6　入力と出力

1.1 節に示した BMI 算出のサンプルプログラムでは、体重や身長の値はプログラムに直接書き込まれていました。しかしこれだと、他の人の BMI を算出する際には、わざわざプログラムを書き換える必要があります。プログラムの実行開始時、あるいはプログラムの実行途中に、利用者が値をキーボード入力できれば、プログラムはもっと便利なものになるに違いありません。

本章では、キーボードなどを通して利用者が値を入力する仕組みとして、実行時の引数、標準入力、について解説します。

続いて標準入力と対称な関係にある標準出力について解説します。標準出力とは、計算機がディスプレイなどに値を出力する仕組みのことです。

6.1　実行時の引数

1.1 節に示した BMI 算出のサンプルプログラムを、以下のように書き換えてみて下さい。

```
public class Bmi {
   public static void main(String[] args) {
     double weight = Double.parseDouble(args[0]);
     double height = Double.parseDouble(args[1]);
     double bmi = weight / (height * height);
     System.out.println("BMI=" + bmi);
   }
}
```

そしてこれを、いままでと違って、以下のように実行してみて下さい。

```
javac Bmi.java
java Bmi 67.0 1.78
```

どうでしょうか？ 以下のような実行結果が表示されましたでしょうか。

```
BMI=21.146319909102385
```

今回のプログラムでは、Bmi.java ファイルには体重や身長の具体的な値は記入されていません。そのかわりに実行時に、「java Bmi 67.0 1.78」と、体重や身長の値を添えて実行することで、BMI の計算を実現しています。このように、実行時に体重や身長の値を書き添えることができれば、他の人の BMI を算出するために、わざわざプログラムを書き換える必要がないので、1.1 節で示したプログラムより実用的である、と考えることができるでしょう。以下、このプログラムについて説明しましょう。

main メソッドに引き渡される実行時の引数

まず、今までにも使われてきた 2 行目の public static void main(String[] args) ですが、main の後のカッコの中にある「String[] args」も、実は変数の型と名前を宣言しています。String というのは文字列を表すクラスのことです（4.2 節参照）。その後の [] は配列を表します（3.5 節参照）。そして、args が文字列を表す変数の名前になります。

続いて 3 行目と 4 行目ですが、args[0] と args[1] はそれぞれ、配列となっている変数 args の 0 番目と 1 番目の値に相当します。ここで 0 番目および 1 番目の値はそれぞれ、実行時に「java クラス名」の後に書き添えられた文字列、つまり 67.0 および 1.78、ということになります。

文字列から実数への変換

上記の「67.0」および「1.78」という文字列は、あくまでも文字列として扱われていて、この時点で計算機は 67.0 および 1.78 を実数だと思っていません。そこで BMI を計算するためには、これらの文字列を実数に変換します。3 行目の Double.parseDouble(args[0]) は、args[0] に代入されている 67.0 という文字列を、double 型の実数に変換する働きを持っています。4 行目の Double.parseDouble(args[1]) も同様です。[7]

6.2 標準入力

前節で説明した「実行時の引数」は、プログラムの実行開始時に、変数の値を指定できるようにする、というものでした。

しかし実際には、プログラムの実行開始時だけでなく、プログラム実行中の任意のタイミングで変数の値をキーボード入力できると、さらに便利なプログラムを開発することができます。本節では、プログラムの途中でのキーボード入力の方法について説明します。

1.1 節に示した BMI 算出のサンプルプログラムを、BmiKeyboard.java というファイルにコピーして、以下のように書き換えてみて下さい。

```
import java.io.*;

public class BmiKeyboard {
   public static void main(String[] args) {
     double weight = 67.0;
     double height = 1.78;
     try {
       InputStreamReader isr = new InputStreamReader(System.in);
       BufferedReader br = new BufferedReader(isr);
       System.out.println("Please input the weight");
       weight = Double.parseDouble(br.readLine());
       System.out.println("Please input the height");
       height = Double.parseDouble(br.readLine());
       br.close();
     }
     catch (Exception e) {
       System.out.println(e);
     }
     double bmi = weight / (height * height);
     System.out.println("BMI=" + bmi);
   }
```

