Java

Inheritance

Max Langer, Leonard Follner, Alexander Hesse

7. November 2016

Java-Kurs

Overview

Visibilities

Visibilities

- public
- private
- protected

Visibilities

```java
public class Student {
    public String getName() {
        return "Peter";
    }

    private String getFavouritePorn() {
        return "...";
    }
}

// [...]
exampleStudent.getName(); // Works!
exampleStudent.getFavouritePorn(); // Error
```

Arrays

Array

An array is a data-type that can hold a **fixed number** of elements. An Element can be any simple data-type or object.

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

        int[] intArray = new int[10];
        intArray[8] = 7; // assign 7 to the 9th element
        intArray[9] = 8; // assign 8 to the last element

        System.out.println(intArray[8]); // prints: 7
    }

```

You can access every element via an index. A n-element array has indexes from 0 to (n-1).

Array Initialization

You can initialize an array with a set of elements.

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

    int[] intArray = {3, 2, 7};

    System.out.println(intArray[0]); // prints: 3
    System.out.println(intArray[1]); // prints: 2
    System.out.println(intArray[2]); // prints: 7
}

```

Alternative Declaration

There two possible positions for the square brackets.

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

        // version 1
        int[] intArray1 = new int[10];

        // version 2
        int intArray2[] = new int[10];
    }

```

2-Dimensional Array

Arrays work with more than one dimension. An m-dimensional array has m indexes for one element.

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

    // an array with 100 elements
    int[][] intArray = new int[10][10];

    intArray[0][0] = 0;
    intArray[0][9] = 9;
    intArray[9][9] = 99;
}

```

Assignment with Loops

Loops are often used to assign elements in arrays.

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

        int[][] intArray = new int[10][10];

        for(int i = 0; i < 10; i++) {
            for(int j = 0; j < 10; j++) {
                intArray[i][j] = i*10 + j;
            }
        }
    }

```

Arrays with objects

Loops are often used to assign elements in arrays.

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

    Student[][] studentArray = new Student[10][10];

    for(int i = 0; i < 10; i++) {
        for(int j = 0; j < 10; j++) {
            intArray[i][j] = new Student();
        }
    }
}

```

Inheritance

A special Delivery

Our class *Letter* is a kind of *Delivery* denoted by the keyword **extends**.

- *Letter* is a **subclass** of the class *Delivery*
- *Delivery* is the **superclass** of the class *Letter*

```java
public class Letter extends Delivery {

}
```

As mentioned implicitly above a class can has multiple subclasses. But a class can only inherit directly from one superclass.

Example

We have the classes: *PostOffice*, *Delivery* and *Letter*. They will be used for every example in this section and they will grow over time.

```java
public class Delivery {

    private String address;
    private String sender;

    public void setAddress(String addr) {
        address = addr;
    }

    public void setSender(String snd) {
        sender = snd;
    }

    public void printAddress() {
        System.out.println(this.address);
    }
}

```

Inherited Methods

The class *Letter* also inherits all methods from the superclass *Delivery*.

```java
public class PostOffice {

    public static void main(String[] args) {

        Letter letter = new Letter();

        letter.setAddress("cafe ascii, Dresden");

        letter.printAddress();
        // prints: cafe ascii, Dresden
    }
}

```

Override Methods

The method printAddress() is now additional definded in *Letter*.

```java
public class Letter extends Delivery {

    @Override
    public void printAddress() {
        System.out.println("a letter for " + this.address);
    }
}

```

`@Override` is an annotation. It helps the programer to identify overwritten methods. It is not neccessary for running the code but improves readability. What annotations else can do we discuss in a future lesson.

Override Methods

Now the method `printAddress()` defined in *Letter* will be used instead of the method defined in the superclass *Delivery*.

```
public class PostOffice {

    public static void main(String[] args) {

        Letter letter = new Letter();

        letter.setAddress("cafe ascii, Dresden");

        letter.printAddress();
        // prints: a letter for cafe ascii, Dresden
    }
}

```

Super()

If we define a **constructor with arguments** in *Delivery* we have to define a constructor with the same list of arguments in every subclass.

```java
public class Delivery {

    private String address;
    private String sender;

    public Delivery(String address, String sender) {
        this.address = address;
        this.sender = sender;
    }

    public void printAddress() {
        System.out.println(address);
    }
}

```

Super()

For the constructor in the subclass *Letter* we can use `super()` to call the constructor from the superclass.

```java
public class Letter extends Delivery {

    public Letter(String address, String sender) {
        super(address, sender);
    }

    @Override
    public void printAddress() {
        System.out.println("a letter for " + this.address);
    }
}

```

Super() - Test

```java
public class PostOffice {

    public static void main(String[] args) {
        Letter letter =
            new Letter("cafe ascii, Dresden", "");

        letter.printAddress();
        // prints: a letter for cafe ascii, Dresden
    }
}

```

Object

Every class is a subclass from the class *Object*. Therefore every class inherits methods from *Object*.

See `http://docs.oracle.com/javase/7/docs/api/java/lang/Object.html` for a full reference of the class *Object*.

toString()

Letter is a subclass of *Object*. Therefore *Letter* inherits the method `toString()` from *Object*.
`System.out.println(argument)` will call `argument.toString()` to receive a printable String.

```java
public class PostOffice {

    public static void main(String[] args) {
        Letter letter =
            new Letter("cafe ascii, Dresden", "");

        System.out.println(letter);
        // prints: Letter@_some_HEX-value_
        // for example: Letter@4536ad4d
    }
}

```

Override toString()

```java
public class Letter extends Delivery {

    public Letter(String address, String sender) {
        super(address, sender);
    }

    @Override
    public String toString() {
        return "a letter for " + this.address;
    }
}
```

Override toString() - Test

```java
public class PostOffice {

    public static void main(String[] args) {
        Letter letter =
            new Letter("cafe ascii, Dresden", "");

        System.out.println(letter);
        // a letter for cafe ascii, Dresden
    }
}
```