Examination in Object Oriented Programming

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

Date, time	July 22nd, 2015, 10:30 – 12:00 (90 min)
Number of pages	11 pages (including title)
Resources	All accepted resources

Study Progam	Room
AI	C004, D002
EI	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	4	Sum
max.	15	5	20	14	54
Points					

Examination OOP S 2015   Prof. Dr. M. Zeller   Page 2 (11)

Name: Mat. no: July 22nd, 2015

Part 1

1.1 (11 Points)

Analyse the program given in section **Constructors** of the handout "Programs and JDK-Documentation". Please keep in mind that there might be more dotted lines than actually needed (this holds for all questions).

What is the output of the program just after line 45 `SailingBoat mayFlower = new SailingBoat();` is executed?

What is the additional output of the program just after line 46 `mayFlower.sail()` is executed?

What is the additional output of the program just after line 47 `Watercraft bounty = new SailingBoat();` is executed?

What is the additional output of the program just after line 48 `bounty.move()` is executed?

Part 1

Examination OOP S 2015   Prof. Dr. M. Zeller   Page 3 (11)

Name: Mat. no: July 22nd, 2015

1.2 (3 Points)

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

```
class Sails {
    private float area = 30;
    Sails() {
        System.out.println("Creating Sails: ");
    }
    public float getArea(){
        return area;
    }
}
```

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

. . . .

Change the method `sail()` so that it can be compiled and it achieves the same effect as the original method, leave class `Sails` unchanged.

```
    void sail(){



    }
```

1.3 (1 Point)

Now we add a method call `bounty.sail()` in the method `main()`

```
public static void main(String[] args) {
        SailingBoat mayFlower = new SailingBoat();
        mayFlower.sail();
        Watercraft bounty       = new SailingBoat();
        bounty.move();
        bounty.sail();    // <== new method call
    }
```

Does this method call compile, and if so whats the output of this method call?

. . . .

. . . .

Part 2

2.1 (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 a runtime-exception.

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

```
catch ( . . . . . ) {
      System.out.println(" --> cought xxxException: ");
```

Replace xxx by "Runtime", "South", "NorthWest" and"North"'. Fill in the catch clauses in the (in a) proper sequence:

```
try {
      int result = foo(1);
}
```

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

Examination OOP S 2015   Prof. Dr. M. Zeller   Page 5 (11)

Name: Mat. no: July 22nd, 2015

Part 3

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

3.1 (4 Points)

In class `PlayList` the method `void addSong(Song aSong)` assigns a new value (a reference) to the member `songList`. The previously stored reference should be added to the new song in order to form a linked list as shown in the sketch of the data structure. Note: The songs are added in the following sequence: `Hip_0`, `Hop_1`, `Rap_2`, `Zap_3` and lastly `House_4`.

Complete the method `void addSong(Song aSong)`:

```
public void addSong(Song aSong) {




}
```
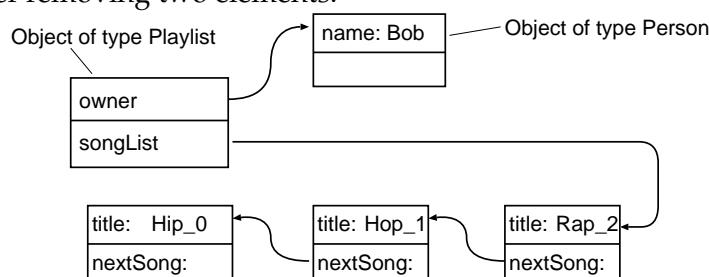
3.2 (4 Points)

The method `public Song removeSong()` removes the first element of the `songList` and returns the removed element. If the `songList` is empty the method just returns `null`.

Complete the method `public Song removeSong()`:

```
public Song removeSong() {





}
```

Datastructure after removing two elements:

Examination OOP S 2015   Prof. Dr. M. Zeller   Page 6 (11)

Name: Mat. no: July 22nd, 2015

3.3 (6 Points)

The method `public void play()` of class `PlayList` prints the name of the owner of the play-list and all titles of all songs on the song list. (A sample output of the program is given at the end of section **Playlist**.) To do so, it may (should) call other methods given in the program. The method must not change any data in the visited objects.

The method does not throw an exception caused by a null-reference. If it encounters a null-reference it simply does not print out anything related to the reference.

```
public void play () {
    System.out.println("\n ————— playing playlist —————" );
```

```
}
```

Examination OOP S 2015   Prof. Dr. M. Zeller   Page 7 (11)

Name: Mat. no: July 22nd, 2015

3.4 (6 Points)

The method `public void play(int index)` of class `PlayList` prints the title of one song of the song list (song no. index + 1). E. g. if `index` has the value 3 it prints the title of the 4th song of the list.

(See also the sample output of the program given at the end of section **Playlist**.) The method must not change any data in the visited objects.

The method does not throw an exception caused by a null-reference. If it encounters a null-reference it simply does not print out anything related to the reference.

```java
public void play(int index) {
        System.out.println("\nplaying song " + index + " ————————");
```



```
}
```

Examination OOP S 2015   Prof. Dr. M. Zeller   Page 8 (11)

Name: Mat. no: July 22nd, 2015

Part 4

This part refers to section **Collection and IO** of the handout "Programs and JDK-Documentation". A program writes objects of type `Customer` to a file, reads objets from a file and stores objects in a HashMap.

