Java プログラミング入門

— クラス —

早稲田大学

クラス

```
pubulic class クラス名 {
  /* フィールド（データの集まり） */
  /* コンストラクタ */
  /* メソッド */
}
```

- クラスは，データとそれを処理する部分をまとめたもの
- フィールドやメソッドのことをクラスのメンバと呼ぶ

⇓

クラス (ひな形) を用いてインスタンス (実体) を作成し，
それを利用するプログラムを作成できる

クラス

```
pubulic class クラス名 {
  /* フィールド（データの集まり） */
  /* コンストラクタ */
  /* メソッド */
}
```

- クラスは，データとそれを処理する部分をまとめたもの
- フィールドやメソッドのことをクラスのメンバと呼ぶ

⇓

クラス (ひな形) を用いてインスタンス (実体) を作成し，
それを利用するプログラムを作成できる

クラスの宣言

例題

ある個人の「体格」についてのデータとして，“名前”，“身長”，“体重”をまとめて管理し，標準体重の計算などの健康管理を行うプログラムを作成しなさい．

```
public class Body {
  String name;        — 名前（文字列）
  double height;      — 身長（実数値）
  double weight;      — 体重（実数値）
}
```


Body クラス

クラスの宣言

例題

ある個人の「体格」についてのデータとして，“名前”，“身長”，“体重”をまとめて管理し，標準体重の計算などの健康管理を行うプログラムを作成しなさい．

```
public class Body {
  String name;          — 名前（文字列）
  double height;        — 身長（実数値）
  double weight;        — 体重（実数値）
}
```

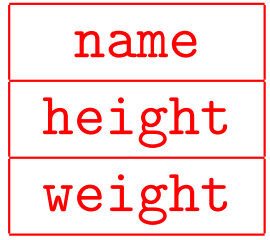

Body クラス

クラスの宣言

例題

ある個人の「体格」についてのデータとして，“名前”，“身長”，“体重”をまとめて管理し，標準体重の計算などの健康管理を行うプログラムを作成しなさい．

```
public class Body {
  String name;          — 名前（文字列）
  double height;        — 身長（実数値）
  double weight;        — 体重（実数値）
}
```



Body クラス

クラスの宣言

- クラスの中でデータのために宣言された変数のことを，フィールドと呼ぶ
- クラスを定義する際は，必要な数だけフィールドを宣言することができる
- クラスを利用することで，宣言したフィールドを一括して扱うことができる

クラスの宣言：「型」の作成

クラスの宣言

- クラスの中でデータのために宣言された変数のことを，フィールドと呼ぶ
- クラスを定義する際は，必要な数だけフィールドを宣言することができる
- クラスを利用することで，宣言したフィールドを一括して扱うことができる

