Prof. Dr.-Ing. Ralf Schenkel
Martin Blum
Trier University

Wintersemester 2024/2025

Exercises for the Class

# Elements of Computer Science: Programming

Assignment 11

Submission of solutions until 30.01.2025 at 10:00
at `moodle.uni-trier.de`

- Every task needs to be edited in a meaningful way in order to get a point!
- Please comment your solutions, so that we can easily understand your ideas!
- If you have questions about programming or the assignments, just ask you teachers!
- **Submission that can't be compiled are graded with 0 points!**

## Exercise 1 (Evaluation: predefined `main` method)

The goal of this task is to familiarize yourself with nested objects. You are already familiar with examples from the lecture in which classes "store" instance variables such as arrays, integers and others. However, classes can of course also store objects of other classes. To illustrate this in a practical example, your task is to implement a simple family tree. In order to simplify the task, we have not included aspects such as more complex family structures, same-sex parents and gender.

To do this, implement the following components:

- Implement a class `Human`, which has a name (`String name`) and a unique ID (`int id`). The ID is required in the event that family members have the same name. The ID should start at 1000.
- Implement a constructor for the class `Human` which only receives a name as input. The ID should be set automatically.
- Next, implement the classes `Man` and `Woman`. Both classes are subclasses of the class `Human`. In this case, they do not contain any additional information other than that of the superclass. With the two subclasses, we are able to differentiate between male and female humans in this implementation.
- Next, extend the class `Human` so that it stores the mother (`Woman mother`) and the father (`Man father`) of a human in addition to the name and an ID. The previously implemented constructor should also be extended so that a mother and a father can be passed and set accordingly.
- For this part, it should be possible to create objects of the class `Human` without information about the parents. To do this, you must implement corresponding constructors. Please note that a mother must always be an object of the class `Woman`. The same applies to the father. You can find examples of the constructors to be implemented in the evaluation.

**Exercise 2 (Evaluation: predefined `main` method)**

An abstract class `Recipe` with four methods is given: `recipeName()`, `ingedientsList()`, `equipment()` and `preparationTime()`. It also contains a (static) `main` method for a simple test of the solution.

Derive three classes `Pizza`, `Sandwich` and `Risotto` from `Recipe`. The `recipeName` and special ingredients should be placed in the constructors. General ingredients, the required cooking utensils and the preparation time result from the respective class:

- `Pizza`:
  General Ingredients: Yeast Dough, Tomatoes, Mozzarella, Oregano
  Equipment: Oven, Pizza Plate, Rolling Pin
  Preparation Time: 40
- `Sandwitch`:
  General Ingredients: Butter
  Equipment: Knife, Cutting Board
  Preparation Time: 3
- `Risotto`:
  General Ingredients: Rice, Vegetable Bouillon, Parmesan
  Equipment: Pot, Stirring Spoon
  Preparation Time: 40

The `ingedientsList` should consist of the general ingredients for the basic recipe and the special ingredients from the constructor.

The default `main` method should therefore lead to the following output:

```
Name:         Pizza Salami
Ingredients: Salami, Yeast Dough, Tomatoes, Mozzarella, Oregano
Equipment:   Oven, Pizza Plate, Rolling Pin
Duration:    40.0
Name:         Pizza Diavolo
Ingredients: Salami, Peperoni, Yeast Dough, Tomatoes, Mozzarella,
             Oregano
Equipment:   Oven, Pizza Plate, Rolling Pin
Duration:    40.0
Name:         Ham Sandwich
Ingredients: Brown Bread, Ham, Butter
Equipment:   Knife, Cutting Board
Duration:    3.0
Name:         Toast Hawaii
Ingredients: Pineapple, Cheese, Butter
Equipment:   Knife, Cutting Board
Duration:    3.0
Name:         Pumpkin Risotto
Ingredients: Pumpkin, Rice, Vegetable Bouillon, Parmesan
Equipment:   Pot, Stirring Spoon
Duration:    40.0
```

**Exercise 3 (Evaluation: predefined `main` method)**

Model different road users in a hierarchical structure. To do this, implement the following model:

a) An abstract class `Traffic`, from which all other classes are derived:

- Road users have common fields: `name` (string), `wheels` (int), `drivenKilometers` (double) and `maxSpeed` (int).
- Write a constructor that receives the expected parameters and writes them into the variables provided.
- A vehicle of class `Traffic` also has an abstract method with the signature

  ```
  abstract boolean licenseNeeded()
  ```

  This method should return whether a driving license is required, depending on type of vehicle (car, motorcycle or bicycle).
- Implement a method with the signature

  ```
  public void addKilometer(int km).
  ```

  This method should be used to increase the number of kilometers traveled.
- `String toString()` should return a string of the following form:

  ```
  name, drivenKilometers, maxSpeed, wheels
  ```

b) Derive from the class `Traffic` the classes `Car`, `Motorbike` and `Bicycle`

- A **bicycle** has only two wheels and a maximum speed of 30 km/h. Furthermore, a bicycle can be used by a courier. Therefore this class should store a value `carrier` (boolean). Design the constructors of the class so that you do not need to specify a number of tires or a maximum speed.
- Like a bicycle, a **motorbike** has only two wheels, but can travel at any speed (depending on the model, etc.). When implementing the constructor, no information about the number of wheels is necessary.
- A **car** has four wheels and can have any maximum speed like a motorcycle.

c) Derive from the class `Car` a class `VintageCar`. This class stores a value `year` (**`int`**) with the year of manufacture of the car. Design an appropriate constructor.

d) Extend all classes derived from `Traffic` (including vintage car) with a `toString()` method. This method should call the `toString()` method of the superclass, but first specify the type of the vehicle. This results in a string of the following form:

```
Car:  name, drivenKilometers, maxSpeed, wheels
```

A vintage car should use the method of the superclass (`Car`) too add the following label:

```
Car:  name, drivenKilometers, maxSpeed, wheels (Vintage)
```

e) Implement two interfaces `Collectible` and `ProtectiveClothing`

- Since any motor vehicle could be a collectible, `Car`, `Motorbike` and `VintageCar` should have the interface `Collectible` The interface has only one method with the signature **`int`** `calcValue()`. The value of a collector's item is, for the sake of simplicity, calculated by multiplying the driven kilometers by the number of wheels.
- The second interface should contain a method with the signature

  ```
  boolean needsProtectiveClothing()
  ```

  All classes derived from `Traffic` should have this interface and implement the method `needsProtectiveClothing()`. Cars and vintage cars do not need special protective clothing.

**Attention**: The tasks are intentionally underspecified. Find meaningful and suitable solutions and familiarize yourself with the `main` method!