Exercises in object-oriented programming

Prof. Dr. M. Zeller
HS Ravensburg-Weingarten

October 28, 2016

1 Getting started

In order to start with the lab course in object oriented programming you should have installed the netbeans IDE (or Eclipse) and you should have access to the e-learning (Moodle) course oriented programming.

In the moodle curse you can find a zip-file for each trainig task. Please download the zip-file and unpack it; it will expand to a folder which can be used in netbeans (see below).

2 The IDE netbeans

2.1 Opening an existing netbeans project

- ▷ start netbeans
- ▷ select "File" → "Open Project"
- ▷ in the file-chooser navigate to the appropriate directory i. e. a directory containing a sub-directory named "nbproject"
- ▷ click the button "Open Project"

2.2 Creating a netbeans project

- ▷ start netbeans
- ▷ select "File" → "New Project"
- ▷ select "Java" and "Java Application"
- ▷ click "next"
- ▷ edit "Project Name"
- ▷ edit "Project Location" if neccessary,
  leave check box "Create Main Class" checked
- ▷ click "finish"

2.3 Starting a netbeans project

- Load the project that contains the program to start.
- In the project window (left side) right-click the project → "Set as Main Project". The name of the project appears in bold face if it is selected as Main Project.

- If there is more than on main()-method defined in your program select one of it: Right-click the project → "Set Configuration" → "Customize . . . ". In the left part of the dialog klick "run". In the right part enter or select a value for the field "Main Class" and close the dialog.

When you klick the "run"-button in the netbeans menu bar, it will start the selected main()-method.

3 First Steps in Java

3.1 Multiplication Table

Start with the skeleton program given in the project `Multiplicationtable_A`. Write a program that shows a multiplication table on the screen:

```
 1  2  3   4  ...
 2  4  6   8  ...
 3  6  9  12  ...
 :  :  :   :
 :  :  :   :
```

First the program asks the user for the number of rows and the number of lines. Then the program shows the table on the screen.

Please, test your program at least with the following input: (4, 6); (0, 0); (8, 2); (127, 4).

3.2 Calculator

Implement a simple calculator running in a loop. Inside the loop the program reads three input values and calculates a result. Start with the skeleton program given in the project `Calculator`.

The first and the third input values are the operands. The second input value is taken as operator, it should be one of the character `{+ - * / q}`.

If the operator is none of the given characters the program issues a message and continues. If the operator is `q` th program exits.

The program should be implemented in two functions. The main-function contains the loop, the code for reading the input values and for displaying the output. A second function should calculate the result. The prototype of the second function is:

```
static float calculate(char operator, float operand_1, float operand_2);
```

Use a switch-statement to distuinguish the different cases within this function. The function should issue an error message if an undefined operator is given.

3.3 Numerics

Run the program given in the project `Numeric` and explain the results.

3.4 Prime Numbers

Write a program that uses two nested loops (`while` or `for`) and the modulus operator (`%`) to detect and print prime numbers. Start with the skeleton program given in the project `PrimeNumbers`

3.5 Multiplication Table stored in an Array

Note: You need to understand the differences between arrays in C and arrays in Java in order to work on this project.

Start with the skeleton program given in the project `Multiplicationtable_B`.

Complete the class `TabHolder` and the main method. The main method creates an object of class `TabHolder` named `aTabHolder` . It calculates the values of the multiplication table and stores these values in the object `aTabHolder`. The method `printTab()` of class `TabHolder` prints the stored table on the screen. After that the main method accesses the the values of the first line of the stored table in object `aTabHolder` and prints them on the screen.

```
Enter number of columns (< 128):  6
Enter number of rows (< 128):  4
    1    2    3    4    5    6
    2    4    6    8   10   12
    3    6    9   12   15   18
    4    8   12   16   20   24

 ---- now storing table in array ----
>    1    2    3    4    5    6
>    2    4    6    8   10   12
>    3    6    9   12   15   18
>    4    8   12   16   20   24

 ---- now access first row of table ----
first row:    1   2   3   4   5   6
---- done ----
```

4 Composition – linking objects

4.1 Project MediaAdmin I

Open the project `MediaAdmin`. It contains three classes: a class Medium, a class MediaCopy and a class MediaAdmin. Please leave the class MediaAdmin unchanged. The class Medium holds two members (type int and type String). The class MediaCopy contains three members (int, String and Medium). Thus a MediaCopy-object can contain a reference to a Medium-object (s. class MediaAdmin line 51).

