

Examination in Object Oriented Programming

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

Date, time 13 February 2016, 10:30 – 12:00 (90 min)
Number of pages 12 pages (including title)
Resources All accepted resources

Study Program	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	5	Sum
max.	5	21	8	26	5	65
Points						

Part 1

1.1 (5 Points) Exception Handling

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

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

Note: The clauses for catching the `ArithmeticException` can take any position except the last.

```
try{  
    foo(i);  
} catch (TimeoutException toe) {  
    System.out.println(toe);  
} catch (MoveException me) {  
    System.out.println(me);  
} catch (GameException ge) {  
    System.out.println(ge);  
} catch (ArithmeticException ae) {  
    System.out.println(ae);  
} catch (Exception ex) {  
    System.out.println(ex);  
}
```

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 **gaming**. The method **public void testGames();** 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).

```
52 public void testGames () {
53     Game aGame = new Game();
54     BoardGame aBoardGame = new BoardGame();
55     Chess aChessGame = new Chess();
56     System.out.println("_____");
57     aGame.start();
58     aBoardGame.start();
59     aChessGame.start();
60     System.out.println("_____");
61
62     aBoardGame.placeBoard();
63     aChessGame.placeBoard();
64     aChessGame.placeChessmen();
65
66     Game theGame = new CardGame();
67     theGame.start();
68 }
```

What is the output of the program when line 53 `Game aGame = new Game();` is executed?

Creating a game

What is the output of the program when line 54 `BoardGame aBoardGame = new BoardGame();` is executed?

Creating a game

Creating a board game, 3

What is the output of the program when line 55 `Chess aChessGame = new Chess();` is executed?

Creating a game

Creating a board game, 3

creating a chess game

What is the output of the program when line 57 `aGame.start();` is executed?

`starting: Game of life, 3`

What is the output of the program when line 58 `aBoardGame.start();` is executed?

`starting: board game, 3`

What is the output of the program when line 59 `aChessGame.start();` is executed?

`Chess: white's move`

What is the output of the program when line 62 `aBoardGame.placeBoard();` is executed?

`placing board for game: board game`

What is the output of the program when line 63 `aChessGame.placeBoard();` is executed?

`placing board for game: Chess`

What is the output of the program when line 64 `aChessGame.placeChessmen();` is executed?

`placing chessmen`

What is the output of the program when line 66 `Game theGame = new CardGame();` is executed?

`Creating a game`

What is the output of the program when line 67 `theGame.start();` is executed?

`shuffle cards, starting card game, 3`

What happens if we add a line `theGame.shuffle();` just after line 67. Does this line compile, if so, what is the output of this line?

The line does not compile since the method `shuffle()` is not defined in class `Game`.

2.2 (6 Points)

Now extend the class **Chess** as follows: The class should contain a member **timeLimit** of type **int**. All classes of the program should be able to obtain the value of this member. All classes of the package **gaming** (the package of class **Chess**) should be able to change the value of this member. Classes outside the package are not allowed to change the value of **timeLimit**. The class **Chess** should ensure that the value of the member is at least 10.

Add the member **timeLimit** and some more code if required. Just write down the new code, do not reproduce any code given in the handout "Programs and JDK-Documentation".

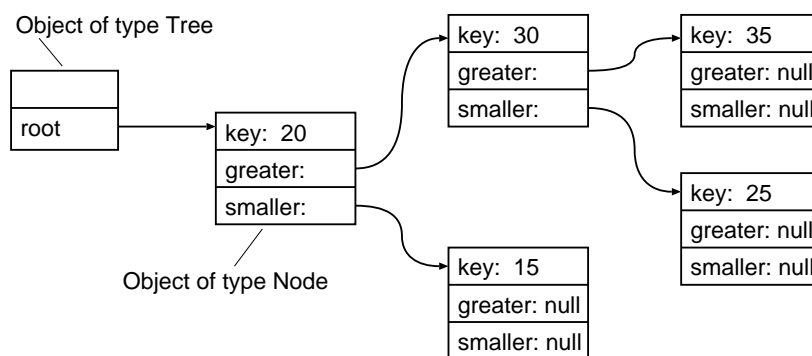
```
public class Chess extends BoardGame {  
  
    :  
  
    private int timeLimit = 600;  
  
    public int getTimeLimit() {  
        return timeLimit;  
    }  
  
    void setTimeLimit(int newTimeLimit) {  
        timeLimit = newTimeLimit;  
        if (timeLimit < 10) {  
            timeLimit = 10;  
        }  
    }  
  
    :  
}
```

Part 3

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

3.1 (8 Points)

What data structure results in the program after after line 30 (`node_3.greater = node_2;`) is executed? Complete the given sketch by drawing the missing objects, the missing references and the values stored in the member-variable **key**.



