

情報処理技法

Javaプログラミング①

第14回

ある1セットの命令をあちこちで利用するには?

人間科学科コミュニケーション専攻

白銀 純子

Copyright © Junko Shiragane, Tokyo Woman's Christian University 2015. All rights reserved.

第14回の内容

- メソッドの作り方・使い方

Copyright © Junko Shiragane, Tokyo Woman's Christian University 2015. All rights reserved.

前回の復習問題の解答

- 「例外」とは何かを説明しなさい。

解答例:

プログラム実行時に発生するエラーのこと。例外が発生することにより、プログラムの実行がその時点で終了してしまう。

Copyright © Junko Shiragane, Tokyo Woman's Christian University 2015. All rights reserved.

メソッドの作り方・使い方

Copyright © Junko Shiragane, Tokyo Woman's Christian University 2015. All rights reserved.

同じ内容の処理って?(p. 290)

- 高校の生徒の英、数、国、理、社の平均点を計算するとき

```
英語の平均 { for (i = 0; i < 50; i++) {  
              englishTotal = englishTotal + english[i];  
            }  
            englishHeikin = englishTotal / 50;  
数学の平均 { for (i = 0; i < 50; i++) {  
              mathTotal = mathTotal + math[i];  
            }  
            mathHeikin = mathTotal / 50;  
            国語の平均, 理科の平均, 社会の平均 ...
```

Copyright © Junko Shiragane, Tokyo Woman's Christian University 2015. All rights reserved.

変数の名前が違っただけ?(p. 290)

- 変数の名前が違っただけで、やっていることは同じ
 - 英語の点数の合計を求めて平均する
 - 数学の点数の合計を求めて平均する
 -
- 変数の名前が違うから、for文やwhile文は使えない
- でも、変数の名前を同じにはできない

「メソッド」を使う

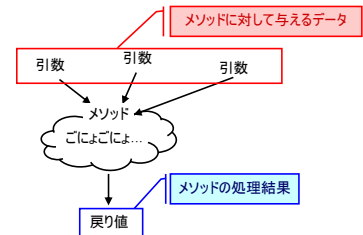
Copyright © Junko Shiragane, Tokyo Woman's Christian University 2015. All rights reserved.

メソッド(p. 291)

- プログラム中で行われる処理の手順をまとめたもの
 - 複数の処理をまとめて、1つの名前を付けたもの
- メソッド名、引数、戻り値(返り値)という構成
 - **メソッド名**: メソッドの名前
 - **引数**: メソッドに渡す情報(計算等の処理の材料にするデータ)
 - 処理の材料にするデータのみ、引数として定義
 - **戻り値(返り値)**: メソッドの内容を実行したときの処理結果
 - 多くの場合、戻り値を変数に代入して利用する
 - 「**変数名 = メソッド**」で、変数に戻り値が代入される

Copyright © Jinko Shingane, Tokyo Woman's Christian University 2015. All rights reserved.

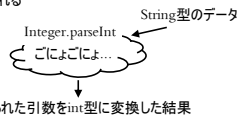
メソッドのイメージ(p. 291)



Copyright © Jinko Shingane, Tokyo Woman's Christian University 2015. All rights reserved.

メソッド(例)(1)(p. 291)

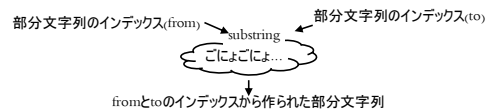
- Integer.parseInt(str)
 - **メソッド名**: Integer.parseInt
 - **引数**: str(String型)
 - **戻り値**: 「str」をint型に変換したデータ
 - 戻り値をint型の変数に代入して利用する
- ➡ 「num = Integer.parseInt(str)」で、変数「num」に、文字列をint型のデータに変換したものが代入される



Copyright © Jinko Shingane, Tokyo Woman's Christian University 2015. All rights reserved.

メソッド(例)(2)(p. 291)

- str.substring(m, n)
 - **メソッド名**: substring
 - **引数**: m(int型)とn(int型)
 - **戻り値**: 「str」のm番目からn番目までの文字で作った部分文字列
 - 戻り値をString型の変数に代入して利用する
- ➡ 「subStr = str.substring(m, n)」で、変数「subStr」に、strの部分文字列が代入される



Copyright © Jinko Shingane, Tokyo Woman's Christian University 2015. All rights reserved.

メソッドを作る(1)(p. 293)

- 今回定義するのは **ごにょごにょ...** の部分

メソッドを作るときのお約束

```
public static 戻り値のデータ型 メソッド名(引数) {
    メソッドでの処理内容
    return 処理結果;
}
```

Copyright © Jinko Shingane, Tokyo Woman's Christian University 2015. All rights reserved.

