<< 人にやさしいプログラミングの哲学　(第2版)/構造化プログラミング編/人にやさしいプログラムの書法/人にやさしいプログラムの書法 >>

1.2 人にやさしいプログラムの書法

　次のプログラムを読んでみましょう。このプログラムがどのようなプログラムかが分かるでしょうか。 答えは大方の人がNoだと思います。（分かったとしても相当時間がかかったのではないでしょうか。）

　これからこのプログラムを、人にやさしいプログラムに直していきましょう。

リスト 1.2.1 例題プログラム

```
 1: public class Example {
 2:
 3:   public static voic main(String[] args) {
 4:           Example example = new Example();
 5:           example.main();
 6:   }
 7:
 8: voic main(){
 9: int a = 49;
10: int b = 73;
11: int c = 100;
12: int d = 45;
13: int e = 25;
14: double x = a + b + c + d + e;
15: double y = x / 5.0;
16: y = y * 10;
17: if((y % 10) >= 5){
18: y = y + 10;
19: }
20: int z = (int)(y / 10);
21: System.out.println(z);
22: }
23: }
```

オマジナイ

　Javaはオブジェクト指向言語であるがゆえに、 どんなに簡単なプログラムでも、初心者にとって理解するのが難しいいくつかの記述が必要です。

そんな"オマジナイ"をここで確認しておきましょう。

クラス宣言

Javaのプログラムは全て「クラス」と呼ばれるプログラムの単位の中に記述されます。（詳しくはオブジェクト指向編で） ですから、どのような小さなプログラムでも必ず一つクラスを作る必要があります。

クラスの記法は次のとおりです。クラス名は自分で決められますが、他はオマジナイです。 中括弧「{}」で囲まれたブロックの中にメソッドなどのプログラムの要素が記述されます。

```
public class [クラス名]{
        (ここにプログラムの要素を記述する)
}
```

また、Javaのプログラムではファイル名に気をつける必要があります。 ファイル名は必ず**クラス名に「.java」をつけたもの**にしなければなりません。

例えば、この例ですと、ファイル名は「Sample.java」にする必要があります。

public static void main(String args[])メソッド

クラスの中に、中括弧で囲まれた記述が２つあります。これをメソッドといいます。 クラスの中には、いくつでもメソッドを記述することができますが、第4章でメソッドを詳しく学習するまで、 2つのメソッドでプログラムを記述します。

このメソッド記述は全てオマジナイです。次のような書式で記述してください。

```
public static void main(String[] args){
                [クラス名] [クラス名の先頭を小文字にした変数] = new [クラス名]();
                [クラス名の先頭を小文字にした変数].main();
}
```

void main()メソッド

2つのメソッドのうち、こちらがアプリケーションプログラムを記述するメソッドです。 プログラムはmain()メソッドから始まる (1) ことを覚えておきましょう。 書式は次のとおりで、中括弧の中にアプリケーションプログラム(行いたい仕事)を記述します。

```
void main(){
                (ここにアプリケーションプログラムを記述する)
}
```

1. 本当はpublic static void main(String[] args)メソッドからプログラムが始まります。今回はpublic static void main(String[] args)からmain()メソッドを呼び出し（メソッドに関しては4章以降を参照）ています。

1.2.1 意味のまとまりを明らかにする

ただプログラムの処理がだらだらと書かれているソースコードは、人にやさしくありません。 処理の意味を考えて、いくつかのまとまりを作ると人間にとってわかりやすくなります。

プログラムにおいて、処理のまとまりのことをブロックと呼びます。 ブロックが分かりやすいよう

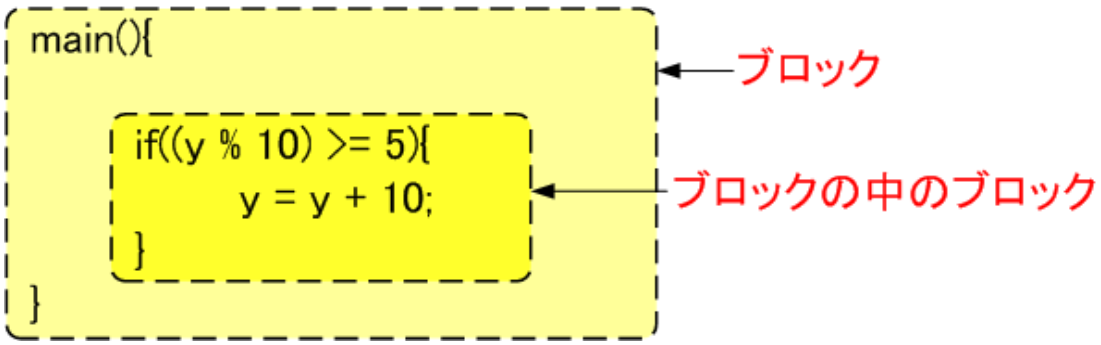
に記述するには、インデントと空行を効果的に使うことが重要です。 例をリスト[例題プログラム（ブロックを意識した）]に示します。

