Examination in Object Oriented Programming

University of Applied Science Ravensburg-Weingarten
Prof. Dr. M. Zeller

Date, time	February 07th, 2015, 10:30 – 12:00 (90 min)
Number of pages	13 pages (including title)
Resources	All accepted resources

Study Progam	Exam. No.	Room
EI	2451	H061
EI	2608	H061

Name: _____ Matriculation number: _____

Reminder:

- Please note name and matriculation number on each sheet.
- If you use additional sheets do not forget to note name and matriculation number on them too.

---

leave blank, please:

Part	1	2	3	4	Sum
max.	27	12	5	20	64
Points					

Preliminary note The examination is quite extensive. Don't be scared – you may miss some points an still get an A grading, you need less than 50 % to pass the exam.

Part 1

1.1 (11 Points)

Analyse the program given in section "Constructors".

What is the output of the program just after line 83 is executed? Note: there might be more dotted lines than actually used.

What is the additional output of the program just after line 86 is executed?

What is the additional output of the program just after line 89 is executed?

```
  in Animal()
  in Mamal()
  in Saddle()
  in Horse()
--------- 1 -----------
 Horse:
  Name: Morningstar
  Saddle color: brown, size:2
--------- 2 -----------
 Horse:
  Name: Morningstar
  Saddle color: dark brown, size:3  Age: 11
```

1.2 (3 Points)

Now we change a part of the definition of class `Saddle` as follows:

```
public class Saddle {
    String color = "brown";
    private int size = 2;
        :
        :
}
```

Without any further changes the compiler now signals an error in class `Horse`, method `changeSaddle`. Which line(s) of the method cause(s) an error?

Change the method `changeSaddle` so that it can be compiled and it achives the same effect as the original method.

```
public class Horse extends Mamal {
    private Saddle theSaddle = new Saddle();

    public void changeSaddle(String newColor, int newSize) {
        theSaddle.color = newColor;
        theSaddle.setSize(newSize);
    }
}
```

1.3 (4 Points)

Now we add the Class `class HorseAdmin` in package `admin`.

Does this class compile? If not, list all lines (give the line number) that will cause the compiler to signal an error. Omit lines which do not cause any error.

Lines:

```
Horse aHorse = new Horse();
aHorse.name = name_ ;
if(aHorse.age < 20){
```

1.4 (9 Points)

Define a class `Elephant` as a subclass of class `Mamal`. The class contains a member `lifespan` with the following properties:

- ▷ It can hold an integer value
- ▷ The value is only accessable within the class `Elephant`
- ▷ The value is 70
- ▷ The value can not be changed.

Complete the definition of member `lifespan` at the ellipsis.

Complete the definition of the constructor. The string-argument is used to set the name of the elephant. The integer-argument should set the member `age` (how can you achive this?).

Complete the method `getRelativeAge()`. The return value is: the value of member `age` times 100 divided by the value of member `lifespan`.

Creating and using an object of the class:

```
Elephant el = new Elephant("Bob", 50);
el.print();
```

Output:

```
Elephant
 Name: Bob
 relative age: 71%
```

```
public class Elephant extends Mamal {
    private final int lifespan = 70;

    Elephant(String name_, int age_) {
        setAge(age_);
        name = name_;
    }
    protected int getRelativeAge() {
        return (getAge() * 100) / lifespan;
    }
    public void print() {
        System.out.println(" Elephant ");
        super.print();
        System.out.println("  relative age: " + getRelativeAge() + "%");
    }
}
```

Part 2

Analyse the program given in section **Railroad**.

Complete the given methods:

2.1 (4 points)

The method `void addLocomotive()` assigns a new value to the member `theLocomotive`.

The method `void addWaggon()` assigns a new value to the member `theWaggons`. The previusly stored reference should be added to the new waggon in order to form a linked list as shown in the figure below.

