
        buf = readString();
        try {
                                // 数値に変換
            inputIntValue = Integer.parseInt(buf);
            break;                // ループ終了
        } catch (NumberFormatException e) {
                                // 数値変換のエラー
            System.out.println(" 整数値を入力してください:" + buf);
            System.out.print(" 再入力>");
        }
    }
    return inputIntValue;        // 変換した値の返却
}

/*
 * 入力プロンプトを表示して整数値を入力するメソッドです。
 * 入力された数字列を整数値に変換して返します。
 */
public int readInt(String msg) {

```

```

        System.out.print(msg + ">");    // プロンプトを表示
        return readInt();                // readInt() の呼び出し
    }
}

```

【FileIn クラス FileIn.java】

```

import java.io.*;

/*
 * テキストファイルから入力されたデータを読み取ります。
 */
public class FileIn {
    BufferedReader br = null;                // BufferedReader クラス

    /* ファイルのオープンを行うメソッド */
    public boolean open(String fname) {
        boolean sts = true;
        try {                                // ファイルオープンに相当する処理
            br = new BufferedReader(new FileReader(fname));
        } catch (IOException e) {
            System.out.println("ファイル名に誤りがあります¥n" + e);
            sts = false;
        }
        return sts;
    }

    /* ファイルからデータの読み込みを行うメソッド */
    public String readLine() {
        String buff;
        try {
            buff = br.readLine();            // 1行分のデータをファイルから入力
        } catch (IOException e) {
            System.out.println("読み込みエラー¥n" + e);
            buff = null;
        }
        return buff;
    }

    /* ファイルのクローズを行うメソッド */
    public boolean close() {

```

```

        boolean sts = true;
        try {
            br.close(); // ファイルのクローズ
        } catch (IOException e) {
            System.out.println("ファイルクローズエラー¥n" + e);
            sts = false;
        }
        return sts;
    }
}

```

【FileOutクラス FileOut.java】

```

import java.io.*;

/*
 * 指定されたデータをテキストファイルへ書き込みます。
 */
public class FileOut{
    BufferedWriter bw = null; // BufferedWriterクラス

    /* ファイルのオープンを行うメソッド */
    public boolean open(String fname) {
        boolean sts = true;
        try { // ファイルオープンに相当する処理
            bw = new BufferedWriter(new FileWriter(fname));
        } catch (IOException e) {
            System.out.println("ファイル名に誤りがあります¥n" + e);
            sts = false;
        }
        return sts;
    }

    /* ファイルへのデータ書き込みを行うメソッド */
    public boolean writeln(String str) {
        boolean sts = true;
        try {
            bw.write(str); // 1行分のデータをファイル出力
            bw.newLine(); // 行区切り文字を出力
        } catch (IOException e) {
            System.out.println("書き込みエラー¥n" + e);
        }
    }
}

```

続く>>>

```

        sts = false;
    }
    return sts;
}

/* ファイルのクローズを行うメソッド */
public boolean close(){
    boolean sts = true;
    try {
        bw.close(); // ファイルのクローズ
    } catch (IOException e) {
        System.out.println("ファイルクローズエラー¥n" + e);
        sts = false;
    }
    return sts;
}
}

```

■問題1.1

【テーマ】配列データの順位付け

【プログラム例 Ranking.java】

```

/*
 * 順位付け
 */
public class Ranking {
    public static void main(String[] args) {
        /* テストデータの設定 */
        int[] arrayMark = { 90, 75, 100, 60, 95, 70, 85, 80, 90, 80 };

        /* 順位テーブル */
        int[] arrayRank = { 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 };

        /* 順位付け */
        for (int i = 0; i < arrayMark.length; i++) {
            for (int j = 0; j < arrayMark.length; j++) {
                if (arrayMark[i] < arrayMark[j]) {
                    arrayRank[i]++;
                }
            }
        }
    }
}

```

続く>>>

```

    }
}

/* 順位付け後の配列要素の表示 */
for (int i = 0; i < arrayMark.length; i++) {
    System.out.println((i + 1) + "人目¥t" +
        arrayMark[i] + "点¥t" +
        arrayRank[i] + "位");
}
}
}

```

■問題1.2

【テーマ】2次元配列 魔方陣

【プログラム例 MagicSqr.java】

