16章解答

1． キーボードをタイプして文字列を入力し、それが辞書にある言葉かどうか検査して結果を表示するプログラムを作成しなさい。クラスには次の 2 つのメソッドを作成します。

A. isWord メソッド

・文字列を仮引数 word に受け取り、それが辞書にあれば true をなければ false を返す

・辞書には次の 5 つの言葉だけを持ち、配列を辞書として使う。辞書配列は次のようである。

```
String dic = {"JT101","JJ200","JX800","JV800","JJX800"};
```

B. main メソッド

・キーボードをタイプして文字列を入力し、それを isWord メソッドに渡して辞書にあるかどうか検査し、結果を「あり」「なし」のどちらかで表示する

【解答欄】@20

```
import lib.Input;
public class P3 {

     public static void main(String[] args) {
          String word = Input.getString();
          if(isWord(word)){
               System.out.println("あり");
          }else{
               System.out.println("なし");
          }

     }
     public static boolean isWord(String word) {
           String[] dic = {"JT101","JJ200","JX800","JV800","JJX800"};
           boolean  match = false;
           for(String s: dic){
                if(s.equals(word)){
                     match = true;
                     break;
                }
           }
           return match;
     }
}
```

《解説》

配列の中を探索する処理をメソッドとする典型的な例です。isWord メソッドの中で boolean 型の変数 match を宣言して最初に false を入れておくのは定石のようなものです。for 文で配列の要素を調べ、該当すればそこで match=true; として break します。これを return true;として戻っても構いませんが、そうするとメソッドの中に return false で戻る return 文が別に必要です。プログラムが複雑になった時、メソッドの中の return はできるだけひとつにする方がよいので、ここでもそのやり方を取っています。

16章解答

２．　例えば 100 人分の身長データなどがあるとき、そのどれもが同じような値か、あるいはいろいろな値が混ざっているかを判断するには、それぞれの値から全体の平均値を引いた値を見ます。この値を偏差といいます。

統計処理ではデータ全体の傾向を見るため、偏差を 2 乗したものを全て合計します。さらに合計をデータ個数で割った値は分散といいます。しかし、よく利用されるのは分散の平方根で、これを標準偏差といいます。標準偏差はデータのばらつき具合を示す指標として利用されます。

例えば、データが{2,3,4,3,3,3}であれば、データ数は 6、平均は 3 ですから次のように計算します。

偏差の 2 乗の合計　vt　=　$(2-3)^2+(3-3)^2+(4-3)^2+(3-3)^2+(3-3)^2+(3-3)^2 = 2$

標準偏差　　sd　=　$\sqrt{\frac{vt}{n}} = \sqrt{\frac{2}{6}} = 0.57735$

ただし、n はデータの数です。この例では 6 になります。

そこで次の 4 つのメソッド持つプログラム（クラス）を作成してください。

A．mean メソッド

・データの配列を仮引数に受け取って平均値を計算して返します

B．vt メソッド

・データの配列を仮引数に受け取って偏差の 2 乗の合計を計算して返す
　ただし，vt は mean メソッドを利用してデータの平均を求めます

C．sd メソッド

・データの配列を仮引数に受け取り vt メソッドを利用して偏差の 2 乗の合計を求め、それから標準偏差を計算して返します

D．main メソッド

・sd メソッドを使って{15.1, 1.5, 17.2, 7.3, 21.0, 3.8, 6.8}の標準偏差を求めて表示します

＜ヒント＞

・平方根は Math.sqrt()を使います

16章解答

【解答欄】@20

```
public class P2 {
    public static void main(String[] args) {
         double[] dt = {15.1, 1.5, 17.2, 7.3, 21.0, 3.8, 6.8};
         double sd = sd(dt);
         System.out.println("標準偏差＝" +sd);
    }
    // 標準偏差を計算する
    public static double sd(double[] a){
        return Math.sqrt(vt(a)/a.length);
    }
    // 偏差平方和（偏差の２乗の合計）を計算する
    public static double vt(double[] a){
        double mean = mean(a);
        double vt = 0;
        for(double x : a){
            vt += Math.pow(x-mean,2);
        }
        return vt;
    }
    // 平均値を計算する
    public static double mean(double[] a){
         double  total = 0;
         for(double d : a){
             total +=d;
         }
         return   total/a.length;
    }
}
```

《解説》

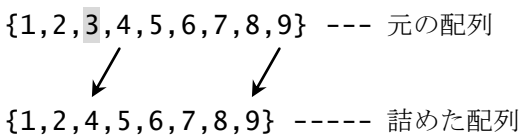
main → sd → vt → mean のように、次々にメソッド呼び出しが連鎖する典型的なプログラムです。これは、段階的詳細化といって、最初 mean で大まかなプログラムを作成し、処理の詳細は sd メソッドで実行することにしています。また、sd メソッドでは面倒な偏差平方和の計算を vt メソッドで行うことにして、処理を簡単にしています。さらに vt メソッドでも平均を取るのは mean メソッドに任せて自分の処理を簡単にしています。