リスト 1.2.1.1 例題プログラム（ブロックを意識した）

```
 1: public class Example {
 2:
 3:   public static voic main(String[] args) {
 4:           Example example = new Example();
 5:           example.main();
 6:   }
 7:
 8:   voic main() {
 9:
10:           int a = 49;
11:           int b = 73;
12:           int c = 100;
13:           int d = 45;
14:           int e = 25;
15:
16:           double x = a + b + c + d + e;
17:           double y = x / 5.0;
18:
19:           y = y * 10;
20:           if ((y % 10) >= 5) {
21:                   y = y + 10;
22:           }
23:           int z = (int) (y / 10);
24:
25:           System.out.println(z);
26:   }
27: }
```

ブロック

プログラムにおいて、処理のまとまりのことをブロック(図[ブロック])と呼び、Java言語では、中括弧「{}」で囲って表現します。（クラスやメソッドもブロックの一種です。）Javaのプログラムはブロックの入れ子で構成されています。



インデント

インデント(図[インデント])とは、字下げをすることです。字下げにはTABやスペースを使用します。（TABなら1つ分、スペースなら2つ分を目安とするとよいでしょう）字下げをすることによって、入れ子になったブロックの関係をはっきり表すことができます。

```
main(){

    if((y % 10) >= 5){
        y = y + 10;
    }
}
```

空行で表現されたブロック

中括弧で囲まれたブロックだけでなく、空行を入れることにより、ブロックの中に意味のまとまりを作って、プログラムをより見やすくします。(図[空行によるブロック])

```
double x = a + b + c + d + e;
double y = x / 5.0;

y = y * 10;
if ((y % 10) >= 5) {
        y = y + 10;
}
int z = (int) (y / 10);

System.out.println(z);
```

1.2.2 変数の目的を明らかにする

Java言語では、クラスや変数、メソッドの名前は自由につけられます。 では「a」や「bbb」のような名前がついていた場合、他の人が果たしてその意味を理解してくれるでしょうか。 人に分かりやすいソースコードを書くためには、 クラスや変数の名前はその意味が分かるような名前にする必要があります。 クラスや変数に名前をつけるときは、子供に名前を付ける気持ちで望みましょう。 クラスや変数、メソッドについて後ほど詳しく説明します。

リスト 1.2.2.1 成績管理プログラム（適切な変数名をつけた）

```
 1: public class ScoreAdministratorSample {
 2:
 3:   public static voic main(String[] args) {
 4:           ScoreAdministratorSample scoreAdministratorSample =
 5:                   new ScoreAdministratorSample();
 6:           scoreAdministratorSample.main();
 7:   }
 8:
 9:   voic main() {
10:
11:           int japanese = 49;
12:           int mathematics = 73;
13:           int science = 100;
14:           int civics = 45;
15:           int english = 25;
16:
17:           double total = japanese + mathematics + science + civics + english;
18:           double average = total / 5.0;
19:
20:           average = average * 10;
```

```
21:             if ((average % 10) >= 5) {
22:                     average = average + 10;
23:             }
24:             int result = (int) (average / 10);
25:
26:             System.out.println(result);
27:     }
28: }
```

1.2.2.1 Javaの命名規則

Java言語を記述するときの一般的な命名規則 (2) の説明をします。 本テキストのソースコードはすべてこの規則に従っています。

- **クラス名**
  - 最初の文字は大文字に
  - 複数の単語の時は各単語の最初の文字を大文字に

  例えば、旅券予約アプリケーションには、「TicketReserveApplication」 となります。クラス名とファイル名は同じにしなければならないので、この場合ファイル名は「TicketReserveApplication.java」です。
- **メソッド名**
  - 最初の文字は小文字に
  - 複数の単語の時は各単語の最初の文字を大文字に