アクセス修飾子(p. 293)

- Javaは、様々なクラスを連携させて動作するもの
- **アクセス修飾子**: クラス同士を連携させたときに、他のクラスから利用できるかどうかを示すもの
 - public: 他のクラスから利用可能
 - ← 今回は、クラスは1つだけなのでpublicでOK
 - protected: 限られた範囲で、他のクラスから利用可能
 - private: 他のクラスからは利用不可

Copyright © Jinko Shingane, Tokyo Woman's Christian University 2015. All rights reserved.

static(1)(p. 293)

- 「static」がついているメソッド(クラスメソッド)
 - 「クラス名.メソッド」の形式で呼び出し可能
 - static付きのメソッド内部で、同じクラスで定義されているメソッドを呼び出すときは、そのメソッドにもstaticが必要
 - mainメソッドから呼び出すときは、「static」がついている必要
- Ex. Integer.parseInt(...)
- Integer: Javaで用意されているクラスの1つ
 - parseInt: Integerクラスで定義されているメソッド

Copyright © Jinko Shingane, Tokyo Woman's Christian University 2015. All rights reserved.

static(2)(p. 293)

- 「static」がついていないメソッド(インスタンスメソッド)
 - 必ず「オブジェクト名.メソッド」の形式で呼び出し
 - 「クラス名.メソッド」の形式では呼び出し不可能
- Ex. BufferedReader br = new BufferedReader(...)
- BufferedReader: Javaで用意されているクラスの1つ
 - br.readLine()という形でメソッドを利用
 - readLine: BufferedReaderクラスで定義されているメソッド

Copyright © Jinko Shingane, Tokyo Woman's Christian University 2015. All rights reserved.

static(3)(p. 293)

- 「static」をつけるかつかないかは、「オブジェクト指向」というプログラミングの方法をきちんと勉強する必要
 - 今回は、mainメソッドの中でメソッドを利用
 - mainメソッドは、クラスメソッドの一種
 - = static付きのメソッド
- ➡ メソッドの宣言に「static」をつける必要

Copyright © Jinko Shingane, Tokyo Woman's Christian University 2015. All rights reserved.

メソッドを作る(2)(p. 293)

- 今回定義するのは「ごにょごにょ...」の部分

戻り値は1つだけ

```
public static 戻り値のデータ型 メソッド名(引数) {
    メソッドでの処理内容
    return 処理結果;
}
```

Copyright © Jinko Shingane, Tokyo Woman's Christian University 2015. All rights reserved.

メソッドを作る(3)(p. 294)

メソッドの名前(名前の付け方は、クラス名や変数名と同じ)

```
public static 戻り値のデータ型 メソッド名(引数) {
    メソッドでの処理内容
    return 処理結果;
}
```

Javaの命名規則としては...

- 先頭の単語は動詞にする
- 複数の単語を連結するときは、先頭の単語はすべて小文字、2つ目以降の単語は先頭の文字のみ大文字、あとは小文字
- 変数と同じ

Copyright © Jinko Shingane, Tokyo Woman's Christian University 2015. All rights reserved.

メソッドを作る(4)(p. 295)

- 「引数のデータ型 引数の変数名」と書く
- 引数はいくつあっても良い「」で区切る
- データ型はそれぞれ異なってもかまわない

```
public static 戻り値のデータ型 メソッド名(引数) {
    メソッドでの処理内容
    return 処理結果;
}
```

Copyright © Jinko Shingane, Tokyo Woman's Christian University 2015. All rights reserved.

引数の定義の注意

- 引数の定義は、一見変数宣言のよう
- 実際、メソッド内で使う変数の宣言とも言える

but...
通常の変数宣言とは違う:
➤ 必ず「データ型 変数名」のセットで書かなければならない

```
public int sum(int num1, int num2, int num3) {
    int result;
    result = num1 + num2 + num3;
    return result;
}
```

```
public int sum(int num1, num2, num3) {
    int result;
    result = num1 + num2 + num3;
    return result;
}
```

Copyright © Junko Shiragami, Waseda University 2014. All rights reserved.

メソッドを作る(5)(p. 296)

処理内容は何を書いても良い
(if, for, while, ...)

```
public static 戻り値のデータ型 メソッド名(引数) {
    メソッドでの処理内容
    return 処理結果;
}
```

※「処理内容」の部分では、「引数」で定義した変数を通常の変数として利用できる

```
public int sum(int num1, int num3) {
    int result;
    result = num1 + num2;
    return result;
}
```

Copyright © Junko Shiragami, Tokyo Woman's Christian University 2015. All rights reserved.