```

/*
 * 「ド・ラ・ルーブルの方法」による奇数魔方陣の作成
 */
public class MagicSqr {
    public static void main(String[] args) {
        /* キーボード入力のインスタンスを生成する */
        KeyIn ki = new KeyIn();

        /* 魔方陣の次数を入力する */
        int numChar = ki.readInt("奇数の魔方陣");

        if ((numChar < 0) || ((numChar % 2) == 0)) {

            /* 1より小さい数と偶数はエラー */
            System.out.println("1以上の奇数を入力してください");

        } else {

            /* 魔方陣を作成する配列を定義する */
            int magicSqr[][] = new int[numChar][numChar];

```

続く>>>

```

/* 魔方陣を作成する */
// ルール (a) 「スタート位置」
int i = 0; // 最上位行の中央
int j = numChar / 2;
magicSqr[i][j] = 1;

// 最終値は次数の2乗
for (int k = 2; k <= (numChar * numChar); k++) {

    // ルール (d), (f) 「下へ」
    if ((k % numChar) == 1) {
        i++; // 次数の倍数は「下」になる

        // ルール (b) 「配列の上側」
    } else if (i == 0) {
        i = numChar - 1; // 右列の最下位行へ
        j++;

        // ルール (c) 「配列の右」
    } else if (j == (numChar - 1)) {
        i--; // 上の行の最左（先頭）列
        j = 0;

        // ルール (e) 「右上」
    } else {
        i--; // 右斜め上の列へ
        j++;
    }
    magicSqr[i][j] = k;
}

/* 魔方陣を表示する */
for (i = 0; i < numChar; i++) {
    for (j = 0; j < numChar; j++) {
        System.out.print(magicSqr[i][j] + " ");
    }
    System.out.println();
}

int wa = numChar * (numChar * numChar + 1) / 2; // 要素の和

```

終く>>>

```

        System.out.println("要素の和=" + wa);
    }
}
}

```

■問題 2.1

【テーマ】線形探索(シーケンシャルサーチ)

【プログラム例 LinearSearch.java】

```

/*
 * 線形探索
 */
public class LinearSearch {
    public static void main(String[] args) {
        /* キーボード入力のインスタンスを生成する */
        KeyIn ki = new KeyIn();

        int[] arrayScore = { 102, 800, 201, 101, 283, 430, 181, 550 };
        String[] arraySname = { "事務機B型", "応接机", "会議用机",
                                "事務機A型", "折り畳み机", "木製机",
                                "脇机", "学生机" };

        /* 商品コードを入力する */
        int scode = ki.readInt("商品コード:");

        /* 線形探索 */
        int idx = 0;
        while ((idx < arrayScore.length) &&
                (scode != arrayScore[idx])) {
            idx++;
        }

        /* 探索結果の判定 */
        if (idx < arrayScore.length) {
            System.out.println(arraySname[idx]);
        } else {
            System.out.println("登録されていない商品コードです");
        }
    }
}

```

続く>>>

```
}  
}
```

■問題 2.2

【テーマ】線形探索(番兵法)

【プログラム例 GuardSearch.java】

```
/*  
 * 線形探索 (番兵法)  
 */  
public class GuardSearch {  
    public static void main(String[] args) {  
        int[] arrayScode = { 102, 800, 201, 101, 283,  
                             430, 181, 550, -1 };  
        String[] arraySname = { "事務機B型", "応接機", "会議用機", ,  
                                 "事務機A型", "折り畳み機", "木製機",  
                                 "脇机", "学生机", " " };  
  
        /* キーボード入力のインスタンスを生成する */  
        KeyIn ki = new KeyIn();  
  
        /* 商品コードを入力する */  
        int scode = ki.readInt("商品コード:");  
  
        /* 線形探索(番兵法) */  
        arrayScode[arrayScode.length - 1] = scode; // 番兵の設定  
        int idx = 0;  
        while (scode != arrayScode[idx]) {  
            idx++;  
        }  
  