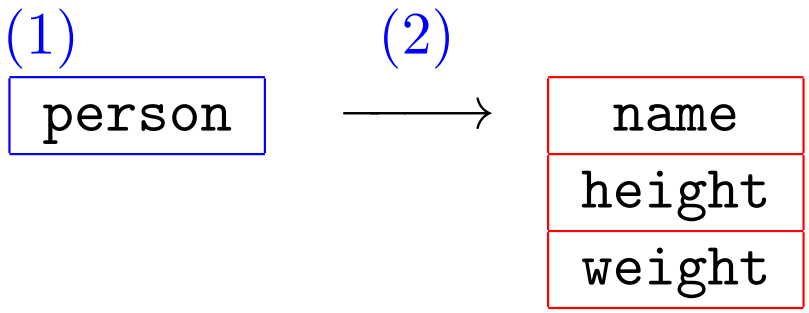
クラスの宣言：「型」の作成

インスタンス

(1) Body 型の変数の宣言

```
Body person;
```

- Body 型の person という変数名が使用できる
- この段階では変数 person の実体は存在しない



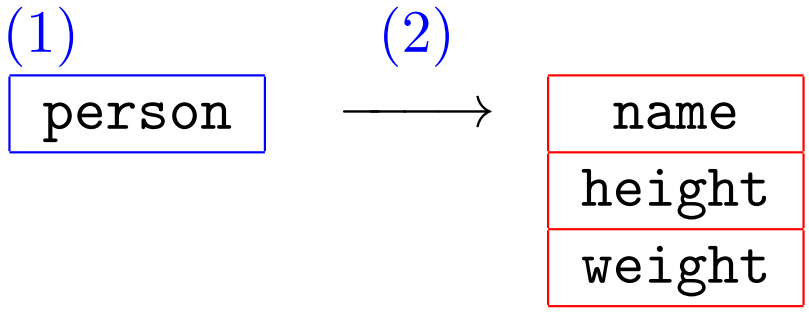
Body 型変数の生成

インスタンス

(2) Body クラスの実体を生成し，変数 person から参照

```
person = new Body();
```

- new 演算子により Body クラスの実体 (インスタンス) を生成



Body 型変数の生成

上の 2 つの手続きをまとめることもできる：

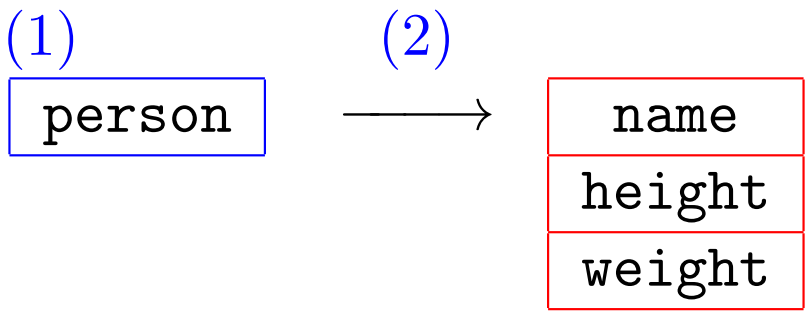
```
Body person = new Body();
```

インスタンス

(2) Body クラスの実体を生成し，変数 person から参照

```
person = new Body();
```

- new 演算子により Body クラスの実体 (インスタンス) を生成



Body 型変数の生成

上の 2 つの手続きをまとめることもできる：

```
Body person = new Body();
```

メンバの参照

変数の各フィールドを参照するには...

変数名 . フィールド名

Body クラスのメンバの参照

値の代入：

```
person.name = "Frank";
person.height = 175.0;
person.weight = 63.5;
```

メンバの参照

変数の各フィールドを参照するには...

変数名 . フィールド名

person

name	···	person.name
height	···	person.height
weight	···	person.weight

Body クラスのメンバの参照

値の代入：

```
person.name = "Frank";
person.height = 175.0;
person.weight = 63.5;
```

person

"Frank"
175.0
63.5

メンバの参照

変数の各フィールドを参照するには...

変数名 . フィールド名

person

name	···	person.name
height	···	person.height
weight	···	person.weight

Body クラスのメンバの参照

値の代入：

```
person.name = "Frank";
person.height = 175.0;
person.weight = 63.5;
```

person

"Frank"
175.0
63.5

プログラム例（1）

Body.java

```
public class Body {
  String name;
  double height, weight;
}
```

プログラム例（1）

SampleBody1.java

```
public class SampleBody1 {
  public static void main(String[] args) {
    Body st1 = new Body();
    st1.name = "Frank"; st1.height = 175.0; st1.weight = 63.5;

    Body st2 = new Body();
    st2.name = "Thomas"; st2.height = 177.0; st2.weight = 72.0;

    System.out.println("Student 1");
    System.out.println(" " + st1.name);
    System.out.println(" " + st1.height + " cm");
    System.out.println(" " + st1.weight + " kg");
    System.out.println("Student 2");
    System.out.println(" " + st2.name);
    System.out.println(" " + st2.height + " cm");
    System.out.println(" " + st2.weight + " kg");
  }
}
```

