Software Engineering and Programming Basics

Object Orientation

Authors of slides:
Prof. Dr.-Ing. Janet Siegmund
Prof. Dr.-Ing. Norbert Siegmund
Prof. Christian Lengauer
Partly extracted from script of PD Dr. Christian Bachmaier

Scope I

- Curly braces show scope of statements:
  - if ([logical expression])
    { [statements when true] }
    else { [statements when false] }
  - Loops: while (…) { [statements in loop] }
  - Methods: void triangularNumber () { [statements in method] }
  - Class: class myClass { [attributes, methods in class] }

- New: Curly braces show where variables are valid

Scope II

- Variables are only valid in the block in which they have been declared (and in sub blocks)

```
int i = 1;
if(i < 5){
  int j = 3;
  i++;
  System.out.println(i + j); // 5
}
System.out.println(i); // 2
System.out.println(j); // Error: j unknow here, is declared in inner block!
```

- Variables with same name in one block (including sub block) are not allowed (exepct class/instance variables)

```
int i = 1;
if(i < 5){
    double i = 1;
    System.out.println(i); // unclear to which i is referred to
}
```

Catching Up: Quiz

- There are two errors per loop. Find and fix them

```
public void errorLoops(char k) {
  for(int i, i < 5; i++) {
    System.out.println(i);
  }

  boolean run = true;
  int x = 0;
  while(k){
    x++;
    if(x > 5)
      continue;
  }

  int counter;
  do {
    int counter = 10;
    counter = counter - 2;
  } while(counter > 0);
}
```

Semi colon instead of comma

i not initialized. Fix: i = 0

No logical expression Fix: while(run)

Endless loop! Fix: break instead of continue

Scope-Error: counter is declared two times

Endless loop! counter is always set to 10

3 Minuten

Catching Up: Calculator

- Implement a calculator on a sheet of paper with the following features:
  - Reads 2 numbers from the user and the operation to be executed [+,-,/,*]
  - Computes the result and prints it
  - After printing the result, it asks the user whether the program should end and reacts accordingly if the user says "yes"

```java
import java.util.Scanner;

public class Test {

public static void main(String[] args) {

String carryOn = "yes";
Scanner sc = new Scanner(System.in);
do {
System.out.print("Please enter the first number:" );
int num1 = sc.nextInt();
System.out.print("Please enter the second number:" );
int num2 = sc.nextInt();
System.out.print("Please enter operator: (+;-;/;*)");
String operator = sc.next();
switch(operator) {
case "+":
System.out.println("Result is: " + (num1 + num2));
break;
case "-":
System.out.println("Result is: " + (num1 - num2));
break;
case "/":
System.out.println("Result is: " + (num1 / num2));
break;
case "*":
System.out.println("Result is: " + (num1 * num2));
break;
default:
System.out.println("Not known/enter valid operation");

}
System.out.println("Do you want to continue?");
carryOn = sc.next();
}
while(carryOn.equals("yes"));
sc.close();
}
}
```

Java

- Structure in Java: classes and objects
- Variables
- Data types
- Constants

```
class Person {
    String firstName;
    String name;
    int age;
    Ort Address;
}
```

Learning Goals

- Getting to know object oriented programming
- Getting to know the difference between classes and objects
- Knowing how to create objects

Structure in Java

Variables

- Variables of classes/objects = attributes
  - They have type and name: `boolean hasSiblings;`
  - Is called variable, because values can vary

```
class Person {
  String firstName;
  String name;
  int age;
  Address address;
}
```

```
[TYPE] [NAME];
OPTIONAL: [TYPE] [NAME] = [WERT];
z.B.: int age = 0; String firstName = "Max";
```

Constants

- Defined once, cannot be changed once the program is running
- Are described by type and name (like variables)
- Name has to consist of capital letters
- In Java, defined by keyword **final** (as in: „Java rocks, and that‘s final!“)

```
final [TYPE] [NAME] = [WERT];

final double PI = 3.14159265359;
final int DAYS_IN_YEAR = 365;
```

What Is Object-Oriented Programming?

- Programming Paradigm
  - Idea: Structure program according to real-world structures
- Different concpets
  - This time: Abstraction – Select only the relevant properties of an object

  – More concepts will follow

Why Do I Need Object-Oriented Programming?

