

# 情報処理技法 (Javaプログラミング)1

第10回  
よりよいプログラムを書くには?

人間科学科コミュニケーション専攻  
白銀 純子

Copyright (C) Junko Shiragane, Tokyo Woman's Christian University 2016. All rights reserved.

## 第10回の内容

- コメント
- インデント
- 命名規則
- 数値の扱い

Copyright (C) Junko Shiragane, Tokyo Woman's Christian University 2016. All rights reserved.

## 前回の復習問題の解答

- for文、while文、do～while文の違いについて、使い方や処理内容の面から解答しなさい。

解答例

- for文: 処理を繰り返す回数が決まっている場合によく利用される。
- while文: 処理を繰り返す回数が実行時にならなければわからない場合によく利用される。ただし、処理内容が1度も実行されないこともある。
- do～while文: while文と同様、処理を繰り返す回数が実行時にならなければわからない場合によく利用される。ただし、処理内容は必ず1度は実行される。

Copyright (C) Junko Shiragane, Tokyo Woman's Christian University 2016. All rights reserved.

## コメント

Copyright (C) Junko Shiragane, Tokyo Woman's Christian University 2016. All rights reserved.

## プログラム中に書けるのは処理だけ?

- プログラムが長くなると、それぞれの場所で何をしているかわからなくなる
  - ◇ たとえ自分が作ったプログラムであっても...
- 企業でソフトウェアを作る人は、何人かのチームを作って1つのソフトウェアを作る

➡ 他の人が書いたプログラムを見て理解しなければならないことも...  
処理内容だけ見て理解できる??

プログラムに「コメント」を書く

Copyright (C) Junko Shiragane, Tokyo Woman's Christian University 2016. All rights reserved.

## コメントって?

- プログラム中に書く、プログラムの説明

◇ それぞれの変数は何のための変数か

int average, result;

- average: 何の平均?
- result: 何の結果?

◇ それぞれの文は何をするための文か

if (result < 10) ...

- 何をするif文?
- 条件は何を判定している?

◇ そもそもこのプログラムは何のためのプログラムか

コメントには何を書いてもOK

Copyright (C) Junko Shiragane, Tokyo Woman's Christian University 2016. All rights reserved.

## コメントの書き方(その1)

- 「//」を使って書く

◇「//」以降、その行の終わりまでがコメントになる

```
//average: 0から4までの数の平均
//sum: 0から4までを足し合わせた結果
int i, average, sum;
sum = 0; //sumを初期化する
for (i = 0; i < 5; i++) {
    sum = sum + i; //0から4までの数を足し合わせる処理
}
//平均を求める
average = sum / 5;
```

コメント部分

Copyright (C) Junho Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

## コメントの書き方(その2)

- 「/\* ... \*/」を使って書く

◇「/\*」以降、「\*/」までがコメントになる

```
/*average: 0から4までの数の平均
sum: 0から4までを足し合わせた結果*/
int i, average, sum;
sum = 0; /*sumを初期化する*/
for (i = 0; i < 5; i++) {
    sum = sum + i; /*0から4までの数を足し合わせる処理*/
}
/*平均を求める*/
average = sum / 5;
```

コメント部分

Copyright (C) Junho Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

## コメントの書き方(その2)の注意

- 一見カッコの関係(カッコは内側から閉じる関係)に見えるが、カッコの関係とは別

```
/*average: 0から4までの数の平均
sum: 0から4までを足し合わせた結果*/
int i, average, sum;
sum = 0; /*sumを初期化する*/
```

コメントの「開始」と「終了」の対応

コメントとはみなされない  
→ コンパイラエラーに

Copyright (C) Junho Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

## コメントの使い方(1)

- 変数の意味について、その変数を宣言している文の直前または直後などに書く
- 処理内容について、その処理を表す文の直前または直後に書く
- プログラムが何であるかについて、ファイルの先頭に書く
- etc.

コメントを書くことは、  
わかりやすいプログラムを書くために重要!

Copyright (C) Junho Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

## コメントの使い方(2)

- プログラムをコンパイルできないとき
- コンパイルはできても処理結果が間違っているとき

➡ プログラム中の間違っている部分をコメントにしてコンパイル・実行してみることで、間違っている部分を探すこともよくある

```
for (i = 0; i < 5; i++) {
    /* for (j = 0; j < 5; j++) {
    } */
}
```

```
for (i = 0; i < 5; i++) {
    // for (j = 0; j < 5; j++) {
    // }
}
```

「コメントアウト」と呼ぶ(プログラムの説明をするのではなく、プログラムの処理部分をコメントにすること)

Copyright (C) Junho Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

## インデント

Copyright (C) Junho Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

## カッコの対応関係

- if文やfor文、while文が入れ子になっている文では、カッコの対応関係はわかりにくい

```
for (i = 0; i < 5; i++) {
  if (i % 2 == 0) {
    for (j = 0; j < 5; j++) {
    }
  } else if {
  }
}
```

```
for (i = 0; i < 5; i++) {
  if (i % 2 == 0) {
    for (j = 0; j < 5; j++) {
    } else if {
    }
  }
}
```

どの開きカッコがどの閉じカッコに対応している??