プログラム例（1）

コンパイル：

```
$ javac␣Body.java ↵
$ javac␣SampleBody1.java ↵
$
```

実行：

```
$ java␣SampleBody1 ↵
Student 1
  Frank
  175.0 cm
  63.5 kg
....
```

プログラム例（1）

コンパイル：

```
$ javac␣Body.java ↵
$ javac␣SampleBody1.java ↵
$
```

実行：

```
$ java␣SampleBody1 ↵
Student 1
  Frank
  175.0 cm
  63.5 kg
....
```

コンストラクタ

SampleBody1.java では...

```
Body st1 = new Body();
st1.name = "Frank";
st1.height = 175.0;
st1.weight = 63.5;
```

- Body クラスのインスタンスを生成し，各メンバに値を設定
- インスタンスの生成と同時にこれらの値を設定したい

コンストラクタ

- インスタンスの初期化を行うための機能.
- インスタンスの作成時に行う処理を記述するための機能.
- クラスの中に記述する.

コンストラクタ

SampleBody1.java では...

```
Body st1 = new Body();
st1.name = "Frank";
st1.height = 175.0;
st1.weight = 63.5;
```

- Body クラスのインスタンスを生成し，各メンバに値を設定
- インスタンスの生成と同時にこれらの値を設定したい

コンストラクタ

- インスタンスの初期化を行うための機能.
- インスタンスの作成時に行う処理を記述するための機能.
- クラスの中に記述する.

コンストラクタの作成

```
public class Body {
  String name;
  double height, weight;

  public Body(String n, double h, double w) {
    name = n; height = h; weight = w;
  }
}
```

- コンストラクタの名前はクラス名と同じにする
- 戻り値がないので「戻り値の型」は記述しない

コンストラクタの使用：

```
Body person = new Body("Thomas", 177.0, 72.0);
```

- インスタンスが生成される際に，コンストラクタの部分がまず実行される

コンストラクタの作成

```
public class Body {
  String name;
  double height, weight;

  public Body(String n, double h, double w) {
    name = n; height = h; weight = w;
  }
}
```

- コンストラクタの名前はクラス名と同じにする
- 戻り値がないので「戻り値の型」は記述しない

コンストラクタの使用：

```
Body person = new Body("Thomas", 177.0, 72.0);
```

- インスタンスが生成される際に，コンストラクタの部分がまず実行される

コンストラクタの多重定義

```
public Body() {                              — 引数なし
  name = "";
  height = 0.0;
  weight = 0.0;
}

public Body(double h, double w) {    — 引数は身長，体重のみ
  name = "";
  height = h;
  weight = w;
}
```

コンストラクタの使用：

```
Body person2 = new Body();                   — 値は設定しない
Body person3 = new Body(170.0, 60.0);    — 身長，体重のみを設定
```

- 引数の数と型を判定し，適したコンストラクタを自動的に呼び出してくれる

コンストラクタの多重定義

```
public Body() {                              — 引数なし
  name = "";
  height = 0.0;
  weight = 0.0;
}

public Body(double h, double w) {    — 引数は身長，体重のみ
  name = "";
  height = h;
  weight = w;
}
```

コンストラクタの使用：

```
Body person2 = new Body();                   — 値は設定しない
Body person3 = new Body(170.0, 60.0);    — 身長，体重のみを設定
```

- 引数の数と型を判定し，適したコンストラクタを自動的に呼び出してくれる

プログラム例（2）

Body.java

```
public class Body {
  String name;
  double height, weight;

  public Body() {
    name = ""; height = 0.0; weight = 0.0;
  }
  public Body(double h, double w) {
    name = ""; height = h; weight = w;
  }
  public Body(String n, double h, double w) {
    name = n; height = h; weight = w;
  }
}
```

プログラム例（2）

SampleBody2.java

```
public class SampleBody2 {
  public static void main(String[] args) {
    Body st1 = new Body("Frank", 175.0, 63.5);
    Body st2 = new Body(177.0, 72.0);
    Body st3 = new Body();

    System.out.println("Student 1");
    System.out.println(" " + st1.name);
    System.out.println(" " + st1.height + " cm");
    System.out.println(" " + st1.weight + " kg");
    ...
  }
}
```