The data structure after line 47 in the method `main()` is executed:

2.2 (8 points)

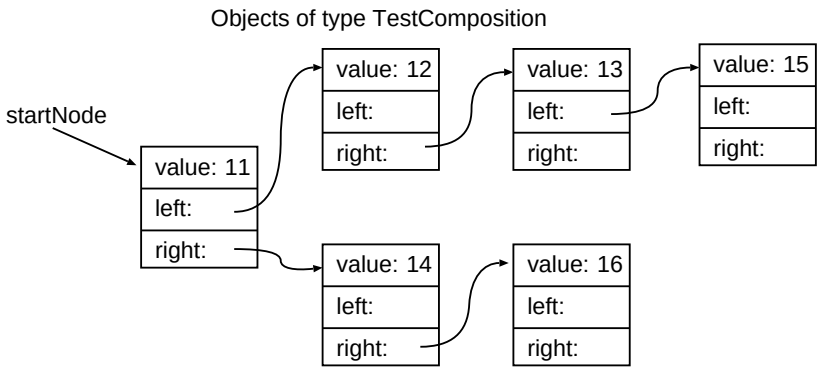
The method `int getWeight()` computes the weight of the train by summing up the weight of the locomotive and the weight of all linked waggons. The method should not throw an exception even if there is neither a locomotive nor any waggon linked to the train.

The output of the program is:

```
(1) Weight: 0
(2) Weight: 55
(3) Weight: 66
```

```
1  public class Train {
2      int trainID;
3      Locomotive theLocomotive;
4      Waggon theWaggons;
5
6      Train(int id) {
7          trainID = id;
8      }
9
10     void addLocomotive(Locomotive newLocomotive){
11         theLocomotive = newLocomotive;
12     }
13
14     void addWaggon(Waggon newWaggon){
15         newWaggon.follower = theWaggons;
16         theWaggons = newWaggon;
17     }
18
19     int getWeight(){
20         int sum = 0;
21         if (theLocomotive != null){
22             sum = theLocomotive.weight;
23         }
24         Waggon currentWaggon = theWaggons;
25         while (currentWaggon != null){
26             sum += currentWaggon.weight;
27             currentWaggon = currentWaggon.follower;
28         }
29         return sum;
30     }
31 }
```

Solution:


Objects of type TestComposition
startNode
value: 11
left:
right:
value: 12
left:
right:
value: 13
left:
right:
value: 15
left:
right:
value: 14
left:
right:
value: 16
left:
right:

Part 3

3.1 (5 points) Exception Handling

Analyse the program given in section **Exception**.

The program defines some exception classes and a main class, Method `bar()` throws a number of different exceptions. In method `foo()` fill in the code for catching these exceptions. The program reports the type of the exception. Note: Each of the exceptions thrown in the program should be cought separately – sequence matters.

Note: The clauses for catching the `LeftException` and for catching the `RightException` may be interchanged. It is crucial that both `LeftException` and `RightException` are caught before `CenterException` and `CenterException` is caught before `Exception`.

```
32      void foo(int num) {
33          try {
34              int result = bar(num);
35              System.out.println("in foo, result: " + result);
36          } catch (LeftException ex) {
37              System.out.println(" --> cought LeftException: " + ex.exNum);
38          } catch (RightException ex) {
39              System.out.println(" --> cought RightException");
40          } catch (CenterException ex) {
41              System.out.println(" --> cought CenterException");
42          } catch (Exception ex) {
43              System.out.println(" --> cought general Exception");
44          }
45          return;
46      }
```

Part 4

This part refers to section "Collection and IO" of the program handout. A program stores objects in an ArrayList, writes these objects to two different files and reads the objects from the file.

4.1 (2 Points)

Define a member `persList` in the class `PersonAdmin`. This member should be typed as a reference to an `ArrayList` that holds objects of type `Person`.

Further complete the Constructor of the class `PersonAdmin` so that the member `persList` actually holds a reference to an `ArrayList`. The `ArrayList` should have an initial capacity of 30.

Complete the program fragment at the ellipsis.

```
47 public class PersonAdmin {
48     ArrayList<Person> persList;
49
50     public PersonAdmin() {
51         persList = new ArrayList<Person>(30);
52     }
```

4.2 (2 Points)

The method call `pao.initializePersonList(3);` in the method `main()` creates random objects an stores them in the `ArrayList persList`. Note: There is no need to analyse and understand the method `initializePersonList()`. The output listing below shows the values of these objects during a certan run of the program. I. e. it shows the output of the first call to `pao.printPersons();`.