4.1 Writing to a file

4.1.1 (2 Points)

The method `getDataOutputStream()` creates a buffered output stream (of type `DataOutputStream`) to save data into a file. The program stops if it can not create the stream. Complete the method `getDataOutputStream()` at the ellipsis.

```
DataOutputStream getDataOutputStream(String fileName) {
    DataOutputStream dos = null;
    try {
        FileOutputStream fos = . . . . . . . . . . . . . . .
        BufferedOutputStream bos = . . . . . . . . . . . . . .
        dos = . . . . . . . . . . . . .
    } catch (IOException ioex) {
        ioex.printStackTrace();
    }
    if ( . . . . . . . . . . . . . . . . ) {
        System.out.println("could not get OutputStream: " + fileName);
        System.exit(-1);
    }
    return dos;
}
```

4.1.2 (2 Points)

The method `saveCustomer()` writes the data of an object of type `Customer` to a `DataOutputStream`. Complete the method `saveCustomer()` at the ellipsis.

```
void saveCustomer(Customer aCustomer, DataOutputStream dos) throws IOException {
    dos . . . . . . . . . . . . . . . . . . . . . . . .
    dos . . . . . . . . . . . . . . . . . . . . . . . .
    dos . . . . . . . . . . . . . . . . . . . . . . . .
}
```

Examination OOP S 2015   Prof. Dr. M. Zeller   Page 9 (11)

Name: Mat. no: July 22nd, 2015

4.2 Reading from a stream

4.2.1 (2 Points)

The method `getDataInputStream()` creates a buffered output stream (of type DataInput-Stream) to read data from a file. The program stops if it can not create the stream. Complete the method `getDataInputStream()` at the ellipsis.

```
DataInputStream getDataInputStream(String fileName) {
    DataInputStream dis = null;
    try {
        FileInputStream fis =  . . . . . . . . . . . . . . . .
        BufferedInputStream bis =  . . . . . . . . . . . . . . .
        dis =  . . . . . . . . . . . . . .
    } catch (IOException ioex) {
        ioex.printStackTrace();
    }
    if ( . . . . . . . . . . . . . . . . . ) {
        System.out.println("could not get InputStream: " + fileName);
        System.exit(-1);
    }
    return dis;
}
```

4.2.2 (2 Points)

The method `readCustomer()` reads the data of an object of type `Customer` from a `DataInputStream`. After reading the data it creates an object of type `Customer`. The new object contains the data read from the stream. If the method could not read all data, it returns `null`. Complete the method `readCustomer()` at the ellipsis.

```
Customer readCustomer(DataInputStream dis) {
    Customer aCustomer = null;
    try {
        int theId        = . . . . . . . . . . . . . . .
        String theName   = . . . . . . . . . . . . . . .
        float theVolume  = . . . . . . . . . . . . . .
        aCustomer        = . . . . . . . . . . . . . . .
        System.out.println("id: " + theId);
    } catch (EOFException ex) {
        System.out.println("done reading file: ");
    }
    catch (IOException ioex) {
        ioex.printStackTrace();
    }
    return aCustomer;
}
```

Examination OOP S 2015   Prof. Dr. M. Zeller   Page 10 (11)

Name: Mat. no: July 22nd, 2015

4.3 Storing Data in a HashMap

4.3.1 (2 Points)

The class `PersonDB` defines a private variable `customerMap`. It is a HashMap for storing objects of type `Customer`. The key to access an object is of type `Integer`.

Fill in the the proper data type for `customerMap` and provide the code to create the HashMap. The initial capacity of the HashMap should be 100.

```
. . . . . . . . . . . .  customerMap;  // provide  proper  data  type
    :
    :
void  testCustomerIO() {
    custIO.writeCustomerFile(5, fileName);
    customerMap = new . . . . . . . . . . . . . . . .  // create  HashMap
```

4.3.2 (2 Points)

The method `readCustomerFile()` receives a file name and a HashMap. It opens a stream, reads objects of type `Customer` from the stream and stores them into the HashMap. The value of the member `customerId` is the key of the HashMap.

Complete the method `readCustomerFile()` at the ellipsis.

```
void readCustomerFile(String fileName, . . . . . . . . . . . . .  customerMap) {
    try (DataInputStream dis = getDataInputStream(fileName)) {
        Customer aCustomer;
        while ((aCustomer = readCustomer(dis)) != null) {

            customerMap. . . . . . . . . . . . . . . . . . ;  // store  aCustomer
        }                                                      // in  customerMap
    } catch (Exception ex) {
        ex.printStackTrace();
    }
    return;
}
```

Examination OOP S 2015   Prof. Dr. M. Zeller   Page 11 (11)

Name: Mat. no: July 22nd, 2015

4.3.3 (2 Points)

The method `testCustomerIO()` stores, retrieves and prints some variables of type `Customer`. When the program reaches line 124 the HashMap `customerMap` contains mappings for the keys: 70529, 48372 and 35830.

```
118      void testCustomerIO() {
119              :
120              :
121         custIO.readCustomerFile(fileName, customerMap);
122              :
123              :
124         Customer aCustomer = custIO.createRandomCustomer();
125         aCustomer.print();
126         customerMap.put(12345, aCustomer);
127         aCustomer = custIO.createRandomCustomer();
128         aCustomer.print();
129         customerMap.put(12345, aCustomer);
130         aCustomer = customerMap.get(12345);
131         aCustomer.print();
132         aCustomer = customerMap.get(1111);
133      }
```

The output of the program starts as follows.

Output of line 125:
```
 ID: 61960, Name: Don, Revenue: 4774.1045
```

Output of line 128:
```
 ID: 86494, Name: Chris, Revenue: 4137.799
```

What is the output of line 131:

. . . .

What happens in line 132: