Classes and Objects

Tobias Hanf, Maik Göken

December 5, 2022

Learn Programming with Java

Outline

Basics of OOP

OOP in Java

Student example

Exercise

Basics of OOP

What is OOP? (recap)

- Programming paradigm
- Model the real world
- Way of structuring Code and Data
- Uses Classes and Objects

What is an Object?

”An entity that exists in the real world”

Example Objects

Attributes and Methods

- Attributes: define the state of an Object
  - Data
  - Describes the Object
  - Other names: fields, properties
- Methods: describes the behavior of an Object
  - Code
  - Changes the state of the object
  - Or interacts with other objects

What is a Class?

”A template for creating a similar type of object”

Example Class

What are possible attribtues and methods for universities?

Example Class

What are possible attribtues and methods for universities?

- Attributes:
  - Name
  - Founding year
  - Number of Students enrolled
  - List of majors
  - List of enrolled students
  - ...

Example Class

What are possible attribtues and methods for universities?

- Attributes:
  - Name
  - Founding year
  - Number of Students enrolled
  - List of majors
  - List of enrolled students
  - ...
- Methods:
  - Give name
  - Enroll a student
  - Graduate a student
  - Give list of majors based on interests
  - ...

OOP in Java

Classes

A class can be defined as follows:

```
// Defining a Class
class <class-name> {
    <attributes>
    <methods>
}

// Creating a new Object
<class-name> <obj-name> = new <class-name>();
```

Every class should be in their own file (there are exceptions).
The filename should be the same as the classname.

Attributes

Attribute declaration same as for variables but inside the class body:

```
class <name> {
    // Defining an attribute
    <type> <var-name>;
}

// Accessing an attribute
<obj-name>.<var-name>;
```

An attribute can be accessed via the `.` operator.

Methods

Methods definition same as for functions:

```
class <class-name> {
    // Defining and implementing a method
    <ret-type> <method-name>(<param-list>) {
      // code
    }
}

// Calling a method
<obj-name>.<method-name>(<arguments>);
```

A method can be called with the `.` operator.

University example

```java
// File University.java

  class University {
      String name;
      int yearOfFounding;
      int numberOfStudents;

      int getAge(int currentYear) {
      return currentYear - yearOfFounding;
      }

      void enrollStudent(){
      numberOfStudents += 1;
      }
  }
```

Student example

Scenario

You, as a programmer, get the task to create a software for managing students of a university. The first version of the software should keep a record of all student currently enrolled in our university. Because our university is very small, only 5 student are currently enrolled. Each student has a name, the year of birth, an enrolment number and the current degree the student is enrolled in. All student should also have an array for storing up to 10 marks.

The customer wants to be able to get age of a student, add marks for a student and get the current average mark.

Attributes

- Name (String)
- Year of birth (Integer)
- Enrolment number (Integer/String)
- Current Degree (String)
- Marks (Array of Float)

Methods

- `float getAge(int year)`
- `boolean addMark(float mark)`
- `float getAverageMark()`

Exercise

The structure

Create a new directory with the name UniversityResourcePlanner. In the new directory create two files, and the corresponding classes, with the names with UniversityResourcePlanner and Student.

Classes

The class UniversityResourcePlanner should contain the main method. The class Student should implement the attributes and methods we discussed in the previous section (Student example). Create an object of the class Student in the main method of UniversityResourcePlanner. Set the attributes (you decide the exact values) and test the different methods.

Program

In the main method of UniversityResourcePlanner create an array for 5 objects of the class Student. Fill the array with 5 different Students (they should have at least the name, year of birth and degree set).

Implement the following three functions:

- Print all students which are enrolled in a certain degree
  - the degree should be given as a argument
- Find the student with the best (lowest) average grade
- Find the youngest student

All functions should also take the array of students as an argument.