        /* 探索結果の判定 */  
        if (idx < (arrayScode.length - 1)) {  
            System.out.println(arraySname[idx]);  
        } else {  
            System.out.println("登録されていない商品コードです");  
        }  
    }  
}
```

```
}
```

■問題 2.3

【テーマ】二分探索(binary search : バイナリサーチ)

【プログラム例 BinarySearch.java】

```
/*
 * 二分探索
 */
public class BinarySearch {
    public static void main(String[] args) {
        /* キーボード入力のインスタンスを生成する */
        KeyIn ki = new KeyIn();

        int[] arrayScore = { 101, 102, 181, 201, 283, 430, 550, 800 };
        String[] arraySname = { "事務機A型", "事務機B型", "脇機",
                                "会議用機", "折り畳み機", "木製機",
                                "学生機", "応接機" };

        boolean found = false ;                // 見つからないことを前提
        int mid = 0;

        /* 商品コードを入力する */
        int scode = ki.readInt("商品コード:");

        /* 二分探索 */
        int lo = 0;                            // 要素の下限
        int hi = arrayScore.length - 1;        // 要素の上限

        while (lo <= hi) {
            mid = (lo + hi) / 2;
            if (scode == arrayScore[mid]) {
                found = true;                  // 見つかった
                break;                         // ループを抜ける
            } else if (scode < arrayScore[mid]) {
                hi = mid - 1;
            } else {
                lo = mid + 1;
            }
        }
    }
}
```

続く>>>

```

    }

    if (found == true) {                                // 成功
        System.out.println(arraySname[mid]);
    } else {                                            // 失敗
        System.out.println("登録されていない商品コードです");
    }
}
}
}

```

■問題 2.4

【テーマ】 区間探索

【プログラム例 SectionSearch.java】

```

/*
 * 区間探索
 */
public class SectionSearch {
    public static void main(String[] args) {
        int[] arrayWeight = { 50, 75, 100, 150, 200, 250,
                               500, 750, 1000, 2000, 3000, 4000 };
        int[] arrayPostage = { 120, 140, 160, 200, 240, 270,
                                390, 580, 700, 950, 1150, 1350 };

        /* キーボード入力のインスタンスを生成する */
        KeyIn ki = new KeyIn();

        /* 重量を入力する */
        int weight = ki.readInt("重量:");

        if ((weight < 0) || (4000 < weight)) {
            System.out.println("取り扱いできません");
        } else {
            /* 区間探索 */
            int idx = 0;
            while (arrayWeight[idx] < weight) {
                idx++;
            }
        }
    }
}

```

続く>>>

```

        System.out.println(arrayPostage[idx] + "円");
    }
}
}

```

■問題 3.1

【テーマ】選択法(selection sort : セレクションソート)

【プログラム例 SelectionSort.java】

```

/*
 * 整列 (選択法: セレクションソート)
 *   小さな値を順に決定して整列する方法
 */
public class SelectionSort {
    public static void main(String[] args) {
        int[] arrayScore = { 102, 800, 201, 101, 283, 430, 181, 550 };
        String[] arraySname = { "事務機B型", "応接机", "会議用机",
                                "事務機A型", "折り畳み机", "木製机",
                                "脇机", "学生机" };

        int tempScore;                // 商品コード交換用作業領域
        String tempSname;              // 商品名交換用作業領域

        /* 選択法による整列 */
        for (int i = 0; i < (arrayScore.length - 1); i++) {
            for (int j = i + 1; j < arrayScore.length; j++) {
                if (arrayScore[i] > arrayScore[j]) {
                    /* 商品コード交換 */
                    tempScore = arrayScore[i];
                    arrayScore[i] = arrayScore[j];
                    arrayScore[j] = tempScore;
                    /* 商品名交換 */
                    tempSname = arraySname[i];
                    arraySname[i] = arraySname[j];
                    arraySname[j] = tempSname;
                }
            }
        }
    }
}

```

続く>>>

```

        /* 整列後の内容の表示 */
        for (int i = 0; i < arrayScore.length; i++) {
            System.out.println(arrayScore[i] + "¥t" + arraySname[i]);
        }
    }
}

```

■問題3.2

【テーマ】挿入法(insert sort : インサートソート)

【プログラム例 InsertSort .java】