- Idea or problem needs to be put from your „head“ to the „computer“
- Object-oriented programming is one way to do it

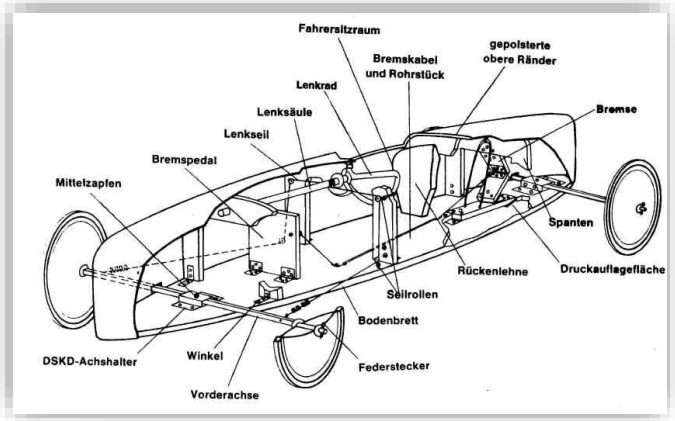
- Example:
  - Store a music collection
  - How do I describe „music collection“ in a program?
    - Class for collection
    - Collection consists of single songs (class for songs)
    - Songs have composers, singers, titel, duration, etc.
  - Saving, searching, filtering, playing are actions that I apply to them

Structure

Behavior

Classes and Objects

- **Classes** = Template/Description of properties that an object must have
- Example: Car
  - 4 Tyres
  - Color
  - Engine (Horsepowers)



- **Object** = Concrete instance of class
  - Example: „My Car“
    - 4 Tyres: Sommer tyres of Pirelli
    - Color: Orange
    - Engine: 170 Horsepowers

First Concrete Example of a class

- Class: Person
  - Given name
  - Name
  - Age
  - Address
- Point
- Line
- Rectangle

In Java

```
class Person {
    String givenName;
    String name;
    int age;
    Ort address;
}
class Point {
    int xCoord,yCoord;
}
class Line {
    Point start;
    Point end;
}
class Rectangle {
    Line horizontal;
    Line vertical;
}
```

Formal: Declaring a Class

- Structure:

```
class [Name] {
   [Declaration of attributes]

   [Declaration of methods]
}
```

- Attributes = Properties of a class
  - Example: Name und Age of a Person
- Methods = Functions of the class
  - Example: maries, isGoingToWork

- Convention: First attributes, then methods (you are helping your tutor a lot if you follow the conventions)

Behavior in Java:
Methods of Objects

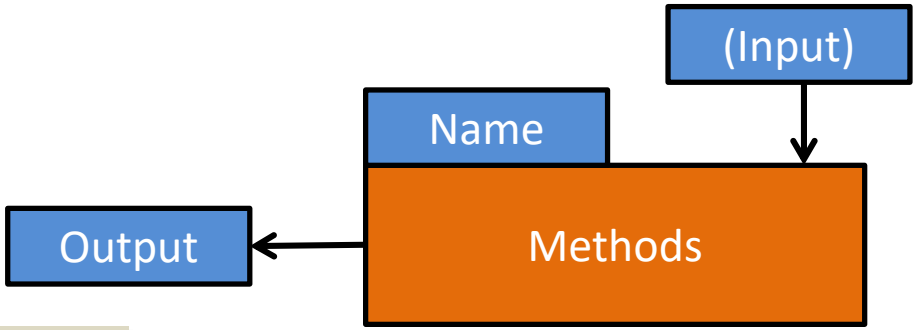
What Does Behavior Mean?

- Describes possible **actions** of an object of a class
  - A car can: drive, be filled up with gas, be repared, or get broken
  - A song can: be played, stopped, rewound, or deleted
- All objects of a class have the **same** behavior

- Behavior is defined via **methods**
  - A piece of code that defines exactly the behavior

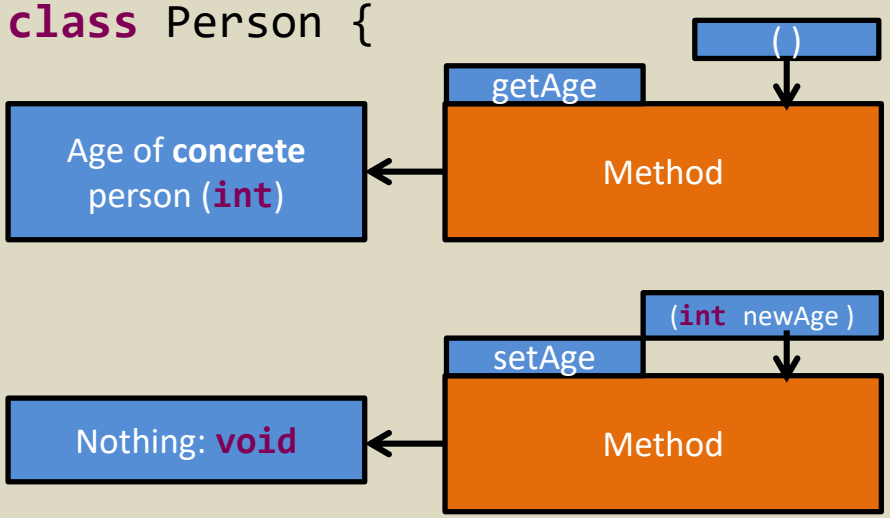
Behavior of Objects

- Is defined via (instance) **methods**




```
class Person {
```

```
int age;
```




```
}
```