Part 4

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

4.1 Print Toy (3 Points)

The method `print()` of class `Toy` prints the data stored in an object of class `Toy` to the screen.

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

Toy: R-Cube, 31427, 1958.7777

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

```
void print() {  
    System.out.print("Toy: " + name);  
    System.out.print(", " + colorCode);  
    System.out.println(", " + weight);  
}
```

4.2 File IO

4.2.1 (3 Points)

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

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

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

4.2.2 (4 Points)

The method `printToyToStream()` takes an object of type `PrintStream` and an object of type `Toy` as arguments. It prints the data stored in the object of class `Toy` to the `PrintStream`.

Three sample lines of the file produced by the method `printToyToStream()` :

```
31427 R-Cube 1958.7777  
10023 Robot 805.5197  
9985 Barbie 1292.639
```

Note: Each Toy is printed on a new line, the member values are separated by a space.

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

```
void printToyToStream(Toy aToy, PrintStream ps) {  
    ps.print(aToy.colorCode + " ");  
    ps.print(aToy.name + " ");  
    ps.println(aToy.weight);  
}
```

4.2.3 (3 Points)

The method `getFileScanner()` takes the name of a file as argument and returns an object of type `Scanner` that reads and scans data from the file. Note: Reading from the file should be buffered.

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

```
    Scanner getFileScanner(String fileName) throws IOException {  
        FileInputStream fis = new FileInputStream(fileName);  
        BufferedInputStream bis = new BufferedInputStream(fis);  
        Scanner fs = new Scanner(bis);  
        fs.useLocale(Locale.US); // depending on the data format  
        return fs;  
    }
```

4.2.4 (3 Points)

The method `readToy()` takes a scanner as argument and returns an object of type `Toy` with data delivered by the scanner.

Note: The required code may take three lines or up to six lines depending on the programming style.

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

```
Toy readToy(Scanner aScanner) {  
    Toy aToy = new Toy();  
    int colorCode = aScanner.nextInt();  
    String name = aScanner.next();  
    float weight = aScanner.nextFloat();  
    aToy.colorCode = colorCode;  
    aToy.name = name;  
    aToy.weight = weight;  
    return aToy;  
}
```

4.3 Storing Data in an ArrayList

4.3.1 (4 Points)

The class `FileIoTest` defines a member `toyList`. It is an object of type `ArrayList` for storing objects of type `Toy`.

Fill in the the proper data type for `toyList` and provide the code to create the object of type `ArrayList`. The initial capacity of the `ArrayList` should be 10 .

```
ArrayList<Toy> toyList ;
```

```
FileIoTest() {  
    toyList = new ArrayList<>(10);  
}
```

4.3.2 (6 Points)

The method `makeToys()` creates toys, stores and (partially) removes them from the `ArrayList` `toyList`.

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-Docmentation".

Complete the output of the method `makeToys()`. Note: There might be more dotted lines than you will need.

```
Toylist:
Toy: Robot, 8649, 844.9267
Toy: Train, 3460, 320.32697
Toy: R-Cube, 19013, 192.16956
Toy: Lego, 26790, 1081.4529
-----
Toylist:
Toy: Robot, 8649, 844.9267
Toy: Train, 3460, 320.32697
-----
Toylist:
Toy: Robot, 8649, 844.9267
Toy: Train, 3460, 320.32697
Toy: Robot, 8649, 844.9267
Toy: Train, 3460, 320.32697
Toy: R-Cube, 19013, 192.16956
Toy: Lego, 26790, 1081.4529
-----
```

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 `geometry`.

5.1 Class Design

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

Hint: Look at line 27 – 29 and 39 – 41. You may introduce a new class.

5.2 New class (2 Point)

If you introduce a new class, define here:

```
public abstract class Shape {  
    public void draw() {  
        :  
    }  
}
```

5.3 Changes to classes (1 Point)

How do you change the classes `circle`, `line` and `rectangle`? Just sketch the changes not the complete classes.

```
public class Circle extends Shape {  
    :  
}  
  
public class Line extends Shape {  
    :  
}  
  
public class Rectangle extends Shape {  
    :  
}
```

5.4 Changes to class GeometryTest (2 Point)

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

```
public class GeometryTest {  
  
    int objCount = 15;  
    Shape[] shapes = new Shape[objCount];  
    :  
  
    for (int i = 0; i < objCount; i++) {  
        shapes[i].draw();  
    }  
    :  
}
```