[7] 参考までに、この Double.parseDouble() は、Double というクラスの中にある parseDouble() というメソッドを呼び出す、ということを意味しています。

}

このプログラムでは、キーボード入力によって体重と身長の値を受けつけ、その値をもとに BMI 値を算出します。なお、多くのオペレーティングシステムでは、キーボードによる文字入力を**標準入力**と呼んでいます。

このプログラムの内容を順に追っていきましょう。

まず 1 行目の import については、7.2 章で後述します。

3,4 行目では、変数 weight および height を宣言し、値を代入しています。

5 行目の try については、7.3 章にて後述します。現時点で注釈しておきたい点として**BufferedReader クラスの readLine() メソッドおよび close() メソッドは、try の後の { と } の間に記述するべきである、**という点をあげておきます。

6 行目の InputStreamReader クラスの変数 isr は、標準入力の情報を文字として扱うために用います。7 行目の BufferedReader クラスの変数 br は、その文字情報を、例えば 1 行単位といったまとまった情報として扱うために用います。

8 行目の System... 行では、Please input the weight という文字列を表示します。これが表示されたら、体重の値をキーボード入力して Enter を押してください。9 行目の br.readLine() というメソッドで、そのキーボード入力した 1 行ぶんの文字情報を抽出します。9 行目の Double.parseDouble() は、その文字情報を double 型の実数に変換します。この値が、変数 weight に代入されます。

10 行目の System... 行では、Please input the height という文字列を表示します。これが表示されたら、身長の値をキーボード入力して Enter を押してください。11 行目の br.readLine() というメソッドで、そのキーボード入力した 1 行ぶんの文字情報を抽出します。11 行目の Double.parseDouble() は、その文字情報を double 型の実数に変換します。この値が、変数 height に代入されます。

12 行目の br.close() は、変数 br による文字情報入力を終了するときに、必ず一度だけ呼び出す必要があります。

14 行目の catch(Exception e) および 15 行目の System.out.println(e) については、7.3 章で後述します。現時点で注釈しておきたい点として、**try の } の直後に必ず catch 文を入れること、**という点をあげておきます。

そして最後に、17 行目で BMI 値を算出して変数 bmi に代入し、18 行目でその値を画面に出力します。

では、このプログラムを実行してみてください。そして、いままで説明したとおりに実行できることを確認してください。

6.3 標準出力

前節で説明した「標準入力」に対称な概念として、**標準出力**があります。多くのオペレーティングシステムでは、ディスプレイ上での文字出力を標準出力と位置づけています。

いままで何度も出てきた System.out.println() は、文字列を標準出力するメソッドです。具体的にいうと、System クラスの中に out 変数があり、out 変数が println メソッドを有している、という関係になります。

標準出力に関する知識は、他にもいろいろあるのですが、本書では割愛します。

6.4 【サンプルプログラム】相性占い

では、標準入力を用いたサンプルプログラムを紹介します。このプログラムは、人物 A の誕生日と、人物 B の誕生日を入力し、この 2 人の相性を 0%から 100%の間の値で算出するものです。[8]

[8] プログラム中のコメントにも書いてあるように、この相性算出式は当てずっぽうですので、算出された値に対して責任を持つことはできませんが、ご了承ください。

このプログラムを Uranai.java というファイル名で保存して、実行してください。そして、以下の順番に誕生日を入力してください。

1. Month of birthday A と表示されたら、人物 A の誕生月を入力する。
2. Day of birthday A と表示されたら、人物 A の誕生日を入力する。
3. Month of birthday B と表示されたら、人物 B の誕生月を入力する。
4. Day of birthday B と表示されたら、人物 B の誕生日を入力する。

