Examination in Object Oriented Programming

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

Date, time	19 July 2016, 10:30 – 12:00 (90 min)
Number of pages	15 pages (including title)
Resources	All accepted resources

Study Program	Room
AI	H069
EI	C004
MD	D002

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	5	Sum
max.	5	19	9	26	6	65
Points						

Part 1

1.1 (5 Points) Exception Handling

Analyse the program given in section **Exception** of the handout "Programs and JDK-Documentation".

The program defines some exception classes and a main class. Method `foo()` may throw any of the exceptions defined in the program a well as an ArithmeticException an another runtime-exception.

What is a proper sequence of catch clauses in order to catch each of these exceptions separately.

```
catch ( . . . . . ex) {
        System.out.println(ex);
```

The catch-clauses should just print out the caught exception. Fill in the catch clauses in a proper sequence:

```
try{
      foo(i);
    }
```

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

Part 2

2.1 (15 Points)

Analyse the program given in section **Constructors** of the handout "Programs and JDK-Documentation". All classes of this program are defined in the same package `working`. The method **public void** testProfessions(); is defined in a class of this package.

When answering the questions please keep in mind that there might be more dotted lines than actually needed (this holds for all questions).

```
67      public void testProfessions() {
68          System.out.println(" 1 ----------");
69          Profession aProfession = new Profession();
70          System.out.println(" 2 ----------");
71          aProfession.work();
72          System.out.println(" 3 ----------");
73          Engineer anEngineer = new Engineer();
74          System.out.println(" 4 ----------");
75          anEngineer.work();
76          System.out.println(" 5 ----------");
77          anEngineer.designSystem();
78          System.out.println(" 6 ----------");
79          SoftwareEngineer aSwEngineer = new SoftwareEngineer(7);
80          System.out.println(" 7 ----------");
81          aSwEngineer.work();
82          System.out.println(" 8 ----------");
83          aSwEngineer.designSystem();
84          System.out.println(" 9 ----------");
85          aSwEngineer.coding();
86          System.out.println("10 ----------");
87          Profession aPerson = new Artist();
88          System.out.println("11 ----------");
89          aPerson.work();
90          // aPerson.getPhone();
91          // Artist anArtist = new Artist();
92          // anArtist.setPhone("0751 501 . . .");
93          // String aPhone = anArtist.phone;
94          // int aLevel = anArtist.level;
95      }
96  }
```

What is the output of the program? Fill in the output step by step. Note: There might be more doted lines than needed.

What is the output of the program when line 69 Profession aProfession = **new** Profession(); is executed?

```
 1 -------------------
. . . . . .
. . . . . .
```

What is the output of the program when line 71 aProfession.work(); is executed?

```
 2 -------------------
. . . . . .
. . . . . .
```

What is the output of the program when line 73 Engineer anEngineer = **new** Engineer(); is executed?

```
 3 -------------------
. . . . . .
. . . . . .
```

What is the output of the program when line 75 anEngineer.work(); is executed?

```
 4 -------------------
. . . . . .
. . . . . .
```

What is the output of the program when line 77 anEngineer.designSystem(); is executed?

```
 5 -------------------
. . . . . .
. . . . . .
```

What is the output of the program when line 79 SoftwareEngineer aSwEngineer = **new** SoftwareEngineer(7 is executed?

```
 6 -------------------
. . . . . .
. . . . . .
. . . . . .
. . . . . .
```

What is the output of the program when line 81 aSwEngineer.work(); is executed?

```
 7 -------------------
. . . . . .
. . . . . .
. . . . . .
```

What is the output of the program when line 83 aSwEngineer.designSystem(); is executed?

```
 8 -------------------
. . . . . .
. . . . . .
```

What is the output of the program when line 85 aSwEngineer.coding(); is executed?

```
 9 -------------------
. . . . . .
. . . . . .
```

What is the output of the program when line 87 Profession aPerson = **new** Artist(); is executed?

```
10 -------------------
. . . . . .
. . . . . .
```

What is the output of the program when line 89 aPerson.work(); is executed?

```
11 -------------------
. . . . . .
. . . . . .
```

2.2 (4 Points)