```

/*
 * 整列 (挿入法)
 *   部分列が整列するように値を挿入して整列する方法
 */
public class InsertSort {
    public static void main(String[] args) {
        int[] arrayScore = { 102, 800, 201, 101, 283, 430, 181, 550 };
        String[] arraySname = { "事務機B型", "応接機", "会議用機",
                                "事務機A型", "折り畳み機", "木製機",
                                "脇机", "学生機" };

        int tempScore;                                // 交換のための作業領域
        String tempSname;                              // 交換のための作業領域

        /* 挿入法による整列 */
        for (int i = 1; i < arrayScore.length; i++) {
            for (int j = i; 0 < j; j--) {
                if (arrayScore[j - 1] > arrayScore[j]) {
                    tempScore = arrayScore[j];
                    arrayScore[j] = arrayScore[j - 1];
                    arrayScore[j - 1] = tempScore;

                    tempSname = arraySname[j];
                    arraySname[j] = arraySname[j - 1];
                    arraySname[j - 1] = tempSname;
                } else {
                    break;
                }
            }
        }
    }
}

```

```

    }

}

/* 整列後の内容の表示 */
for (int i = 0; i < arrayScore.length; i++) {
    System.out.println(arrayScore[i] + "¥t" + arraySname[i]);
}
}
}

```

■問題 3.3

【テーマ】 交換法 (bubble sort : バブルソート)

【プログラム例 BubbleSort.java】

```

/*
 * 整列 ( 交換法 : バブルソート )
 *   小さな値を配列の先頭に移動させて整列する方法
 */
public class BubbleSort {
    public static void main(String[] args) {
        int[] arrayScore = { 102, 800, 201, 101, 283, 430, 181, 550 };
        String[] arraySname = { "事務機B型", "応接機", "会議用機",
                                "事務機A型", "折り畳み機", "木製機",
                                "脇機", "学生機" };

        /* 交換法による整列 */
        for (int i = 0; i < (arrayScore.length - 1); i++) {
            boolean sortFlag = true;           // 整列フラグtrue
            for (int j = (arrayScore.length - 1); i < j; j--) {
                if (arrayScore[j - 1] > arrayScore[j]) {
                    int tempScore;              // 交換のための作業領域
                    String tempSname;           // 交換のための作業領域

                    tempScore = arrayScore[j];
                    arrayScore[j] = arrayScore[j - 1];
                    arrayScore[j - 1] = tempScore;

                    tempSname = arraySname[j];
                    arraySname[j] = arraySname[j - 1];

```

続く>>>

```

        arraySname[j - 1] = tempSname;

        sortFlag = false;                // 整列フラグfalse
    }
}
/* スキャン中に一度も交換されなければ整列完了 */
if (sortFlag){
    break;
}
}
/* 整列後の内容の表示 */
for (int i = 0; i < arrayScode.length; i++) {
    System.out.println(arrayScode[i] + "¥t" + arraySname[i]);
}
}
}

```

■問題4.1

【テーマ】スタック(stack)

【プログラム例 Stack.java (ループ処理の部分)】

```

/* スタック処理の種別を入力する */
while (0 <= (kind = ki.readInt("種別(0:表示,1:push,2:pop):"))) {

    switch (kind) {
    case 0:                                // 表示
        if (sp == STACK_EMPTY) {
            System.out.println("スタックされていません");
        } else {
            for (int i = sp ; STACK_EMPTY < i; i--) {
                System.out.println "[" + i + "]" +
                    arrayStack[i]);
            }
        }
        break;

    case 1:                                // push
        if (sp == (arrayStack.length - 1)) {

```

続く>>>

```

        System.out.println("スタックが一杯です");
    } else {
        inData = ki.readInt("データ:");
        sp++;
        arrayStack[sp] = inData;
    }
    break;

case 2:                                     // pop
    if (sp == STACK_EMPTY) {
        System.out.println("スタックされていません");
    } else {
        System.out.println(arrayStack[sp]);
        sp--;
    }
    break;

default:                                    // 種別エラー
    System.out.println("種別の入力エラーです");
    break;
}
}
System.out.println("---- プログラム終了 ----");

```

■問題 4.2

【テーマ】待ち行列(queue : キュー)

【プログラム例 Queue.java (switch caseの部分)】