このように、複雑な処理を子メソッドに任せて、「処理の中身が分かり易いプログラム」を作成する手法が段階的詳細化の手法です。プログラム作成時の、有力な思考方法です。

16章解答

３． 配列の途中の要素を削除して残りを左へ詰める操作を行う delete メソッドを作成します。例は、要素３を削除し、４以下をひとつ左へ詰める処理を示しています。

例： 3 を削除した場合



削除とはいっても、元の配列の中から直接削除するのではなく新しい配列を作成します。元の配列をコピーして新しい配列を作成しますが、k 番目の要素を削除するのであれば、新しい配列には k 番目の要素をコピーしないという処理を行います。新しい配列の要素数は元の配列よりも１少なくなります。

次のようにメソッドを作成してください。

A. delete メソッド

・元の配列と削除する要素の要素番号を受け取る。元の配列を受け取る仮引数として int[]a, 要素番号を受け取る仮引数として int k を持つ
・削除後の配列 b を作成する。要素数は a より１少ない
・b に a をコピーするが、k 番目の要素だけはコピーしない
・コピーした配列 b を戻り値として返す

B. main メソッド

・int[]a = {10, 5, 15, 7, 2, 13, 11, 8, 9, 21}を作成する
・delete メソッドを呼び出して、a の 2 番目の要素(0,1,2･･･と数えるので、15 がその要素)を削除した配列 b を作成する
・b のすべての要素をコンソールに表示する

16章解答

【解答欄】@20

```
public class P3 {
    public static void main(String[] args) {
        int[] a = {10, 5, 15, 7, 2, 13, 11, 8, 9, 21};
        int k=2;
        int[] b = delete( a, k);
        //int[] b = another_delete( a, k); // 別解
        for(int x : b){
            System.out.print(x + "¥t");
        }
    }
    public static int[] delete(int[] a, int k){
        int[] b = new int[a.length-1];// 一つ少ない要素数でb[]を作成
        int   j=0;     // 配列の要素番号
        for(int i=0; i<a.length; i++){ // a[]の要素数だけ繰り返す
            if(i==k) continue;          // i==kならスキップする
            b[j++] = a[i];              // コピー
        }
        return b;
    }

    public static int[] another_delete(int[] a, int k){ //  別解
        int[] b = new int[a.length-1];
        for(int i=0; i<k; i++){
            b[i] = a[i];
        }
        for(int i=k+1; i<a.length; i++){
            b[i-1] = a[i];
        }
        return b;
    }
}
```

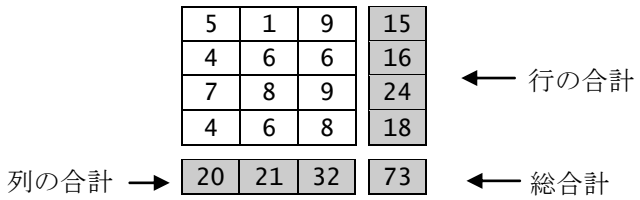
《解説》

delete は配列要素のうち、指定された k 番目の要素だけを除外して別の配列にコピーすることにより、k 番目の要素を含まない新しい配列を作成します。

まずは、下線を引いている箇所のように、新しい配列を new 文で作成しておくことが絶対必要です。次にコピーするわけですが、k 番目になったらスキップするという方法や 0～k-1 番目までと、k+1 番目から以降とに分けてコピーする方法（別解）が考えられます。

16章解答

４． 図のように表の縦方向の合計、横方向の合計と総合計を求める処理を作成します。表のサイズは図では４行×3 列ですが、どんなサイズの表でも計算できるようなプログラムにします。



表は 2 次元の配列で表現します。上の例では次のように定義できます。

```
int[][] a = {
              {5,1,9},
              {4,6,6},
              {7,8,9},
              {4,6,8}
            };
```

クラス（プログラム）では、次の 3 つの下請けメソッドを作成します。

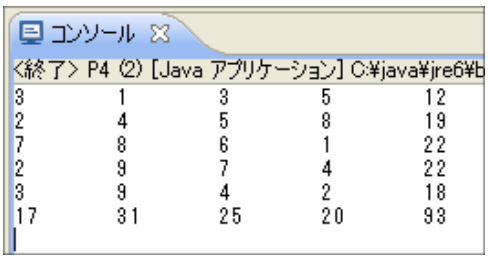
A. sumRow　配列 a を仮引数に受け取り行の合計を計算し結果を配列で返す