The program runs up to line 51 `// - - - 1 - - - .`

Complete the output of the program so far (Note: There might be more doted lines than you will need).

The call `pao.persList.add(1, aPerson)` adds the new person as second element of the list; `pao.persList.remove(3)` removes the fourth (here last) entry of the list.

```
 --- List of persons ---
Bob Mills, born in 1924
Don North, born in 1936
Claire Mills, born in 1920
 --- List of persons ---
Bob Mills, born in 1924
Finn Pony, born in 2015
Don North, born in 1936
```

4.3 (3 Points)

The program creates a buffered output stream to save data into a file. Complete the method `getBufferedInputStream()` at the ellipsis.

```
BufferedOutputStream getBufferedOutputStream(String fileName)
                    throws IOException {
    FileOutputStream fos = new FileOutputStream(fileName);
    BufferedOutputStream bos = new BufferedOutputStream(fos);
    return bos;
}
```

4.4 (7 Points)

The program iterates through the `persList` and uses a `DataOutputStream` to save some data of each object to a file. Method call: pao.savePersToFile("persFile.dat"). The method should work for an arbitrary number of objects inside the `persList`.

It first saves the number of objects to write. Inside the loop it writes the value of the members `yearOfBirth`, `firstName` and `lastName` to the output-stream. Complete the method `savePersToFile()` at the ellipsis.

```
public void savePersToFile(String dataFileName) {
    DataOutputStream dos = null;
    try {
        BufferedOutputStream bos = getBufferedOutputStream(dataFileName);
        dos = new DataOutputStream(bos);
        dos.writeInt(persList.size());
        for (int i = 0; i < persList.size(); i++) {
            Person aPerson = persList.get(i);
            dos.writeShort(aPerson.yearOfBirth);
            dos.writeUTF(aPerson.firstName);
            dos.writeUTF(aPerson.lastName);
        }
    } catch (IOException ex) {
        ex.printStackTrace();
    }
    try {
        dos.close();
    } catch (IOException ex) {
        ex.printStackTrace();
    }
    return;
}
```

4.5 (2 Points)

In line 55 the method `printPersonsToFile()` is called. It writes the data stored in the objects of type `Person` to a file. The file is human readable e. g. in an editor:

```
Bob Mills 1924
. . . . . .
```

Which type of stream do you use to write this file? Fill in the appropriate type and an instruction to create the stream.

```
void printPersonsToFile(String fileName) {
        PrintStream pos = null;
        try {
```

```
        BufferedOutputStream bos = getBufferedOutputStream(fileName);
        pos = new PrintStream(bos);

        for (int i = 0; i < persList.size(); i++) {
            Person aPerson = persList.get(i);
            pos.print(aPerson.firstName);
            pos.print(" ");
            pos.print(aPerson.lastName);
            pos.print(" ");
            pos.println(aPerson.yearOfBirth);
        }
    } catch (IOException ex) {
        ex.printStackTrace();
    }
    if (pos != null) {
        pos.close();
    }
    return;
```

4.6 (3 Points)

Later the progam reads data via a `DataInputStream` from the file and constructs new objects from these data.

First it reads the number of data sets stored in the file. For each data set it reads a short value and two strings. It then creates a new objct of type `Person` feeding the constructor with the values read. Then it adds the object to the ArrayList persList.

Complete the method `readPersFromFile` at the ellipsis.

```
public void readPersFromFile(String dataFileName) {
    DataInputStream dis = null;
    try {
        BufferedInputStream bis = getBufferedInputStream(dataFileName);
        dis = new DataInputStream(bis);
        int persCnt = dis.readInt();
        for (int i = 0; i < persCnt; i++) {
            short yob = dis.readShort();
            String fName = dis.readUTF();
            String lName = dis.readUTF();
            Person aPerson = new Person(fName, lName, yob);
            persList.add(aPerson);
        }
    } catch (IOException ex) {
        ex.printStackTrace();
    }
    try {
        dis.close();
    } catch (IOException ex) {
        ex.printStackTrace();
    }
    return;
}
```

4.7 (1 Point)

What happens during the call `pao.persList.add(10, aPerson)` in line 57?

The method call `persList.add(10, aPerson);` throws an exception since it is not allowed to insert an element at a position greater than `persList.size()`.