Examination in Object Oriented Programming

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

Date, time	February 15th, 2014, 10:30 – 12:00 (90 min)
Number of pages	10 pages (including title)
Resources	All accepted resources

Study Progam	Exam. No.	Room
EI	2608/14/	C004

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	Sum
max.	16	22	17	55
Points				

Note: The given programs contain some methods which you do not need to analyse. Namely the methods `createRandomXXX()`.

Part 1

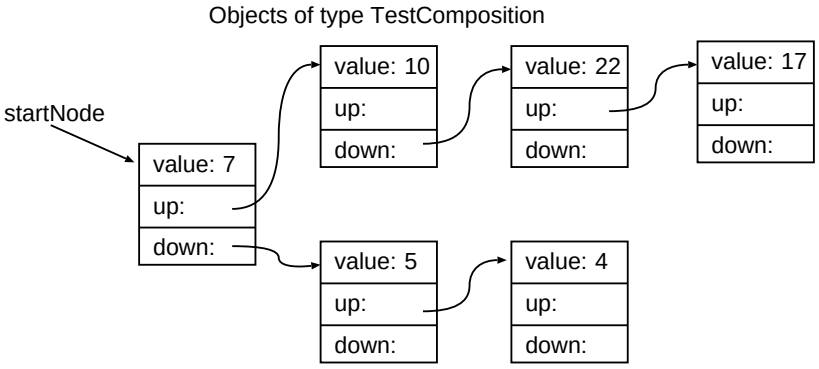
1.1 (5 points)

Given the following program, which data structure results in the main-method afte line 18 is executed?

Complete the given sketch by drawing the missing objects, the missing references and the values stored in the member-variable `value`.

```
1  public class TestComposition {
2      int value;
3      TestComposition up;
4      TestComposition down;
5
6      TestComposition(int value_) {
7          value = value_;
8      }
9
10     public static void main(String[] args) {
11         TestComposition startNode = new TestComposition(7);
12         startNode.down = new TestComposition(5);
13         startNode.up = new TestComposition(10);
14         TestComposition tmpComp = startNode.up;
15         tmpComp.down = new TestComposition(22);
16         tmpComp = tmpComp.down;
17         tmpComp.up = new TestComposition(17);
18         startNode.down.up = new TestComposition(4);
19         return;
20     }
21 }
```

Solution:



1.2 (7 Points)

Analyse the program given in section "Constructors". What is the output of the program?

```
DataObject: id:  99 , name: No name, tmp: 109
DataObject: id:  33 , name: extendedDO, value: 44
DataObject: id:  99 , name: Bob, value: 99
DataObject: id:  99 , name: Anne, tmp: 432 LastDataObject bye bye . . .
```

1.3 (4 Points)

The classes `PlainDataObject`, `ExtendedDataObject`, `AltDataObject` and `lastDataObject` belong to the package `polymorphism`. Look at the class `VeryLastDataObject` given below.

```
1  package collection_io;
2  import polymorphism.LastDataObject;
3
4  public class VeryLastDataObject extends LastDataObject{
5
6      VeryLastDataObject(String name, String msg) {
7          super(msg);
8          dataName = name;
9          value = -1;
10         System.out.print(dataMsg);
11     }
12 }
```

Does this class definition compile? If not which line(s) is/are illegal (X all applicable answers):?

- (X) `super(msg);` The corresponding constructor in LastDataObject has package access (i. e. neither public nor protected).
- ( ) `dataName = name;`
- ( ) `value = -1;`
- (X) `System.out.print(dataMsg);` The member `datamsg` of LastDataObject is private (i. e. neither public nor protected).

Part 2

2.1 (8 points)

Given the program presented in section "Program Array". The program creates objects of type DataObject with random values for the different members and stores these objects in an array (page 3 line 15, 32). It also fills an integer-array with random values (line 15, 28).

Define the class `ExtendedException` as a subclass of `RareException`. The class `ExtendedException` has one member: `int exNumber`. Define two constructors for this class:

The first constructor takes no argument. This constructor should only be callable from classes within the same package and from subclasses of `ExtendedException`

The second constructor takes one argument which is used to set the member. This constructor should be callable from all classes of the program.

An exception of this type is thrown in line 50 of the class `ArrayTest`.

```
public class ExtendedException extends RareException {
    int exNumber;
    public ExtendedException(int value) {
        exNumber = value;
    }
}
```

In the lines 20 to 23 the program catches all possible types of exceptions separatly. Note: A built-in `RuntimeException` might be thrown in the program. Your code might take more than 4 lines (actually it should).

Each catch-clause indicates the type of the cought exception:
`System.out.println("cought . . . Exception");`

Provide the four necessary catch-clauses in a sensible squence.

```
catch (ExtendedException eex) {
    System.out.println("cought ExtendedException");
} catch (RareException rex) {
    System.out.println("cought RareException");
} catch (BasicException bex) {
    System.out.println("cought BasicException");
} catch (Exception ex) {
    System.out.println("cought Exception");
}
```

2.2 (2 Points)

The last method call in the program is in line 18 of Page 3 `ArrayTest.setAndPrint()`.

Complete the last two lines of the output of the program:

```
number[0]: 42
dtObjects[0].idNumber: 55
number[0]: 42
dtObjects[0].idNumber: 88
```

2.3 (6 Points)

Method `void printMatchingDataObjects()` iterates through all values of array `numbers`. For each value in the array `numbers` it calls the method `public void printDoMatchingNumber()` providing the value as argument to the method.

Method `public void printDoMatchingNumber(int num)` iterates through the array `dtObjects` and compares the value of parameter `num` with the value of member `idNumber` for each object in `dtObjects`. If the values are equal the method prints out the object.