```

switch (kind) {                             // 処理種別の判定
case 0:                                     // 表示
    if (front == rear) {
        System.out.println("データが格納されていません");
    } else if (front < rear) {
        for (int i = front ; i < rear; i++) {
            System.out.println "[" + i + " ] " +
                                arrayQueue[i]);
        }
    } else {
        // リングバッファの表示
    }
}

```

続く>>>

```

        for (int i = front ; i < element; i++) {
            System.out.println "[" + i + " ] " +
                arrayQueue[i];
        }
        for (int i = 0 ; i < rear; i++) {
            System.out.println "[" + i + " ] " +
                arrayQueue[i];
        }
    }
    break;

case 1:                                     // 格納 (enqueue)
                                           // ひとつの空きの要素を残す
    if (((rear + 1) % element) == front) {
        System.out.println("キューが一杯です");
    } else {
        inData = ki.readInt("データ:");
        arrayQueue[rear] = inData;
        rear = (rear + 1) % element;
    }
    break;

case 2:                                     // 取り出し (dequeue)
    if (front == rear) {
        System.out.println("データが格納されていません");
    } else {
        System.out.println(arrayQueue[front]);
        front = (front + 1) % element;
    }
    break;

default:                                   // 種別エラー
    System.out.println("種別の入力エラーです");
    break;
}

```

【テーマ】 リスト構造(list)

【プログラム例 ListSearch.java】

```

/*
 * リスト探索
 */
public class ListSearch {
    public static void main(String[] args) {
        /* キーボード入力のインスタンスを生成する */
        KeyIn ki = new KeyIn();
        /* 確認用テストデータ */
        final int END_OF_LIST = -1;          // リストの終端
        int[] arrayNumber = { 20, 40, 10, 0, 30, 0, 0, 0 };
        int[] arrayScore = { 22, 44, 11, 0, 33, 0, 0, 0 };
        int[] arrayNext = { 4, -1, 0, 5, 1, 6, 7, -1 };
        int lip = 2;                          // リストの先頭を示すポインタ
        int eip = 3;                          // 空きリストの先頭を示すポインタ
        int idx;                              // 配列の添字

        int lot;                              // 探索する学生番号

        /* 番号に0が入力されるまで繰り返す */
        while (0 < (lot = ki.readInt("学生番号(終了=0):"))) {

            /* リスト探索 */
            idx = lip;
            while ((idx != END_OF_LIST) &&
                    (arrayNumber[idx] != lot)) {
                idx = arrayNext[idx];
            }

            /* 探索結果の表示 */
            if (idx != -1) {
                System.out.println("点数=" + arrayScore[idx]);
            } else {
                System.out.println("見つかりませんでした");
            }
        }
    }
}

```

続く>>>

```

    }
    System.out.println("---- プログラム終了 ----");
}
}

```

■問題 4.4

【テーマ】 二分探索木(binary search tree)

【プログラム例 BinarySearchTree.java】

```

/*
 * 二分木探索
 */
public class BinarySearchTree {
    public static void main(String[] args) {
        /* キーボード入力のインスタンスを生成する */
        KeyIn ki = new KeyIn();
        /* 確認用テストデータ */
        final int END_OF_TREE = -1;          // 要素の終端
                                           // 学籍番号
        int[] arrayNumber = { 5, 2, 7, 9, 1, 3, 0, 0 };
                                           // 点数
        int[] arrayScore = { 50, 20, 70, 90, 10, 30, 0, 0 };
                                           // 左ポインタ部
        int[] arrayLeft = { 1, 4, -1, -1, -1, -1, -1, -1 };
                                           // 右ポインタ部
        int[] arrayRight = { 2, 5, 3, -1, -1, -1, -1, -1 };
        int pt = 0;                          // 先頭要素のポインタ

        int lot;                              // 探索する学籍番号
        int idx;                              // 配列の添字