B. sumCol　配列 a を仮引数に受け取り列の合計を計算し結果を配列で返す

C. sum　配列 a を仮引数に受け取り全体の合計を計算して返す

さらに main メソッドを作成し、下の表について縦方向の合計、横方向の合計と総合計を表示してください。

3	1	3	5
2	4	5	8
7	8	6	1
2	9	7	4
3	9	4	2



＜ヒント＞

・sumRow では a.length 個の要素を持つ配列を作成します

・sumCol では a[0].length 個の要素を持つ配列を作成します

16章解答

【解答欄】@20

```
package pass;
public class P4_別 {
    public static void main(String[] args) {
        int[][]a = new int[][]{
                    {3, 1, 3, 5},
                    {2, 4, 5, 8},
                    {7, 8, 6, 1},
                    {2, 9, 7, 4},
                    {3, 9, 4, 2}
        };
        int[] row =  sumRow(a); // 行合計
        int[] col =  sumCol(a); // 列合計
        for(int i=0; i<a.length; i++){
            for(int elm : a[i]){ // i 番目の行配列（a[i]）の要素をすべて表示する
                System.out.print(elm + "¥t");
            }
            System.out.println(row[i]); // 第 i 行の合計を表示する
        }
        for(int col_elm : col){
            System.out.print(col_elm + "¥t"); // 列計をすべて表示する
        }
        System.out.println(sum(a)); // 最後に総合計を表示する
    }
    // 行の合計
    public static int[] sumRow(int[][] a){
        int[] sum_row = new int[a.length]; // 行の数＝a.length
        for(int i=0; i<a.length; i++){
            int sum   =  0;
            for(int elm : a[i]){ // i 番目の行配列（a[i]）の要素をすべて合計する
                sum += elm;
            }
            sum_row[i] = sum;
        }
        return sum_row;
    }
    // 列の合計
    public static int[] sumCol(int[][] a){
        int[] sum_col = new int[a[0].length]; // 列の数＝a[0].length
        for(int[] gyo : a){
            for(int j=0; j<sum_col.length; j++){
                sum_col[j] += gyo[j]; // 行配列の各要素 gyo[j]を列計配列 sum_col[j]に加算
            }
        }
        return sum_col;
    }
    // 全体の合計
    public static int   sum(int[][]a){
        int    total = 0;
        int[]  row = sumRow(a); // 行合計を求める
        for(int elm : row){       // 行合計をすべて合計して総合計を求める
            total  += elm;
        }
        return  total;
    }
}
```

16章解答

《解説》
　main メソッドでは、行の合計、列の合計を最初に求めて配列変数 row と col にセットしています。配これを使って、最初の for 文では以下のような、行方向の合計までを表示します。列の処理は、できるだけ拡張 for 文を使うのが原則ですが、行ごとの処理に行番号 i が必要なので、下線のように通常の for 文も合わせて使います。

```
3	1	3	5	12
2	4	5	8	19
7	8	6	1	22
2	9	7	4	22
3	9	4	2	18
```

```
int[] row　　=　　sumRow(a);　　// 行合計
int[] col　　=　　sumCol(a);　　// 列合計
for(int i=0; i<a.length; i++){
　　for(int elm : a[i]){　　// i 番目の行配列（a[i]）の要素をすべて表示する
　　　　System.out.print(elm + "¥t");
　　}
　　System.out.println(row[i]); // 第 i 行の合計を表示する
}
```

　次に最下段の列合計を、配列 col と総合計 sum(a)を使って表示します。

```
3	1	3	5	12
2	4	5	8	19
7	8	6	1	22
2	9	7	4	22
3	9	4	2	18
17	31	25	20	93
```

```
for(int col_elm : col){
　　System.out.print(col_elm + "¥t"); // 列計をすべて表示する
}
System.out.println(sum(a)); // 最後に総合計を表示する
```

行の合計を求める sumRow メソッドでも、同様な理由で下線を引いた部分は普通の for 文を使います。それは行のループ変数として i が必要だからです。行合計を入れる配列 sum_row は i 個の要素を持っているので、各行の合計を sum に求めて、それを sum_row[i]に代入します。

```
// 行の合計
public static int[] sumRow(int[][] a){
　　int[] sum_row = new int[a.length]; // 行の数＝a.length
　　for(int i=0; i<a.length; i++){
　　　　int sum　=　　0;
　　　　for(int elm : a[i]){ // i 番目の行配列（a[i]）の要素をすべて合計する
　　　　　　sum += elm;
　　　　}
　　　　sum_row[i] = sum;
　　}
　　return sum_row;
}
```

列の合計を求める sumCol メソッドでは、拡張 for 文を使って a[][]から 1 行文の配列 gyo[]を取り出します。さらに、下線を引いた箇所のように普通の for 文を使って、gyo[j]を、列合計配列 sum_col[j]に加算して合計を求めています。