メソッド

SampleBody2.java では...

```
Sytem.out.println("Student 1");
Sytem.out.println("  " + st1.name);
Sytem.out.println("  " + st1.height + " cm");
Sytem.out.println("  " + st1.weight + " kg");
```

- 変数の内容表示のために，各フィールドの値を表示
- メソッドを利用して効率よく表示させたい

メソッド

- データの処理等を行う
- 引数と戻り値を持つ
- クラスの中に記述する

メソッド

SampleBody2.java では...

```
Sytem.out.println("Student 1");
Sytem.out.println("  " + st1.name);
Sytem.out.println("  " + st1.height + " cm");
Sytem.out.println("  " + st1.weight + " kg");
```

- 変数の内容表示のために，各フィールドの値を表示
- メソッドを利用して効率よく表示させたい

メソッド

- データの処理等を行う
- 引数と戻り値を持つ
- クラスの中に記述する

メソッド

```
public class Body {
  String name;
  double height; weight;

  /* コンストラクタの宣言 */

  public void print() {
    System.out.println("  name : " + name);
    System.out.println("height : " + height + " cm");
    System.out.println("weight : " + weight + " kg");
  }
}
```

- メソッドはクラスの中で定義する
- メソッドはフィールドと同様に，クラスのメンバである

メソッドの使用：

```
Body person = new Body("Frank", 175.0, 63.5);
person.print();
```

メソッド

```
public class Body {
  String name;
  double height; weight;

  /* コンストラクタの宣言 */

  public void print() {
    System.out.println("  name : " + name);
    System.out.println("height : " + height + " cm");
    System.out.println("weight : " + weight + " kg");
  }
}
```

- メソッドはクラスの中で定義する
- メソッドはフィールドと同様に，クラスのメンバである

メソッドの使用：

```
Body person = new Body("Frank", 175.0, 63.5);
person.print();
```

プログラム例（3）

Body.java

```
public class Body {
  String name;
  double height, weight;
  public Body() {
    name = ""; height = 0.0; weight = 0.0;
  }
  public Body(double h, double w) {
    name =""; height = h; weight = w;
  }
  public Body(String n, double h, double w) {
    name = n; height = h; weight = w;
  }
  public double stdWeight() {
    return height * height * 22 / 10000;
  }
  public void print() {
    System.out.println(" name : " + name);
    System.out.println("height : " + height + " cm");
    System.out.println("weight : " + weight + " kg");
  }
}
```

プログラム例（3）

SampleBody3.java

```
public class SampleBody3 {
  public static void main(String[] args) {
    double sw;
    System.out.println("== Student 1 ==");
    Body st1 = new Body("Frank", 175.0, 63.5);
    st1.print();
    sw = st1.stdWeight();
    System.out.println("standard weight : " + sw);
    System.out.println("== Student 2 ==");
    Body st2 = new Body("Thomas", 177.0, 72.0);
    st2.print();
    sw = st2.stdWeight();
    System.out.println("standard weight : " + sw);
  }
}
```

基本型と参照型の違い

- 基本型 (int 型，double 型など) は値を入れる変数
- 参照型 (クラス) はインスタンスの位置を示す変数

SampleBody4.java

```
public class SampleBody4 {
    public static void main(String[] args) {

        Body st1 = new Body("Frank", 175.0, 63.5);
        Body st2;
        st2 = st1;
        st2.name = "Robert";
        System.out.println("== Student 2 ==");
        st2.print();
        System.out.println("== Student 1 ==");
        st1.print();
    }
}
```

実行例

実行：

```
$ java␣SampleBody4 ↵
== Student 2 ==
name : Robert
height : 175.0 cm
weight : 63.5 kg
== Student 1 ==
name : Robert
height : 175.0 cm
weight : 63.5 kg
```

何故このような事が起こったのか：

- クラスの変数 (参照型) はインスタンスの位置を示す変数
- st1 と st2 は同じインスタンスの位置を示してしまっている