        /* 学籍番号に0が入力されるまで繰り返す */
        while (0 < (lot = ki.readInt("学籍番号 (終了=0):"))) {

            /* 二分探索木の探索 */
            idx = pt;
            while ((idx != END_OF_TREE) &&
                    (arrayNumber[idx] != lot)) {

```

続く>>>

```

        if (arrayNumber[idx] < lot) {
            idx = arrayRight[idx];
        } else {
            idx = arrayLeft[idx];
        }
    }

    /* 探索結果の表示 */
    if (idx != END_OF_TREE) {
        System.out.println("点数=" + arrayScore[idx]);
    } else {
        System.out.println("見つかりませんでした");
    }
}
System.out.println("---- プログラム終了 ----");
}
}

```

■問題 5.1

【テーマ】キーボードからのデータ入力、ソートとファイル出力

【プログラム例 **SeisekiSort.java**】

```

/*
 * キーボード入力した成績データをソートしてファイル (csv形式) 出力
 */

import java.text.*;

public class ResultSort {
    public static void main(String [] args) {
        final String COMMA = ","; // 区切り文字 (定数)

        /* インスタンス生成 */
        FileOut fo = new FileOut(); // ファイル出力クラス
        KeyIn ki = new KeyIn(); // キーボード入力クラス
        // 書式設定クラス

        DecimalFormat form = new DecimalFormat();
        boolean ret; // メソッドの戻り値
    }
}

```

続く>>>

```
String fileName = null; // 入力ファイル名
String buff; // 入力バッファ
int num; // ファイルのレコード件数

/* 処理対象のデータ件数を入力 */
while (true) {
    num = ki.readInt("データ件数指定 (1-100):");
    if ((0 < num) && (num <= 100)) break;
    System.out.println("入力できるデータ件数は、100件までです:" + num);
}

/* レコード件数分の配列を定義 */
int[] arrayNumber = new int[num]; // 配列学籍番号
String[] arrayName = new String[num]; // 配列氏名
int[] arrayScore = new int[num]; // 配列試験点数

/* 指定された件数のデータを入力 */
for (int i = 0; i < arrayNumber.length; i++) {
    while (true) {
        arrayNumber[i] = ki.readInt("学籍番号:");
        if (arrayNumber[i] <= 9999999) break;
        System.out.println("学籍番号が7桁を超えています:"
            + arrayNumber[i]);
    }
    while (true) {
        arrayName[i] = ki.readString("氏名");
        if (arrayName[i].length() <= 16) break;
        System.out.println("氏名が16桁を超えています:"
            + arrayName[i]);
    }
    while (true) {
        arrayScore[i] = ki.readInt("点数");
        if (0 <= arrayScore[i] && arrayScore[i] <= 100) break;
        System.out.println("点数が0~100の範囲を超えています:"
            + arrayScore[i]);
    }
}
System.out.println("---- データ入力終了 ----");
```

```

/* セレクションソート（スキャンごとの交換） */
for (int i = 0; i < (num - 1); i++) {
    int minNumber = arrayNumber[i];           // 最小の学籍番号
    int minPos = i;                           // 最小データの添字
    for (int j = (i + 1); j < num; j++) {
        if (arrayNumber[j] < minNumber) {
            minNumber = arrayNumber[j];
            minPos = j;
        }
    }
}

/* スキャン終了後、データの入れ替えが必要か判定 */
if (minPos != i) {
    int tempScore;                           // 点数の作業領域
    String tempName;                         // 氏名の作業領域

    /* データの交換 */
    arrayNumber[minPos] = arrayNumber[i];
    arrayNumber[i] = minNumber;

    tempName = arrayName[i];                 // 氏名
    arrayName[i] = arrayName[minPos];
    arrayName[minPos] = tempName;

    tempScore = arrayScore[i];               // 点数
    arrayScore[i] = arrayScore[minPos];
    arrayScore[minPos] = tempScore;
}
}

System.out.println("---- 学籍順Sort完了 ----");

/*
 * ファイル名をキー入力し、そのファイル名でファイルオープンを行う処理
 * ファイルオープンが正常に行われるまでループ
 */
System.out.println("データを成績ファイルに出力します");
System.out.println("ファイル名を入力してください");
ret = false;

```