Now we add some code to the class `Artist` and we uncomment line 90 – 94 of method **public void** testProfessions();

```
public class Artist extends Profession {
    public void work() {
        perform();
        System.out.println(", enjoying applause, ");
    }
    public void perform(){
        System.out.print("performing, " + level);
    }

    private String phone = "+49 172 . . . ";

    public String getPhone() {
        return phone;
    }
    void setPhone(String phone) {
        this.phone = phone;
    }
}
```

```
67      public void testProfessions() {
68          System.out.println(" 1 ----------------------");
69                  :
70                  :

87          Profession aPerson = new Artist();
88          System.out.println("11 ----------------------");
89          aPerson.work();
90          aPerson.getPhone();
91          Artist anArtist = new Artist();
92          anArtist.setPhone("0751 501 . . .");
93          String aPhone = anArtist.phone;
94          int aLevel = anArtist.level;
95      }
96  }
```

Some of the new lines do not compile. Indicate which lines will be rejected by the compiler. Note: You receive a point for each correct indication; for each wrong indication one point is deducted. If you can't decide, cross "don't know" – no point is added or deducted. The minimum score is zero points even if you marked less correct answers than wrong ones.

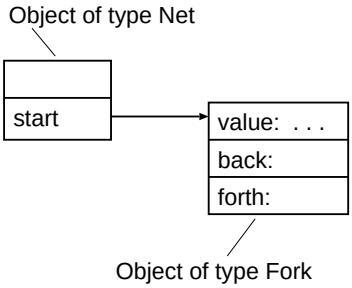
Code	correct	illegal	don't know
aPerson.getPhone();			
anArtist.setPhone("0751 501 . . .");			
String aPhone = anArtist.phone;			
int aLevel = anArtist.level;			

Part 3

Analyse the program given in section **Tree** of the handout "Programs and JDK-Documentation".

3.1 (5 Points)

What data structure results in the program after after line 21 (branch_1.right = branch_4;) is executed? Complete the given sketch by drawing the missing objects, the missing references and the values stored in the member-variable `value`.



3.2 (4 Points)

The method `print()` of class `Tree` traverses any data structure build of objects of type `Branch`. It calls the method `print()` of each object of type `Branch`. Complete the `print()` of class `Tree`.

Hint: Implement the method recursively. It first checks if thevalue of the parameter is `null`. If so, it just returns. If the parameter references an object, it calls the method `print()` of the referenced object. Then it calls itself using the values of member `left` and `right` of the referenced object respectively.

```
void printTree(Branch aBranch){
    . . . . . .
    . . . . . .
    . . . . . .
    . . . . . .
    . . . . . .
    . . . . . .
    . . . . . .
    . . . . . .
}
```

Part 4

This part refers to section **Collections and IO** of the handout "Programs and JDK-Documentation". A program creates objects of type `Truck`, stores them in an ArrayList, writes them to a file and reads the objects from the file.

4.1 Print Truck (3 Points)

The method `print()` of class `Truck` prints the data stored in an object of class `Truck` to the screen. It first prints the member `identifier`, then `capacity` and lastly `stateCode`.

A sample output of the method `print()` :

```
Truck: Renault   , 1622,59, 125
```

Complete the method print() at the ellipsis.

```
void print() {
    System.out.print("Truck: "  . . . . . .
    . . . . . .
    . . . . . .
}
```

4.2 File IO

4.2.1 (3 Points)

The method `getDataOutputStream()` takes the name of a file as argument and returns an object of type `DataOutputStream` to write to the file. Note: The stream should be buffered.

Complete the method getDataOutputStream() at the ellipsis.

```
DataOutputStream getDataOutputStream(String fileName) throws IOException {
    FileOutputStream fos =  . . . . . .
    . . . . . .
    DataOutputStream dos =  . . . . . .
    return dos;
}
```

4.2.2 (4 Points)

The method `saveTruckToStream()` takes an objekt of type `DataOutputStream` and an objekt of type `Truck` as arguments. It writes the data stored in the object of class `Truck` to the DataOutputStream. It first writes the member `identifier`, then `capacity` and lastly `stateCode`.