例えば、1 月 3 日生まれの人と、4 月 12 日生まれの人の相性を計算するときは、1, 3, 4, 12 の順に 4 つの数字を入力してください。そうすると、0 から 100 の間の数字が出力されます。これが相性をパーセントで表現したものになります。

```
import java.io.*;

// 相性占いのクラス
public class Uranai {

   // 元旦からの合計日数を求める
   static int calculateTotalDays(int month1, int day1) {
     int total = 0;
     int days[] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

     // 誕生月の前の月までの合計日数を求める
     for(int i = 0; i < (month1 - 1); i++) {
       total += days[i];
     }

     // 合計日数に日を足す
     total += day1;

     // 合計日数を返す
     return total;
   }

   // 相性をパーセントで求める
   static int calculateChemistry(int total1, int total2) {
     int chemistry;

     // 相性の算出式 （当てずっぽうな算出式なので信頼できるものではない）
     chemistry = (total1 + total2) % 101;

     // 相性を返す
     return chemistry;
   }
```

```
  // メインメソッド
  public static void main(String[] args) {

    // 2 人の誕生日の月と日を代入する変数
    int month1 = 0, month2 = 0, day1 = 0, day2 = 0;

    // 人物 A の月、日、人物 B の月、日、の順に標準入力する
    try {
      InputStreamReader isr = new InputStreamReader(System.in);
      BufferedReader br = new BufferedReader(isr);
      System.out.println("Month of birthday A");
      month1 = Integer.parseInt(br.readLine());
      System.out.println("Day of birthday A");
      day1 = Integer.parseInt(br.readLine());
      System.out.println("Month of birthday B");
      month2 = Integer.parseInt(br.readLine());
      System.out.println("Day of birthday B");
      day2 = Integer.parseInt(br.readLine());
      br.close();
    }
    catch (Exception e) {
      System.out.println(e);
    }

    // 人物 A の誕生日の元旦からの合計日数を求める
    int total1 = calculateTotalDays(month1, day1);
    // 人物 B の誕生日の元旦からの合計日数を求める
    int total2 = calculateTotalDays(month2, day2);
    // 人物 A と人物 B の相性をパーセントで求める
    int chemistry = calculateChemistry(total1, total2);
    // 人物 A と人物 B の相性を標準出力する
    System.out.println("chemistry=" + chemistry);
  }
}
```

なお、このプログラムで注意すべき新しい点を、いくつか説明します。

初期値を指定する配列の宣言方法

calculateTotalDays メソッドにて、days という配列の変数が使われています。本来なら 3.5 節で説明したとおり、配列の変数を宣言するときには、

```
int days[] = new int[12];
```

というような宣言が必要です。しかし、配列を構成する各々の値がすでに確定しているときには、

```
int days[] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
```

と記載することで、適切な個数の配列を確保し、その各々の値を代入してくれます。

複数の変数の 1 行での宣言

main メソッドにて、以下の 4 つの int 型の変数が宣言されています。

```
int month1 = 0, month2 = 0, day1 = 0, day2 = 0;
```

変数の宣言においては上記のように、複数の変数をカンマ (,) で区切って、1 行につなげて記載することが許されています。

文字列から整数への変換

6.1 節にて「文字列から実数への変換」について説明しましたが、これと同様に「文字列から整数への変換」を施すこともできます。main メソッドには Integer.parseInt(br.readLine()) という処理が 4 回呼び出されていますが、これは br.readLine() によって抽出された文字列を整数に変換するために行われています。

メソッドによる処理の小分け

main メソッドでは、引数を入れ替えて 2 回、calculateTotalDays メソッドを呼び出しています。**このように、複数回にわたって同じ処理をする場合、しかもそれが異なる変数を用いる場合には、その処理部分をメソッドとして分けることで、プログラムを読みやすく、また保守しやすくします。** 特に大きなプログラムを書くときには、積極的にメソッドをつくって処理を小分けにする、ということを心がけましょう。