Complete the implementation of the classes Medium and MediaCopy. More details are given inside the files Medium.java and MediaCopy.java. Start with class Medium; MediaCopy is slightly more complicated.

The output of the program should be:

```
Medium, id: 11, title: aaa
comparing media:
medium_1, medium_2: false
medium_4, medium_3: false
medium_1, medium_4: true
comparing copies:
copy_1, copy_2: false
copy_1, copy_3: true
copy_1, copy_3: false
Copy signature: 321, location: K102
Copy signature: 321, location: K102 Medium, id: 11, title: aaa
```

4.2 Project StudentSubjects

Open the project `StudentSubjects`. It contains four classes: a class `StudentSubjects`, a class `CopyStudent`, a class `Subject` and a class `Registration`. Please leave the classes `StudentSubjects` and `Subject` unchanged for the moment.

The program creates one object of type Student and 10 Objects of type Subject.

The class `Registration` hosts a reference to an object of type `Subject` and a reference to an object of its own type. Thus objects of this type can be linked to form a list. Furthermore the class has a member year (`int`) and a member examPlanned (`boolean`) to indicate whether the student planes to take the exam in this subject.

The class `Student` has two members: The name (`String`) and a reference to an object of type `Registration`. The latter is the head of a list of all registrations of the student. The method `pushSubject` adds a new registration to the beginning of the list.

Complete the methods of class `Student`, some hints are given in the program. Start with the methods `pushSubject()` and `popSubject()`. These methods modify the head of the

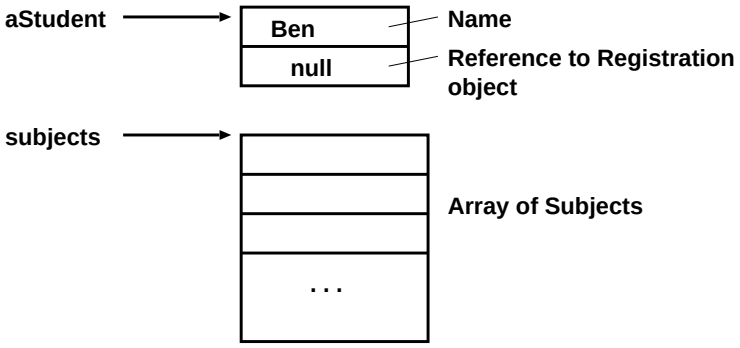


Figure 1: Data structure of the program after line 14 of `StudentSubjects` completed

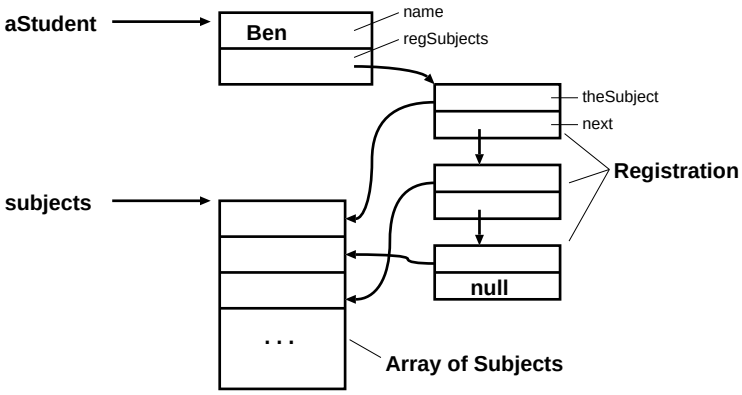
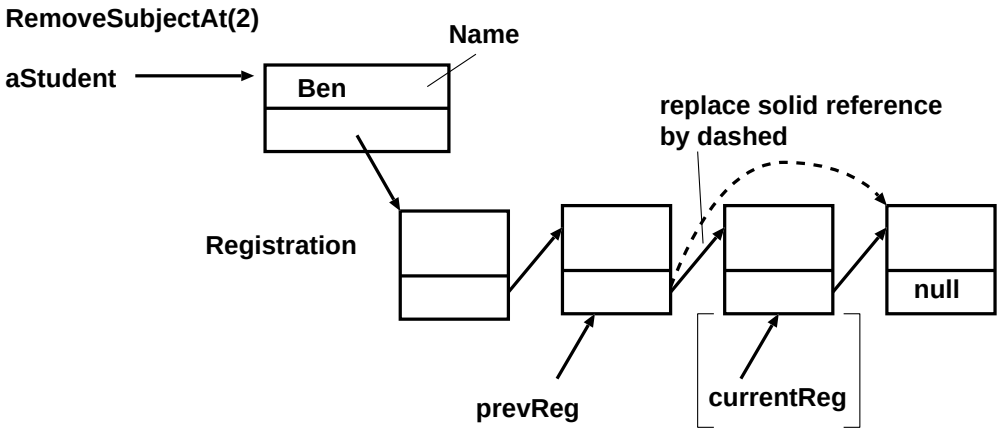


