Java Programming

— Class —

Waseda University

Class

```
pubulic class ClassName {
  /* Fields */
  /* Constructors */
  /* Methods */
}
```

- Classes consist of variables and methods.
- Member variables in a class are called fields.

$$\Downarrow$$

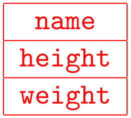
Creating an instance of a class, we can utilize this in our programs.

Declaring Classes

Example

Define the class `Body` which has a name, a height and weight as fields and the method for calculating the standard weight.

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



Body Class

Declaring Classes

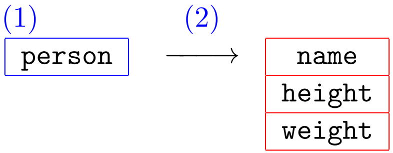
- Member variables in a class are called **fields**
- You can define an arbitrary number of fields.
- The grouped lists of variables can be used by defining a class.

Instance of the class `Body`

(1) Declaring the class `Body`

```
Body person;
```

- `Body person;` means only declaring a variable of the class `Body`
- The variable `person` has no instance of the class `Body`.



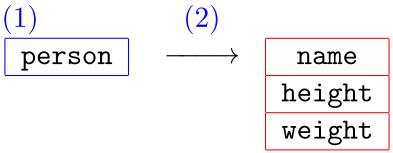
Creating the variable of `Body` class

Instance

(2) Creating the instance of the class `Body`

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

- The instance of `Body` is created by `new` operator.
- `person = new Body();` actually creates a space in memory.



Creating the instance of the class `Body`

You can also write the following code :

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

Referring fields of a instance

How to refer the fields of the instance

```
VariableName.FieldsName
```

person

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

Referring the members of the class `Body`

assigning the values to the fields :

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

person

"Frank"
175.0
63.5

Example (1)

Body.java

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

Example (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");
  }
}
```

Example (1)

Compile :

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

Run :

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

Assigning values to members

In SampleBody1.java...

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

- The above codes created the instance of the `Body` class and assigned the values to fields.
- But we would like to assign the values to members at the same time that the instance of the `Body` class is created.

Constructor

Constructor

- A constructor constructs an instance of a class.
- When the instance of a class is created, then a constructor is called.
- An initial setting is written in a constructor.
- A constructor is in a class.

Declaring a constructor

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

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

- The name of the constructor must have same name as the class name
- A constructor doesn't have ReturnDatatype.

How to call the constructor

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

- To create a new `Body` instance, a constructor is called by `new` operator.

Overloaded constructor

```
public Body() {                          — no parameter
  name = "";
  height = 0.0;
  weight = 0.0;
}

public Body(double h, double w) {    — parameters are height and weight
  name = "";
  height = h;
  weight = w;
}
```

How to use the constructor :

```
Body person2 = new Body();                — no parameters
Body person3 = new Body(170.0, 60.0);     — set a height and a weight
```

- You can define more than one constructor with different parameters.(Overloaded constructor)

Example (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;
  }
}
```

Example (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");
    ...
  }
}
```

Method

In SampleBody2.java...

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

- Display the members of the class Body using a method

$$\Downarrow$$

Method

- A method is a collection of statements that are grouped together to perform an operation.
- A method may have parameters and return value
- Methods appear inside a `class body`.

Method

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

  /* Declaring a class */

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

- Method is defined in a class
- Methods and fields are member of class.

How to use a method in a class :

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

Example (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 * 22.0 / 10000;
  }
  public void print() {
    System.out.println(" name : " + name);
    System.out.println("height : " + height + " cm");
    System.out.println("weight : " + weight + " kg");
  }
}
```

Example (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);
  }
}
```

The difference between primitive variable and reference variable

- The primitive variable (e.g. int and double) is the variable which has a value.
- The reference variable is the variable which has an address

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();
    }
}
```

Example (4)

Run :

```
$ 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
```

Why did the result occur?

- A variable for class(reference variable)is the variable which has a address of object.
- the address of `st1` is the same as `st2`.