Complete the following two methods.

```
public void printMatchingDataObjects() {
    for (int i = 0; i < numbers.length; i++) {
        printDoMatchingNumber(numbers[i]);
    }
    return;
}

public void printDoMatchingNumber(int num) {
    for (int i = 0; i < dtObjects.length; i++) {
        if (dtObjects[i].idNumber == num) {
            System.out.printf("Num: %2d ", num);
            dtObjects[i].print();
        }
    }
    return;
}
```

2.4 (6 Points)

The method `printRecurrentNames()` prints all names of the array `dtObjects` which occure more than once in the array.

Example: The name "Fred" occurs two times in the array so "Fred" is printed out once. If a name occurs more than two times in the array it may be printed out several times (see "Edward" in the sample output at the end of section "Program Array").

Complete the following method.

```
void printRecurrentNames() {
      System.out.print("Recurrent Names: ");
      for (int i = 0; i < dtObjects.length; i++) {
          for (int j = i + 1; j < dtObjects.length; j++) {
              String nameI = dtObjects[i].dataName;
              String nameJ = dtObjects[j].dataName;
              if (nameI.equals(nameJ)) {
                  System.out.print(nameI + ", ");
              }
          }
      }
      System.out.println();
      return;
  }
}
```

Part 3

This part refers to section "Collection and IO" of the program handout. A program stores objects in an ArrayList and writes these objects to a file.

3.1 (2 Points)

Define a member `dataList` in the class `CollectionIO`. This member should be typed as a reference to an `ArrayList` that holds objects of type `ExtendedDataObject` (s. section Program Constructor).

Further complete the Constructor of the class `CollectionIO` so that the member `dataList` actually holds a reference to an `ArrayList`. The `ArrayList` should have an initial capacity of 25.

```
public class CollectionIO {

    ArrayList<ExtendedDataObject> dataList;
    int elemCount;

    public CollectionIO(int elemCount_) {
        dataList = new ArrayList<ExtendedDataObject>(25);
        elemCount = elemCount_;
    }
```

3.2 (3 Points)

The method `getBufferedOutputStream()` takes a file name as argument and returns a buffered output stream which is connected to a file with the given file name. The method should not catch any exception (whats is required instead?).

Complete the code of the method:

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

3.3 (4 Points)

The progam saves all objects of the `dataList` through an `ObjectOutputStream` to a file. The method should work for an arbitry number of objects inside the `dataList`. Which change in the class `ExtendedDataObject` is required to allow this way of writing objects to a file?

Change to `ExtendedDataObject`: . . . . . .

Complete the following programm.

```
public void saveDOtoFile(String osFileName) {
    ObjectOutputStream oos = null;
    try {
        BufferedOutputStream bos = getBufferedOutputStream(osFileName);
        oos = new ObjectOutputStream(bos);
        for (int i = 0; i < dataList.size(); i++) {
            oos.writeObject(dataList.get(i));
        }
    } catch (Exception ex) {
        ex.printStackTrace();
    }
    try {
        oos.close();
    } catch (Exception ex) {
        ex.printStackTrace();
    }
    return;
}
```

3.4 (4 Points)

The progam writes all objects of the `dataList` through a `PrintStream` to a file. The method should work for an arbitry number of objects inside the `dataList`. The file should look the same as the output printed on the screen by the method `printDOs()` i.e. :

```
 --- List of stored objects ---
DataObject: id: . . . . . .
. . . . . .
```

```
    public void printDOsToFile(String printFileName) {
        PrintStream pos = null;
        try {
            BufferedOutputStream bos = getBufferedOutputStream(printFileName)
            pos = new PrintStream(bos);
            pos.println(" --- List of stored objects --- ");
            for (int i = 0; i < dataList.size(); i++) {
                ExtendedDataObject edo = dataList.get(i);
                pos.print("DataObject: id:");
                pos.print(edo.idNumber);
                pos.print(" , name: ");
                pos.print(edo.dataName);
                pos.print(" , value: ");
                pos.println(edo.value);
            }
        } catch (Exception ex) {
            ex.printStackTrace();
        }
        try {
            pos.close();
        } catch (Exception ex) {
            ex.printStackTrace();
        }
        return;
    }
}
```

3.5 (3 Points)

The method call `cio.initializeDataList()` creates random objects an stores them in the `ArrayList dataList`. The output listing at the end of section "Collection and IO" (program handout) shows the values of these objects.

Then the program adds two objects, and prints the resulting list (line 14, 15). After that the program removes two objects an prints the resulting list (line 16, 17).

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

```
 --- List of stored objects ---
DataObject: id:  55 , name: Dan, value: 11
DataObject: id:  92 , name: Berta, value: 14
DataObject: id:  43 , name: Doris, value: 17
 --- List of stored objects ---
DataObject: id:  55 , name: Dan, value: 11
DataObject: id:  92 , name: Berta, value: 14
DataObject: id:  99 , name: Second, value: 2345
DataObject: id:  99 , name: First, value: 1234
DataObject: id:  43 , name: Doris, value: 17
 --- List of stored objects ---
DataObject: id:  55 , name: Dan, value: 11
DataObject: id:  99 , name: First, value: 1234
DataObject: id:  43 , name: Doris, value: 17
Exception in thread "main" java.lang.IndexOutOfBou . . .
  . . .
```

3.6 (1 Point)

What happens during the call `cio.insertDOs(20)` in line 20?

. . . . . .

. . . . . .

The method call `dataList.add(pos, newDo);` inside of `cio.insertDOs()` throws an exception since it is not allowed to insert an element at a position greater than `dataList.size()`.