  メソッドはある処理を行うものなので大抵は動詞から始めます。 例えば、現在の時刻を調べるメソッドは、「getCurrentTime」 といったメソッド名になります。
- **変数名**
  - 最初の文字は小文字に
  - 複数の単語の時は各単語の最初の文字を大文字に

  変数名は、メソッド名と同様の規則です。 例えば、苗字を記憶する変数名は、「familyName」といった変数名になります。

2. クラス名やメソッド名、変数名に関しては、数字が先頭にくるもの、予約語、演算子を含むものは使用できません。

1.2.3 プログラムの目的を明らかにする

1.2.3.1 コメントをつける

ソースコードの中にコメントを書き込むことができます。このコメントはプログラムの処理には全く影響しません。 コメントは内容が重要です。内容については次項で詳しく議論します。 とりあえずコメントをつけたプログラムをリスト[成績管理プログラム（とりあえずのコメントを付けたもの）]に示します。

リスト 1.2.3.1.1 成績管理プログラム（とりあえずのコメントを付けたもの）

```
1: public class ScoreAdministratorSample {
2:
3:   public static voic main(String[] args) {
4:           ScoreAdministratorSample scoreAdministratorSample =
5:                   new ScoreAdministratorSample();
6:           scoreAdministratorSample.main();
```

```
 7:    }
 8:
 9:    voic main() {
10:            //変数を宣言し値を代入する
11:            int japanese = 49; //整数型japaneseという変数
12:            int mathematics = 73; //整数型mathematicsという変数
13:            int science = 100; //整数型scienceという変数
14:            int civics = 45; //整数型civicsという変数
15:            int english = 25; //整数型englishという変数
16:
17:            //合計を保存しておく変数totalを5で割り、変数averageに代入する
18:            double total = japanese + mathematics + science + civics + english;
19:            double average = total / 5.0;
20:
21:            //averageを10倍し，10で割ったあまりを調べる
22:            average = average * 10;
23:            if ((average % 10) >= 5) { //余りが5以上なら
24:                    average = average + 10; //10を加える
25:            }
26:            //結果をresultに代入する
27:            int result = (int) (average / 10);
28:
29:            //変数resultの値を表示する
30:            System.out.println(result);
31:    }
32: }
```

1.2.3.2 Javaでのコメントの記述方法

コメントの記述方法を間違えると、コンパイルエラーになるので注意しましょう。

範囲指定コメント

プログラム中、「/*」 から 「*/」までのすべての文字は、コメントとして扱われます。複数行でも構いません。

```
/* この中がコメント */
```

Javaの慣習として（Javadocというドキュメントを自動生成するため）始まりは「/**」とし、 複数行にわたる場合は、行の始めに「*」をつけます。

```
/**
 * Javaでの標準形式
 */
```

行コメント

プログラム中、「//」からその行の最後までのすべての文字は、コメントとして扱われます。 次の行までの効果はありません。

```
int x;//この行のここから先はすべてコメント
```

先ほどの(リスト[成績管理プログラム（とりあえずのコメントを付けたもの）])を読んでみてこのプログラムが何をするものであるのか考えてみましょう

1.2.3.3 プログラムの目的を明らかにするコメント

コメントはただ書けばよいというものではありません。

具体的に見てみましょう。次のコードは先のリスト[成績管理プログラム（とりあえずのコメントを付けたもの）]から抜き出したものです。

```
//averageを10倍し，10で割ったあまりを調べる
average = average * 10;
if ((average % 10) >= 5) { //余りが5以上なら
        average = average + 10; //10を加える
}
```

このコメントを次のものと比べてみましょう。

```
//平均を四捨五入する
average = average * 10;
if ((average % 10) >= 5) { //１の位が５以上なら
        average = average + 10; //繰り上げる
}
int result = (int) (average / 10);
```