16章解答

```
// 列の合計
public static int[] sumCol(int[][] a){
    int[] sum_col = new int[a[0].length]; // 列の数＝a[0].length
    for(int[] gyo : a){
        for(int j=0; j<sum_col.length; j++){
            sum_col[j] += gyo[j]; // 行配列の各要素 gyo[j]を列計配列 sum_col[j]に加算
        }
    }
    return sum_col;
}
```

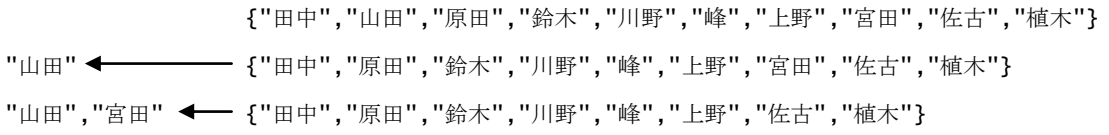
最後に、全体の合計は行の合計を求めるメソッド sumRow()で行合計の配列を求めておいてから、その要素をすべて合計して、総合計としています。

```
// 全体の合計
public   static int sum(int[][]a){
    int        total = 0;
    int[]      row = sumRow(a); // 行合計を求める
    for(int elm : row){             // 行合計をすべて合計して総合計を求める
        total+=  elm;
    }
    return     total;
}
```

16章解答

５． 抽選プログラムを作成します。抽選とは、例えば 10 人から 2 人を選び出すような処理です。次の例では 10 人の名前が入った String の配列から 2 人を抽選しています。

最初に山田さんが選ばれたので、山田さんを除外した残り 9 人の配列を新しく作成し、2 人目はこの中から抽選します。2 回目ではこの中から宮田さんが選ばれました。抽選後は、残り 8 人の配列を新しく作成し直します。



ここで、10 人の中から 1 人を選び出すには、1～10 の間の乱数を作成して決めています。また、9 人から 1 人を選び出すには、1～9 の間の乱数を作成して決めています。つまり、一般に n 人の中から 1 人を選び出すには１～n の範囲の乱数を作成して決めることになります。

以上から次のようなメソッドからなるクラス（プログラム）を作成してください。
なお、delete メソッドは３．の問題で作成した delete メソッド再利用できます。型を int[] から String[]に変更した delete メソッドを作成してください。

A．number メソッド
・上限の値 n（名前配列の要素数）を仮引数として受け取る
・1~ｎの範囲の乱数をひとつ作成する（抽選にあたります）
・作成した乱数を戻り値として返す

B．lot メソッド
・名前の配列を仮引数として受け取る
・配列の要素数を引数にして number メソッドを呼び出し、抽選された要素番号 k を得る
・k-1 を戻り値として返す（k は 1～n の値。配列の要素番号なので 0～n-1 の値にして返す）

C．delete メソッド
・配列と要素番号を仮引数に受け取る
・指定された要素番号の要素だけを含まない新しい配列を作成する
・作成した配列を戻り値として返す

D．main メソッド
・名前の配列 tbl を作成する
{"田中","山田","原田","鈴木","川野","峰","上野","宮田","佐古","植木"}
・2 名を抽選で選ぶので、以下の処理を 2 回繰り返す
・lot メソッドを呼び出して当選番号 k（配列の要素番号）を得る

16章解答

・名前配列 tbl の k 番目を当選者として表示する

・delete メソッドを呼び出して tbl の k 番目の要素を削除した配列を得、その参照を tbl 代入する（古い tbl は消失し、tbl は新しい配列になります）

【解答欄】@20

```
public class P5 {
    public static void main(String[] args) {
        String[] tbl
            = {"田中","山田","原田","鈴木","川野","峰","上野","宮田","佐古","植木"};
        for(int i=0; i<2; i++){
            int k  =  lot(tbl);
            System.out.print(tbl[k] + " ");
            tbl = delete(tbl, k);
        }
    }
    // 配列からひとつの要素を抽選してその要素番号を返す
    public static int lot(String[] tbl){
        int k  =  number(tbl.length);
        return k-1;
    }
    // 1～n の間の整数乱数を返す
    public static int number(int n){
        int r = (int)(Math.random()*n + 1);
        return r;
    }
    // 配列の k 番目の要素を削除した配列を返す
    public static String[] delete(String[] a, int k){
        String[] b = new String[a.length-1];// 一つ少ない要素数で b[]を作成
        int   j=0;    // 配列の要素番号
        for(int i=0; i<a.length; i++){ // a[]の要素数だけ繰り返す
            if(i==k) continue;          // i==k ならスキップする
            b[j++] = a[i];              // コピー
        }
        return b;
    }
}
```

《解説》

delete メソッドは、問題３と同じで構いません。配列が int[]から Stirng[]に変わるだけです。

int r = (int)(Math.random()*n + 1); は１～n までの整数型乱数を作成する時の定石です。