メソッドを作る(6)(p. 296)

- 「処理結果」を返すという意味、「変数名 = メソッド名(引数)」と書く、「変数名」の中に処理結果が代入される
- この文でメソッドの内容が終わる(この後には文を書かないこと)
- 処理結果のデータ型と戻り値のデータ型は同じもの

```
public static 戻り値のデータ型 メソッド名(引数) {
    メソッドでの処理内容
    return 処理結果;
}
```

Copyright © Junko Shiragami, Tokyo Woman's Christian University 2015. All rights reserved.

return文(p. 296)

- 戻り値のデータ型とreturn文で返すデータ型は同じでなくてはならない

```
public int sum(int num1, int num2) {
    int result;
    result = num1 + num2;
    return result;
}
```

データ型が同じ

```
public String sum(int num1, int num2) {
    int result;
    result = num1 + num2;
    return result;
}
```

違うデータ型

Copyright © Junko Shiragami, Tokyo Woman's Christian University 2015. All rights reserved.

メソッドの作成例(p. 297)

引数に与えられた3つの数の平均を求めるメソッド

```
public static double average(int num1, int num2, int num3) {
    double result;
    result = (double)(num1 + num2 + num3) / 3;
    return result;
}
```

処理結果を「double」型で返す

引数の定義

処理内容

「return」で、「結果を返す」という意味
(「return」の後に書く処理結果のデータ型は、
戻り値のデータ型と同じにすること)

Copyright © Junko Shiragami, Tokyo Woman's Christian University 2015. All rights reserved.

メソッドを作る場所(p. 300)

mainの上か下に作るど
ちらに作っても良い

```
public class ファイル名 {
    ...
    public static void main(String[] args) {
        ...
    }
}
```

Copyright © Junko Shiragami, Tokyo Woman's Christian University 2015. All rights reserved.

メソッドの作成例(p. 300)

```
public class Sample {
    public static double average(int num1, int num2, int num3) {
        double result;
        result = (double)(num1 + num2 + num3) / 3;
        return result;
    }
    public static void main(String[] args) {
    }
}
```

Copyright © Jinko Shingane, Tokyo Woman's Christian University 2015. All rights reserved.

作ったメソッドを使う(p. 301)

- メソッドを使う部分は、「public static void main」の中を書く
- 「メソッド名(引数の値)」でメソッドを呼び出す

Copyright © Jinko Shingane, Tokyo Woman's Christian University 2015. All rights reserved.

作ったメソッドを使う例(1)(p. 301)

```
public class Average {
    public static double average (int num1, int num2, int num3) {
        int sum;
        double result;
        sum = num1 + num2 + num3;
        result = (double)sum / 3;

        return result;
    }
    public static void main(String[] args) {
        double result;
        result = average(10, 20, 30);
    }
}
```

メソッドの定義

Copyright © Jinko Shingane, Tokyo Woman's Christian University 2015. All rights reserved.

作ったメソッドを使う例(2)(p. 301)

```
public class Average {
    public static double average (int num1, int num2, int num3) {
        int sum;
        double result;
        sum = num1 + num2 + num3;
        result = (double)sum / 3;

        return result;
    }
    public static void main(String[] args) {
        double result;
        result = average(10, 20, 30);
    }
}
```

引数には具体的な値または変数を書く。引数の順番は、メソッドを作ったときの順番と同じに。

averageというメソッドを呼び出す

Copyright © Jinko Shingane, Tokyo Woman's Christian University 2015. All rights reserved.

作ったメソッドを使う例(3)(p. 301)

```
public class Average {
    public static void main(String[] args) {
        double result;
        result = average(10, 20, 30);
    }
    public static double average (int num1, int num2, int num3) {
        int sum;
        double result;
        sum = num1 + num2 + num3;
        result = (double)sum / 3;

        return result;
    }
}
```

処理の流れ

- mainメソッドで、averageメソッドの引数に10, 20, 30を指定する
- 指定された10, 20, 30というデータがaverageメソッドの引数の変数「num1」、「num2」、「num3」にそれぞれ代入される
- averageメソッド内で引数の値を使って計算され、変数resultに結果が代入される
- averageメソッドの変数resultの値がmainメソッドの変数resultに代入される

Copyright © Jinko Shingane, Tokyo Woman's Christian University 2015. All rights reserved.

メソッド～引数の扱い～

Copyright © Jinko Shingane, Tokyo Woman's Christian University 2015. All rights reserved.