```

while (ret != true) {
    fileName = ki.readString("ファイル名:"); // ファイル名入力
    ret = fo.open(fileName);                // ファイルオープン
}

/* 各レコードを編集して、ファイルに出力 */
form.applyPattern("00000000");             // 学籍番号の書式 (7桁)
for (int i = 0; i < arrayNumber.length; i++) {

    buff = form.format(arrayNumber[i]) + COMMA
            + arrayName[i] + COMMA
            + Integer.toString(arrayScore[i]);
    ret = fo.writeln(buff);
    if (ret == false) {                     // エラー発生時の処理
        System.out.print("プログラムを異常終了します");
        ret = fo.close();
        System.exit(1);
    }
}

/* ファイルクローズ */
ret = fo.close();                          // ファイルクローズ
if (ret == false) {                        // エラー発生時の処理
    System.out.print("プログラムを異常終了します");
    System.exit(1);
}
System.out.println("==== プログラム終了 ====");
}
}

```

【テーマ】 ファイル入力とサーチ

【プログラム例 ResultSearch.java】

```

/*
 * 成績ファイル (csv形式) の入力とサーチ
 */

import java.util.*;

public class ResultSearch {
    public static void main(String [] args) {
        final String COMMA = ",";           // 区切り文字 (定数)

        /* インスタンス生成 */
        FileIn fi = new FileIn();           // ファイル入力クラス
        KeyIn ki = new KeyIn();             // キーボード入力クラス

        boolean ret;                        // メソッドの戻り値
        String fileName = null;             // 入力ファイル名
        String buff;                        // 入力バッファ
        int num;                            // ファイルのレコード件数
        /*
         * ファイル名をキー入力し、そのファイル名でファイルオープンを行う処理
         * ファイルオープンが正常に行われるまでループ
         */
        ret = false;
        while (ret != true) {
                                                    // ファイル名入力
            fileName = ki.readString("成績ファイル名:");
            ret = fi.open(fileName);           // ファイルオープン
        }

        /* レコード件数のカウント */
        num = 0;
        while (fi.readLine() != null) {
            num++;
        }
    }
}

```

```

/* ファイルクローズ */
ret = fi.close(); // ファイルクローズ
if (ret == false) { // エラー発生
    System.out.print("プログラムを異常終了します");
    System.exit(1);
}

/* レコード件数分の配列を定義 */
String[] arrayNumber = new String[num]; // 配列学籍番号
String[] arrayName = new String[num]; // 配列氏名
String[] arrayScore = new String[num]; // 配列試験点数
String searchlot; // 検索対象学籍番号

/* もう一度、ファイルオープン */
fi.open(fileName);

/*
 * ファイルから複数のレコードを入力
 * 各データを切り出して配列に格納
 * すべてのレコードを入力し終わったらループから抜ける
 */
for (int i = 0; (buff = fi.readLine()) != null; i++) {
    try {
        /*
         * StringTokenizerクラスを用いてトークンを指定して
         * カンマごとのデータを切り出す
         */
        StringTokenizer tkn =
            new StringTokenizer(buff, COMMA);
        arrayNumber[i] = tkn.nextToken(); // 学籍番号
        arrayName[i] = tkn.nextToken(); // 氏名
        arrayScore[i] = tkn.nextToken(); // 試験点数

        /* ファイル中のデータに誤りがあった場合の例外処理 */
    } catch (NoSuchElementException e) { // カンマ区切りの誤り
        System.out.println("データに誤りがあります:" + buff);
        System.out.println("プログラムを異常終了します:" + e);
        fi.close();
        System.exit(1);
    }
}

```

続く>>>

```

    } catch (Exception e) { // 予期せぬエラー
        System.out.println(" 予期せぬエラーでプログラムを異常終了します："
            + e);

        fi.close();
        System.exit(1);
    }
}

/* ファイルクローズ */
ret = fi.close(); // ファイルクローズ
if (ret == false) { // エラー発生
    System.out.print(" プログラムを異常終了します");
    System.exit(1);
}