Complete the method saveTruckToStream() at the ellipsis.

```
void saveTruckToStream(Truck aTruck, DataOutputStream dos) throws IOException
        . . . . . .
        . . . . . .
        . . . . . .
}
```

4.2.3 (3 Points)

The method `getDataInputStream()` takes the name of a file as argument and returns an object of type `DataInputStream` that allows for reading data from the file. Note: Reading from the file should be buffered.

Complete the method getDataInputStream() at the ellipsis.

```
DataInputStream getDataInputStream(String fileName) throws IOException {
    FileInputStream fis = . . . . . .
    . . . . . .
    DataInputStream dis = . . . . . .
    return dis;
}
```

4.2.4 (3 Points)

The method `readTruckFromStream()` takes an objekt of type `DataInputStream` as argument and returns an object of type `Truck` with data read from the stream.

Complete the method readTruckFromStream() at the ellipsis.

```
Truck readTruckFromStream(DataInputStream dis) throws IOException {
    Truck aTruck = new Truck();
    . . . . . .
    . . . . . .
    . . . . . .
    return aTruck;
}
```

4.3 Storing Data in an ArrayList

4.3.1 (4 Points)

The class `FileIoTest` defines a member `truckList`. It is a reference to an object of type `ArrayList` for storing objects of type `Truck`.

Fill in the the proper data type for `truckList` and provide the code to create the object of type `ArrayList`. The initial capacity of the `ArrayList` should be 20 .

```
public class FileIoTest {
    String truckFileName = "truckDataFile.dat";

    . . . . . truckList;
    TruckIO truckIoMgr;

    FileIoTest() {
        truckList = . . . . . .
        truckIoMgr = new TruckIO();
    }
```

4.3.2 (6 Points)

The following excerpt from the listing shows the code that deals with the ArrayList. For the complete code see section **Collection and IO** of the handout "Programs and JDK-Documentation".

The method `main()` creates trucks and stores them in an ArrayList `truckList`. Then it partially removes them from the ArrayList, reads in trucks stored in the file and adds these trucks to the ArrayList. In between the methods prints the list of trucks to the screen. Complete the output of this method on the next page.

Note: You don not need to analyse the method `createRandomTrucks()`. It is sufficient to regard the result (see next page).

```
void main() {
        :
    fiot.createRandomTrucks(5); // this method populates the truckList
    fiot.printTrucks();         // the output of this call is given below
        :                       // saving trucks to file (see saveTrucks())
    fiot.truckList.remove(1);
    fiot.truckList.remove(1);
    fiot.printTrucks();  // what is the output of this call?
        :
    fiot.readTrucks();   // adds the trucks read from the file
                         // see line 40 and line 41 of the program
        :
    printTrucks();       // what is the output of this call?
}
```

Complete the output of the method main(). Note: It is sufficient to write the identifier of the trucks e. g. `Scania`, you may omit the printout of other members. There might be more doted lines than you will need.

```
Trucklist:
Truck: Mitsubishi, 2324,65, 117
Truck: Mercedes  , 1964,49, 121
Truck: Toyota    , 1286,73, 114
Truck: Scania    , 2885,94, 110
Truck: Renault   , 2769,36, 107
--------------------

Trucklist:

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

--------------------

Trucklist:

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .
```

Part 5

Analyse the program given in section **Class Design** of the handout "Programs and JDK-Documentation". All given classes are defined in the same package.

5.1 Class Design

Modify the program in order to simplify the class `LetFlyExample`. The output of the program should remain the same but it should take less code (in class `LetFlyExample`) to produce it.

Hint: Look at line 22 – 24 and 44 – 46. You may introduce a new class.

5.2 New class (2 Points)

If you introduce a new class, define here:

class

}

5.3 Changes to classes (1 Point)

How do you change the classes `Eagle`, `Condor` and `Flamingo`? Just sketch the changes not the complete classes.

public class Eagle

}

public class Condor

}

public class Flamingo

}

5.4 Changes to class LetFlyExample (3 Points)

How do you change the classes `LetFlyExample`? Just sketch the changes not the complete class.

public class LetFlyExample

}