引数が配列の場合(定義)(p. 302)

- 引数の定義の部分を「**引数のデータ型[]** **引数の変数名**」または「**引数のデータ型** **引数の変数名[]**」のように書く

```
public static int add(int[] num) {
    .....
    return result;
}
```

または

```
public static int add(int num[]) {
    .....
    return result;
}
```

Copyright © Junko Shiragane, Tokyo Woman's Christian University 2015. All rights reserved.

引数が配列の場合(使用)(p. 302)

- 引数の定義の部分に配列変数をそのまま入れる

```
public class Calculate {
    public static int add(int[] num) {
        .....
        return result;
    }
    public static void main(String[] args) {
        int[] number = new int[100];
        int sum;
        sum = add(number);
    }
}
```

添え字なども必要なく、
配列変数の名前のみ

Copyright © Junko Shiragane, Tokyo Woman's Christian University 2015. All rights reserved.

引数がない場合(p. 302)

- 引数の定義の部分に何も書かず、メソッドの利用時の引数も何も書かない

```
public class Calculate {
    public static int add() {
        .....
        return result;
    }
    public static void main(String[] args) {
        int sum;
        sum = add();
    }
}
```

Copyright © Junko Shiragane, Tokyo Woman's Christian University 2015. All rights reserved.

戻り値が配列の場合(p. 302)

- 戻り値の定義の部分を「**戻り値のデータ型[]**」のように書く

```
public class Calculate {
    public static int[] add(int n) {
        int[] result = new int[100];
        .....
        return result;
    }
    public static void main(String[] args) {
        int[] sum;
        int num;
        sum = add(num);
    }
}
```

Copyright © Junko Shiragane, Tokyo Woman's Christian University 2015. All rights reserved.

戻り値がない場合(p. 302)

- 戻り値の定義の部分を「**void**」と書く
- メソッドの最後の「return」文も不要

```
public class Calculate {
    public static void add(int n) {
        .....
    }
    public static void main(String[] args) {
        int num;
        add(num);
    }
}
```

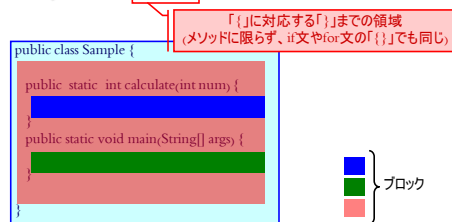
Copyright © Junko Shiragane, Tokyo Woman's Christian University 2015. All rights reserved.

メソッド～変数の有効領域～

変数のスコープ(1)(p. 316)

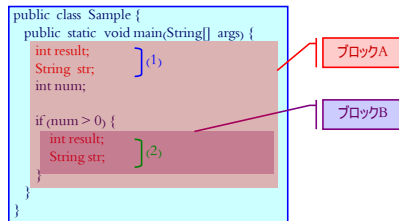
- 変数は、どこで宣言したかによって使える場所が違う

➡ 宣言したブロック内では使えない(変数のスコープ)



Copyright © Junko Shiragane, Tokyo Woman's Christian University 2015. All rights reserved.

変数のスコープ(2)(p. 316)

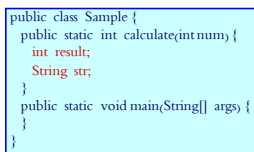


広いブロックの変数は、そのブロックに含まれる狭いブロックでも使える

➡ (1)の変数(ブロックAの変数)はブロックBでも使えるので、
(2)の変数宣言はコンパイルエラー

Copyright © Junko Shiragane, Tokyo Woman's Christian University 2015. All rights reserved.

変数のスコープ(3)(p. 316)

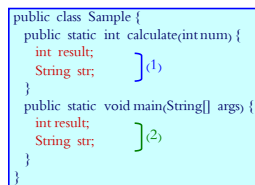


変数は、宣言したブロック内では使えない

➡ result, strは、「public static void main...」の中では使えない

Copyright © Junko Shiragane, Tokyo Woman's Christian University 2015. All rights reserved.

変数のスコープ(4)(p. 316)



変数は、宣言したブロック内では使えない

➡ (1)の変数と(2)の変数は違う変数として扱われる

Copyright © Junko Shiragane, Tokyo Woman's Christian University 2015. All rights reserved.

第3回課題と期末試験

- 第3回課題: 授業のページから
- 期末試験
 - 7月28日(火) 2限 24102教室
 - 90分
 - 持ち込みすべて可の実技メインの試験
 - 筆記も少しあり

Copyright © Junko Shiragane, Tokyo Woman's Christian University 2015. All rights reserved.