/*
 * 検索対象学籍番号をキー入力し対象となるデータを配列から検索
 * 対象となるデータがあればそのデータを画面に出力
 */
while (true) {
    searchlot = ki.readString(" 検索対象番号:");
    /* 終了判定 */
    if ((searchlot == null) ||
        searchlot.equals("")) break;

    /* 対象データのサーチ */
    boolean searchFlag = false; // サーチフラグ
    int i = 0;
    while (i < arrayNumber.length) {
        if (searchlot.compareTo(arrayNumber[i]) == 0) {
            searchFlag = true;
            break;
        } else if (searchlot.compareTo(arrayNumber[i]) < 0) {
            break;
        }
        i++;
    }
    if (searchFlag) {
        System.out.println(" 学籍番号: " + searchlot);
        System.out.println(" 氏名      : " + arrayName[i]);
    }
}

```

続く>>>

```

        System.out.println("点数    : " + arrayScore[i]);
    } else {
        System.out.println(" 指定されたデータは見つかりませんでした ");
    }
}
System.out.println("---- プログラム終了 ----");
}
}

```

■問題6.1

【テーマ】最大公約数(ユークリッドの互除法)と最小公倍数を求める

【プログラム例 Euclid.java】

```

/*
 * 最大公約数(ユークリッドの互除法)
 * 最小公倍数
 */
public class Euclid {
    public static void main(String[] args) {
        /* キーボード入力のインスタンスを生成する */
        KeyIn ki = new KeyIn();

        /* 最大公約数と最小公倍数を求める2つの数値を入力する */
        int a = ki.readInt("a:");           // aの入力
        int b = ki.readInt("b:");           // bの入力
        int div;                             // 除数
        int rem;                             // 余り
        int gcd;                             // 最大公約数
        int lcm;                             // 最小公倍数

        /* ユークリッドの互除法 */
        gcd = a;
        div = b;
        while (0 < div) {
            rem = gcd % div;
            gcd = div;
            div = rem;
        }
    }
}

```

続く>>>

```

        lcm = (a * b) / gcd;                                // 最小公倍数の算出
        System.out.println("GCD = " + gcd);                // 最大公約数
        System.out.println("LCM = " + lcm);                // 最小公倍数
    }
}

```

■問題 6.2

【テーマ】 エラトステネスのふるいの方法により素数を求める

【プログラム例 Eratosthenes.java】

```

/*
 * 「エラトステネスのふるい(篩)」法による素数表の生成
 *      100まで      25個
 *      1000まで     168個
 *      10000まで    1229個
 *      100000まで   9592個
 *      1000000まで  78498個
 */
public class Eratosthenes {
    public static void main(String[] args) {
        /* キーボード入力のインスタンスを生成する */
        KeyIn ki = new KeyIn();

        /* 素数を求める範囲を入力する */
        int maxNumber = ki.readInt("いくつまで:");

        /* 判定の終端 */
        int limits = (int) Math.sqrt((double) maxNumber);

        /* ふるい: 要素の0番目を使用しないので(計算範囲 + 1)の要素が必要 */
        boolean[] arrayPrime = new boolean[maxNumber + 1];

        /*
         * 初期設定
         *      2は素数なのでtrueを設定する
         *      以降の2の倍数は素数ではないので、trueを設定しない
         *      3以上の奇数にtrueを設定する
         */

```

続く>>>

```

arrayPrime[2] = true;
for (int i = 3; i < arrayPrime.length; i += 2) {
    arrayPrime[i] = true;
}

/* nの倍数をfalseに設定する */
for (int i = 3; i <= limits; i += 2) {
    if (arrayPrime[i]) { // 素数であれば、以降その倍数をfalseにする
        for (int j = i * 2; j < arrayPrime.length; j += i) {
            arrayPrime[j] = false;
        }
    }
}

/* 素数を表示する */
int count = 0; // 求めた素数の個数
for (int i = 2; i < arrayPrime.length; i++) {
    if (arrayPrime[i]) {
        System.out.print(i + "¥t");
        count++;
    }
}
System.out.println();
System.out.println(count + "個");
}
}

```

[illegible]

[illegible]

