Department Mathematik/Informatik, Abteilung Informatik
Software & System Engineering
Weiterführende Konzepte der Programmierung, SS 2025

Niklas Bockholt, Chiara Hafner, Alicja Bezpalko,
Jiyan Alis, Dilara Günay,
Simon Wolf, Adrian Bajraktari, Dr. Mersedeh Sadeghi

Homework 2. Object-Oriented Programming

Exercise 1. Textbook Info System

Implement the following class hierarchy:

- Class `Publication` with attributes `title` (String) and `year` (int), and a method `getInfo()` that returns a formatted string.
- Class `Book` that extends `Publication`, adds `author`, and overrides `getInfo()` to include the author.
- Class `Textbook` that extends `Book`, adds `subject`, and overrides `getInfo()` to also include the subject.

Example output:

```
Data Structures, published in 2020, by Jane Doe Subject: Computer Science
```

Exercise 2. LibrarySystem

Given the following Java code:

```
1  class User {
2      String name = "Unknown";
3      public String getRole() {
4          return "General user";
5      }
6
7      public String getName() {
8          return this.name;
9      }
10 }
11
12 class Librarian extends User {
13     String department = "Reference";
14     @Override
15     public String getRole() {
16         return this.name + "Librarian";
17     }
18
19     public void work() {
```

```
20          //do some work
21      }
22  }
23
24  public class LibrarySystem {
25      public static void main(String[] args) {
26          User u1 = new User();
27          Librarian l1 = new Librarian();
28          User u2 = l1;
29
30          System.out.println(u1.getRole());
31          System.out.println(l1.getRole());
32          System.out.println(u2.getRole());
33
34          l1.getName();
35          u1.getName();
36
37          l1.work();
38          u2.work();
39      }
40  }
```

Draw the memory diagram of the code, including:

- Stack and heap memory.
- Objects, their attributes and references.
- Classes and vTables.
- Show exactly which methods are inherited.
- Provide indices for all object attributes and vTable entries.

Say what the print statements print to the console and explain briefly why.

Then, show how the

- three method calls in lines 30–32 of `getRole()`,
- two calls in lines 34–35 of `getName()`,
- and two calls in lines 37–38 of `work()`

would be translated using static indices only. It could be that some of the calls can not be compiled at all. In this case, explain why

Exercise 3. Mailbox

Create a very simple e-Mail inbox. For this, create

- a class `Mail`.
- a class `Inbox`.

The `Mail` class should store the sender address, the subject, the message, the datetime and whether it has been read. It should offer methods to mark it as read and to print it in a formatted manner. Example format: `<subject> from <sender> on <datetime>: <message>`.

The `Inbox` class should store all emails. It should have a method that prints all e-mail headers (all information except the message). Example format: `<read> | <subject> | <sender> | <datetime>`. It should have another method `Inbox::open(int index)` to open (print) a specific e-mail at the index in the list.

Implement a method `Inbox::countUnread()` to count all unread messages in the inbox.

Add a guard to the `Inbox::open(int index)` method that checks that the `index` is not larger or equal than the size of the list.

In your `main` method, create an inbox, add some e-mails to it and mark some of them as read. Let it print the number of unread e-mails. Open some e-mails.