Java

Object-Oriented Programming

Nico Westerbeck

October 11, 2016

Overview

Java-tools

Datatypes

- int, long
- float, double
- String

Control statemenets

- if, else if, else
- while
- for

Mind-tools

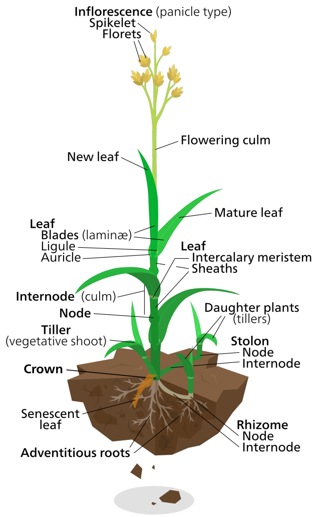
Think in code!

Think in objects!

Think in objects!

The representation of a certain contributor to a problem

Examples



Inflorescence (panicle type)
Spikelet
Florets
Flowering culm
New leaf
Mature leaf
Leaf
Blades (laminæ)
Ligule
Auricle
Leaf
Intercalary meristem
Sheaths
Internode (culm)
Node
Daughter plants
(tillers)
Tiller
(vegetative shoot)
Stolon
Node
Internode
Crown
Senescent
leaf
Rhizome
Node
Internode
Adventitious roots

tion Groenplaats
TICKET

Class vs Instances - the Peter-rule

Class *Student*

```java
public class Student {

    // Attributes
    private String name;
    private int matriculationNumber;

    /**
     * Changes the name
     * @param name The new name of the student
     */
    public void changeName(String name) {
        this.name = name;
    }
}
```

Creation

We learned how to declare and assign a primitive datatype.

```java
        int a; // declare a
        a = 273; // assign 273 to a

```

The creation of an object works similar.

```java
        Student example; // declare example
        example = new Student(); // create an instance
    of Student

```

The **object** derived from a **class** is also called **instance**. The variable[1] is called the **reference**.

[1] in this listing *example*

Calling a Method

```java
public class Student {
    private String name;

    public void changeName(String newName) {
        name = newName;
    }

    public void printName() {
        System.out.println(name);
    }
}

```

The class *Student* has two methods: *void printTimetable()* and *void printName()*.

Calling a Method

```
    public class Main {

        public static void main(String[] args) {
            Student example = new Student(); //
creation
            example.changeName("Jane"); // method call
            example.printName(); // Prints "Jane"
        }
    }

```

You can call a method of an object after its creation with **reference.methodName();**.

Calling a Method

```java
public class Student {
    private String name;

    public void changeName(String newName) {
        name = newName;
        printName();    // Call own method
        this.printName(); // Or this way
    }

    public void printName() {
        System.out.println(name);
    }
}

```

You can call a method of the own object by simply writing **methodName();** or **this.methodName();**

Methods with Arguments

You can call a method with e.g. two arguments via `methodName(arg1, arg2)`.

```
public class Calc {

    public void add(int summand1, int summand2) {
        System.out.println(summand1 + summand2);
    }

    public static void main(String[] args) {
        int summandA = 1;
        int summandB = 2;
        Calc calculator = new Calc();
        System.out.print("1 + 2 = ");
        calculator.add(summandA, summandB); //
prints: 3
    }
}

```

Methods with Return Value

A method without a return value is indicated by **void**:

```java
public void add(int summand1, int summand2) {
    System.out.println(summand1 + summand2);
}

```

A method with an **int** as return value:

```java
public int add(int summand1, int summand2) {
    return summand1 + summand2;
}

```

Calling Methods with a return value

```java
public class Calc {

    public int add(int summand1, int summand2) {
        return summand1 + summand2;
    }

    public static void main(String[] args) {
        Calc calculator = new Calc();
        int sum = calculator.add(3, 8);
        System.out.print("3 + 8 = " + sum); //
prints: 3 + 8 = 11
    }
}

```

Constructors

```
public class Calc {
    private int summand1;
    private int summand2;

    public Calc() {
        summand1 = 0;
        summand2 = 0;
    }
}
```

A constructor gets called upon creation of the object

Constructors with Arguments

```
public class Calc {
    private int summand1;
    private int summand2;

    public Calc(int x, int y) {
        summand1 = x;
        summand2 = x;
    }
}
[...]
Calc myCalc = new Calc(7, 9);

```

A constructor can have Arguments aswell!

An Example

You want to program an enrollment system, for a programming course.

Your classes are:

student	who wants to attend the course
lesson	which is a part of the course
tutor	the guy with the bandshirt
room	where your lessons take place
...	

Your task

- Hope for your tutor to send the classes he showed
- Open+compile them in Eclipse
- Look at them, find something I did not explain yet ;-)
- Add 3 more methods to any of the classes
- Add 3 more fields and use them in your methods
- Call at least 1 of the methods in a given one

Images stolen at:

- **Shire image** Taken from film Lord of the Ring - The Fellowship of the Ring
- **Grass schema** By Thorsten Schmidt, via Wikimedia Commons
- **Ticket-vending-machine** By Thorsten Schmidt, via Wikimedia Commons