Methods of Objects

- Example:

```
class Person {
  String givenName;
  String name;
  int age;
  Residence address;

  void setName(String name) {
    this.name = name;
  }

  String getName() {
    return this. name;
  }

  void isBirthday () {
    this.age = this.age + 1;
  }

  void movesHouse(Residence newAddress) {
    this.address = newAddress;
  }
}
```

- Instance methods are defined within a class, yet they apply to objects (instances) of this class
- Thus, you need to first create an object, before you can use an instance method

- In other words:
- Before I can ask about age, I need a concrete person

Example

Class Person:

Name:	
GivenName:	
Age:	
Address:	

```
void isBirthday() {
    this.age = this.age + 1;
}

int getAge() {
    return this.age;
}

void setAge(int age) {
    this.age = age;
}
```

Object person1

Sisko
Benjamin
42
Deep Space Nine

Object person2

Picard
Jean-Luc
52
Enterprise

Object person3

Janeway
Kathryn
37
Voyager

Methods of Objects II

- Which behavior could be defined by the following classes?

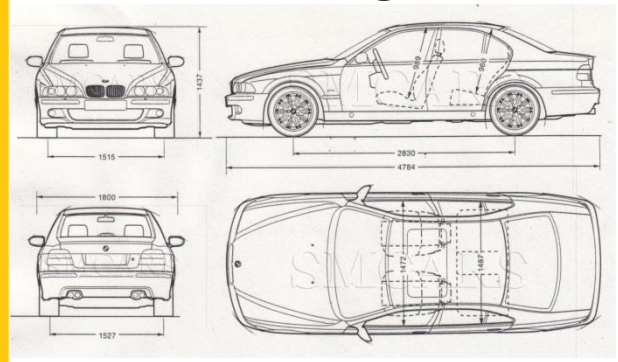
```
class Book {…};
class MusicCollection {…};
class Tweet {…}
```

3 to 5 minutes

Creating Objects
Constructor and Instantiation

Constructor

- How do I get from the class to an object



Factor / Constructor

Constructor in Java I

- Special method: creates new object (a.k.a instance) of a class
- Syntactical difference between a constructor and a method:
  - Constructor has the **same name** as the class of which it creates an object
  - Is the only method **without a return type** (side effect: creation of an object of that class, return: pointer to the object)
- More than one constructor per method is possible, but with a different signature

Constructor in Java II

- Example:

```
class Person {
 String firstName;
 String name;
 int age;

 Person(String firstName, String name) {
  this.firstName = firstName;
  this.name = name;
  this.age = 0;
 }
 Person(int age) {
  this.firstName = "John";
  this.name = "Doe";
  this.age = age;
 }
}
```

- Default-Constructor in Java
  - Is added automatically by the compiler and is empty
  - Does not contain any parameters or statements

Instantiation

- Creation of new object with the **new** Operator
- **new** triggers a call to the constructor
- Example:

```
Person peter = new Person("Peter","Petersen");
Person unknown = new Person(22);
```

- Consequences:
  - Object is created physically in memory
  - Return of call to constructor is pointer to new object of the class

Constructor and Instantiation

- What are the signatures of the constructors for the classes Book and MusicCollection?
- What is the call to the constructors of the class Tweet?

```
class Book {
      Book(String author, String title){…}
      Book(int jahr){…}
}
class MusicCollection {
      MusicCollection(boolean myFavorite){…}
      MusicCollection(char category, double cost, String owner){…}
}
class Tweet {
     public static void main(String[] args) {
              Tweet myTweet = new Tweet("Hello World!");
              Tweet repeatTweet = new Tweet("yes",5);
              Tweet delayTweet = new Tweet("Hihi",3.5,true);
     }
}
```

3 to 5 minutes

Method Calls

Call to Methods of an Object

- Done with „.“ (dot)
- Variable + method name + (correct parameters)

```
peter.isBirthday();
int age = peter.getAge(); // 1

peter.isBirthday();
age = peter.getAge(); // 2
```

- Access to attributes of object also with „.“ (dot)

```
peter.age = 20;
```

Calling Methods of the Same Object

- What if I am the person William and want to call my methods?

```
william.turnsOlder();

void turnsOlder()
{
    this.isBirthday();
    isBirthday();
}
```

Here, I am the object, that is William

- I can refer to „myself“ via `this` (or omit this, in case it is unambiguous)
- Using this can avoid naming conflicts (`this.x` refers to attribute x; x possibly to local variable of a method)

Scope in Object-Oriented Programming

- With a class/instance variable of the same name as a local variable, the local variable has precedence

```
class Person {
  String firstName;
  String name;
  int age;
```

Instance variables (store name and age for every object of type Person)

Local variable with the same name

```
 Person(String firstName, String name) {
    this.firstName = firstName;
    this.name = name;
    this.age = 0;
}…}
```

Local variable has precedence

Using this, the instance variable is used

Take Aways

- Class ≠ objects
  - A class is a form or template
  - An object of a class is an instance (a real representative) with according format

- Necessity and special properties of constructors
- Role of the keyword `this`