「インデント」をする

Copyright (C) Junho Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

## インデントって?

- インデント = 字下げ

- 開きカッコのある文の開始位置と閉じカッコの位置をあわせる
- カッコの内側にある文は、開始位置の前にスペースを入れて、外側の文よりも後ろに書く

※プログラムでは、文と文の区切り、単語と単語の区切りにはスペースや改行をいづつ入れてもOK

Copyright (C) Junho Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

## インデントをすると...

- カッコの対応関係が少しわかりやすくなる

2文字ずつインデントしている例  
(カッコの内側にある文は、すぐ外側の文よりも2文字分後ろに下がっている)

```
for (i = 0; i < 5; i++) {
  if (i % 2 == 0) {
    for (j = 0; j < 5; j++) {
    }
  } else if {
  }
}
```

2文字 2文字

- 開始カッコの後ろにはコメント以外のものは書かないこと
- 1つの閉じカッコに1行使うこと(1行に2つも3つも閉じカッコを書かない)

Copyright (C) Junho Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

## Javaの命名規則

## チームでプログラムを作ろうとすると...

- 他人が自分のプログラムを読んで理解する
- 自分が他人のプログラムを読んで理解する

プログラムの読みやすさが重要!!

読みやすいプログラムを作るには?

- コメントをたくさん入れて、プログラムの各所で何をしているかをわかりやすくする
- インデントをする
- 個々の単語が何を意味しているか、わかりやすくする
  - クラス名や変数の名前として、意味を理解しやすいものをつける
  - ルールに従って名前をつける

Javaの命名規則

Copyright (C) Junho Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

## Javaの命名規則

- クラスや変数、メソッドなどに名前をつけるときの、Javaでのルール

- 守らなくてもコンパイルエラーなどのエラーになるものではない
- 広く普及しているので、慣れて従うことが望ましい
  - ルールに従うことで、プログラム内の名前を見ると、それがクラス名・変数名・メソッド名・etc.なのかがわかりやすくなる

Copyright (C) Junho Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

## クラスの命名規則

- 先頭の文字をアルファベットの大文字とする
  - ◇ 2文字目以降は小文字
- 複数の単語を連結する場合は、2つ目以降の単語の先頭を、アルファベットの大文字とする
  - ◇ 2文字目以降は小文字

Ex1. 「student」という名前をクラス名にしたい

→ クラス名: **S**tudent  
大文字 小文字

Ex2. 「student」、「mark」、「management」という名前を連結してクラス名にしたい

→ クラス名: **S**tudent**M**ark**M**anagement  
大文字 小文字

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

19

## 変数の命名規則

- 先頭の文字をアルファベットの小文字とする
  - ◇ 2文字目以降も小文字
- 複数の単語を連結する場合は、2つ目以降の単語の先頭を、アルファベットの小文字とする
  - ◇ 2文字目以降は小文字

Ex1. 「english」という名前を変数名にしたい

→ 変数名: **e**nglish  
小文字 小文字

Ex2. 「english」、「mark」という名前を連結して変数名にしたい

→ 変数名: **e**nglish**m**ark  
小文字 小文字

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

20

## 数の桁数

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

21

## int型で扱える数

- -2147483648 ~ 2147483647

2147483647より1つ大きい数は???

→ 2147483647 + 1 = -2147483648

コンピュータが扱える数の限界を超えたために起こる現象

桁あふれ  
(オーバーフロー)と呼ぶ

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

27

## 参考(int以外の整数のデータ型)

- int型よりもっと多くの桁数を扱うには?  
→ 「long」型: -9223372036854775808 ~ 9223372036854775807
- int型よりも少ない桁数を扱うには?  
→ 「short」型: -32768 ~ 32767 (2進数で16桁)

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

29

## 小数

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

30

## 小数の扱い(1)

- 小数部分が無限のものを扱うわけではない
  - ◇ 例えば割り算で割り切れない数や円周率

➡ 小数部分を適当なところで切り捨てる(四捨五入ではない)

例えば... $1 \div 7$ :  
コンピュータは「0.142857...142」のように考える  
本来はこの後も無限に続く

Copyright (C) Junho Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

31

## 小数の扱い(2)

- 計算結果が小数のものをint型の変数に代入すると...