Figure 2: Three registrations are pushed to the list of student Ben

list of registrations. The method `appendSubject()` adds a registration at the end of the list.

The method `removeSubjectAt()` removes a registration from an arbitrary position of the list. Please see the following graphics to implement `removeSubjectAt()` and `removeFromRegistration` respectively.



The method `removeSubjectAt_II()` uses a second reference: `currentReg`. Languages

without garbage collection such as C, C++ require this second reference in order to release the memory of the removed object. In Java the garbage collector releases the memory.

4.3 Project MediaAdmin II

Create two subclasses of medium. One for books, one for videos.

The class `Book` contains two additional members: Number of pages and author The class `Video` also contains two additional members: length and producer

For both clases implement a constructur and the method `print()` using the method of the superclass.

An example for a program run:

```
medium command (b, v, d, p, a): b
media id: 11
  media title: aaa
  author: Anne
  number of pages: 111

m (medium) or c (copy) or q (quit): m
medium command (b, v, d, p, a): b
media id: 22
  media title: bbb
  author: Bob
  number of pages: 222

m (medium) or c (copy) or q (quit): m
medium command (b, v, d, p, a): p
print, media id: 22
 Book by Bob,  Medium, id: 22, title: bbb, 222 pages.


m (medium) or c (copy) or q (quit): m
medium command (b, v, d, p, a): p
print, media id: 33
 Video by Clare,  Medium, id: 33, title: ccc, 333 minutes.


m (medium) or c (copy) or q (quit): m
medium command (b, v, d, p, a): a
 Video by Clare,  Medium, id: 33, title: ccc, 333 minutes.
 Book by Bob,  Medium, id: 22, title: bbb, 222 pages.
 Book by Anne,  Medium, id: 11, title: aaa, 111 pages.
```

```
m (medium) or c (copy) or q (quit):
```

5 Exceptions

Open the project `ExceptionTest`. It contains the class `ExceptionTest` and three classes `xxxException`.

Complete the class `GreenException`.

In class `ExceptionTest` add catch clauses for the different types of exceptions. Note that it makes a difference if you place an additional catch clause befor or after the existing line `catch (Exception ex)`.

Add a print statement to each catch clause so you can see which type of exception is cought (a self-defined Exception or a RuntimeException).

Run the program with input ranging from 1 to 6.

6 Java IO

- Load the project Java_IO_Example into the IDE. In the file Java_IO_Example.java fill in the missing pieces. You might want to have a look in the file `CarManager.java` of project `CarComposition_IO`.
- Add a new member into DataHolder.java: `short testValue` and store a random-value between 10 and 99 in it. Extend the methods in Java_IO_Example.java in order to write an read the new member.

7 ArrayList

7.1 Example ListExample

- Load the project ListExample (Moodle ListExample_...) into the IDE.
- Fill in the missing code pieces to make the program run again. You might want to take a look at the documentation of the interface `Iterator`.
- Change the class of variable `myList` from `ArrayList` to `LinkedList`. Does this change require other changes in the program?

7.2 Example CollectionArrayList

- Load the project CollectionArrayList (Moodle CollectionArrayList_21_12_2014) into the IDE.
- Fill in the missing code pieces to make the program run again.
- Note the data type of variable `studentList` – change the used class from `ArrayList` to `LinkedList`.
- Change the `save-` and `read-` methods. They should make only one call to `writeObject()` and `readObject()` respectively in order to save and read the complete list of objects.

8 The Java Native Interface

- Load the project JniTest (Moodle JNI_Example_20_01_2014) into the IDE.
- Add a new native Method to the class SensorProxy: `native int setTwoData(int, double)`. Go through the process of providing the corresponding function in the C-module
  jnitest_SensorProxy.