人にやさしいコメントをつけたプログラムをリスト[成績管理プログラム（人にやさしいコメント付）]に示します。

リスト 1.2.3.3.1 成績管理プログラム（人にやさしいコメント付）

```
 1: /**
 2:  * ○○中学校の成績管理プログラム
 3:  *
 4:  * 五教科（国語・数学・理科・公民・英語）の平均（四捨五入済み）を求める
 5:  *
 6:  * @author Manabu Sugiura
 7:  * @version $Id: ScoreAdministratorSample.java,v 1.1 2004/03/26 05:41:21 duskin Exp $
 8:  */
 9: public class ScoreAdministratorSample {
10:
11:   public static void main(String[] args) {
12:         ScoreAdministratorSample scoreAdministratorSample =
13:                 new ScoreAdministratorSample();
14:         scoreAdministratorSample.main();
15:   }
16:
17:   void main() {
18:
19:         //各教科の点数を設定する
20:         int japanese = 49; //国語
21:         int mathematics = 73; //数学
22:         int science = 100; //理科
23:         int civics = 45; //公民
24:         int english = 25; //英語
25:
26:         //５教科の合計点を求める
27:         double total = japanese + mathematics + science + civics + english;
28:         //５教科の平均を求める
29:         double average = total / 5.0; //平均を計算する
30:
31:         //平均を四捨五入する
```

```
32:             average = average * 10;
33:             if ((average % 10) >= 5) { //１の位が５以上なら
34:                     average = average + 10; //繰り上げる
35:             }
36:             int result = (int) (average / 10);
37:
38:             //四捨五入した平均を表示する
39:             System.out.println(result);
40:     }
41: }
```

1.2.3.4 コメントの種類

コメントは書く位置によって役割が異なります。次の3種類を目安にすると分かりやすいでしょう。

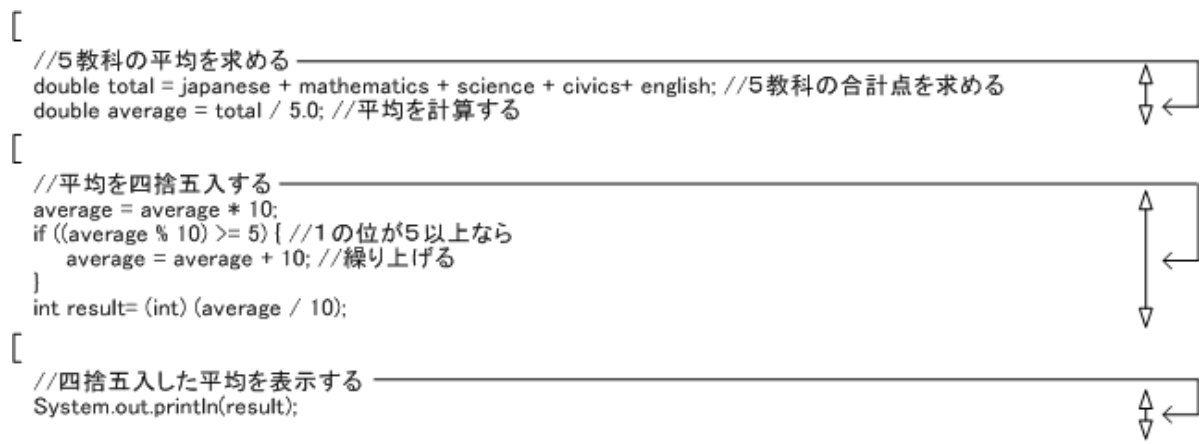
- **タイトル（見出し）コメント**
  タイトルコメントは、プログラムの一番初めに書くコメントで、 そのプログラムが何を目的としたものであるかを簡潔に表記します。 タイトルコメントの要素として以下のものが挙げられます。

  標題(タイトルとプログラム目的)
  プログラムの名称と、その機能を簡潔に表したもの。

  ファイル名
  そのプログラム自身のファイル名。

  作者
  プログラムを作成した人の名前。設計した人が別の人ならば、設計者の名前を書いておきます。後からそのプログラムを変更する人が、誰に尋ねれば情報を得られるのかを簡単に知ることができます。複数の人でプログラムを書くときに役に立ちます。

  バージョンアップの履歴
  改造や修正によっていったん出来上がったプログラムが変更されると、そのプログラムのバージョンが変わることになります。このバージョンアップが行われた日付と、行った人の名前を記入しておくと、誰によっていつ変更されたものなのかが分かります。
  このテキストの例では、プログラム開発環境の機能を利用して自動的に履歴を入れたものになっています。
- **ブロックコメント**
  対象ブロックが何を目的としたプログラムなのかを記述するコメントです。（ブロックが始まる前に書く） ブロック全体に通用します。(図[ブロックコメントの有効範囲])



- **行コメント**
  その行は何を目的として書かれているのかを行末に書くコメントです。 その行だけに通用する小

目的を書きます。

<< 人にやさしいプログラミングの哲学　(第2版)/構造化プログラミング編/人にやさしいプログラムの書法/人にやさしいプログラムの書法 >>