➡ 小数部分を切り捨てる(四捨五入ではない)

例えば...  

```
int result;
int first = 18, second = 10;
result = first / second;
```

  
➢ 本来の「first / second」の計算結果は1.8  
➢ でもresultの値は「1」

Copyright (C) Junho Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

32

## 小数の扱い(3)

- 小数部分が無限のものは適当なところまで切り捨てられる
- 小数をint型に代入すると、小数部分が切り捨てられる

本来の数よりも、小数の桁数が小さくなってしまう現象

桁落ちと呼ぶ

Copyright (C) Junho Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

33

## 桁落ちの例(間違いやすいもの)

```
int i, average = 0;
int[] score = {98, 79, 44, 87, 64, 49, 51, 68, 93, 77};
for (i = 0; i < 10; i = i + 1) {
    average = average + score[i] / 10;
}
```

この部分の数は...

iが0のとき: 9, iが1のとき: 7, iが2のとき: 4, iが3のとき: 8, iが4のとき: 6, iが5のとき: 4,  
iが6のとき: 5, iが7のとき: 6, iが8のとき: 9, iが9のとき: 7

➡  $9 + 7 + 4 + 8 + 6 + 4 + 5 + 6 + 9 + 7 = 65$

桁落ち

人間が計算すると...

$9.8 + 7.9 + 4.4 + 8.7 + 6.4 + 4.9 + 5.1 + 6.8 + 9.3 + 7.7 = 71$

Copyright (C) Junho Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

34

## 桁落ちが起こると...

- 数が本来の数よりも小さくなってしまふ
  - ◇ 微妙な数値が必要な場合には要注意
- 桁落ちをした数に大きな数をかけると、本来の数に大きな数をかけたときとの差が大きくなる

例えば... $1 \div 7$ の結果が桁落ちし、0.142になる」とすると

- 0.142に100000をかけると14200
- $1 \times 100000$ を7で割ると(計算の順序を変えると)14285.714
- 本来の $1/7$ に100000をかけると、14285.714285.....

➡ コンピュータで計算をするときは、計算の順番に注意  
(割り算はなるべく後にすること)

➢ 例えば「 $1 \div 7 \times 100000$ 」の計算は、「 $1 \times 100000$ 」をしてから7で割る

Copyright (C) Junho Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

35

## 平均の計算での桁落ちの防止

- 平均の計算の手順(その1)

1. 数の合計を計算する
2. 1.の合計を、数の個数で割り算する

コンピュータにさせてもOK

- 平均の計算の手順(その2)

- ◇ 各数を、数の個数で割ったものを足していく

コンピュータにさせるのはNG(人間がするのはOK)

```
int i, sum = 0, average;
int[] score = {98, 79, 44, 87, 64, 49, 51, 68, 93, 77};
for (i = 0; i < 10; i = i + 1) {
    sum = sum + score[i] / 10;
}
average = sum / 10;
```

合計をしてから割り算

Copyright (C) Junho Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

36

## データ型の変換

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

## 数値のデータ型の変換(1)

- 小数のデータ型(float, double型)の変数の値をint型に代入したい

```
float data;
int num;
data = 3.0;
num = data;
```

コンパイルエラー(精度が落ちている可能性)

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

## 数値のデータ型の変換(2)

- int型の変数同士の計算結果を小数のデータ型(float, double型)に代入したい

```
double data;
int first, second;
first = 5;
second = 2;
data = first / second;
```

桁落ち(「data」の値は「2.0」に)

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

## 数値のデータ型の変換(3)

- 小数のデータ型の変数の値をint型に代入してもコンパイルエラーを起こさないためには?
  - ◇ ただし、桁落ちは起こってもかまわない場合
- int型の変数同士の計算結果を小数のデータ型に代入しても桁落ちを起こさないためには?
- その他
  - ◇ int型の変数の値をlong型に代入したい

「キャスト」を利用

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

## キャストって?

- あるデータ型の値を、別のデータ型に変換すること
- 変数の前に「(変換したいデータ型)」を書く

```
first: int型(「(double)」をつけることで、double型として扱われる
    > 割り算の場合は、割られる数の変換が必要
    (割る数の変換はしなくても良い)
```

例えば...  
int型の計算結果をdouble型に代入したい  
➢ result = (double) first / second;

double型の値をint型に代入したい

```
➢ num = (int) data;
data: double型(「(int)」をつけることで、
    int型として扱われる
```

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

## 変換できるのは数値のデータ型のみ

- 値が数のみであっても、データ型がString型の場合には変換できない

```
int num;
String str;
str = "50";
num = (int) str;

int num;
String str;
num = 50;
str = (String) num;
```

コンパイルエラー

※現状の内容の中では、変換できるものは数値のデータ型のみ

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

## 次回

● 課題の質問受け付け

◇ 復習問題の解答はなし