TEST

Software Systems: Programming

course code: 202001024
date: 15 January 2021
time: 9:00 – 12:00

General

- You may use the following (unmarked) materials when making this test:
  - Module manual.
  - Slides of the Programming topics.
  - The book
    David J. Eck. *Introduction to Programming Using Java*. Version 8.1.2, December 2020.
  - A dictionary of your choice.
  - IntelliJ and/or Eclipse.
  - Documentation of Java 11
    `https://docs.oracle.com/en/java/javase/11/docs/api/`

  You are *not* allowed to use any of the following:

  - Solutions of any exercises published on Canvas (such as recommended exercises or old tests);
  - Your own materials (copies of (your) code, solutions of lab assignments, notes of any kind, etc.).
- When you are asked to write Java code, follow code conventions where they are applicable. Failure to do so may result in point deductions. It is recommended that you use IntelliJ and/or Eclipse to write code.
- You do *not* have to add Javadoc or comments, unless explicitly asked to do so. Invariants, preconditions and postconditions should be given only when they are explicitly asked.
- You are not allowed to leave the room during the first 30 minutes or the last 15 minutes of the exam.
- Place your student ID card on the table as well as documentation that grants extra time (if applicable).

Question 1 (4 points)

a. *(2 pts)* Explain the general difference between compile-time errors and runtime errors.

b. *(1 pts)* Give an example of a compile-time error

c. *(1 pts)* Give an example of a runtime error

Question 2 (4 points)

a. *(2 pts)*

What is printed if we compile and run the following code fragment:

```
int a = 1;
int b = 1;
System.out.println(a + "+" + b + "=" = a + b);
```

(a) `1+1=11`
(b) `1+1=1+1`
(c) There will be a compilation error
(d) There will be a runtime error

Explain your answer.

b. *(2 pts)* The intended result was `1+1=2`. Explain how to change the code to get this result, without explicitly using the numbers 1 and 2.

Question 3 (4 points)

Consider the following Java classes:

```
public class A {
    protected int value = 12;
    protected A() {
    }
    public A(int value) {
        value = value;
    }
    protected int getValue() {
        return value;
    }
}

public class B extends A {
    private B() {
        value = 9;
    }
    public B(int value) {
        super(value);
    }
    public int getValue() {
        return super.getValue() + 1;
    }
}
```

Consider also the following Java code:

```
B b = new B(10);
System.out.println("Value: " + b.getValue());
```

What is printed when compiling and executing the given code fragment? Explain your answer.

a. `Value: 9`
b. `Value: 10`
c. `Value: 11`
d. `Value: 12`
e. `Value: 13`
f. Nothing, there will be an error.

Question 4 (4 points)

Consider the following Java classes:

```
public interface A {
    public int a();
}
public class ExampleClass extends A {
    private ExampleClass() {
    }
    public int a() {
        return b();
    }
    private static int b() {
        return c();
    }
    protected final int c() {
        return 4;
    }
    public final void main(String[] args) {
        ExampleClass.a();
        A ex = new ExampleClass();
        ex.a();
    }
}
```

This code fragment has three compile-time and/or runtime errors. Explain these errors by referring to what is wrong, why that is wrong, and how it could be fixed.

Question 5 (4 points)

Model-View-Controller is a well known design pattern. Imagine we program a tic tac toe game. Explain for the following examples whether they belong to the view, to the controller, or to the model, and why.

a. *(1 pt.)* A ButtonEventListener that receives the event that someone clicked one of the empty spaces on the game board.
b. *(1 pt.)* A TicTacToeCanvas that displays the current state of the game.
c. *(1 pt.)* A TicTacToeBoard that stores the current game board.
d. *(1 pt.)* An algorithm that checks whether someone won the game.

Question 6 (15 points)

Suppose we have a class hierarchy with the class `Appliance`, which has subclasses `Table` and `Chair` and `Bookcase`. The `Bookcase` class has methods `addBook(Book book)`, `removeBook(Book book)`. The `Table` class has methods `addItem(Item item)` and `removeItem(Item item)`. Both `Appliance` and `Book` implement the interface `Item`.

Consider we have the following code:

```
Table t = new Table();
Appliance b = new Bookcase();
Book book = new Book();
```

a. *(6 pts)* For each of the following lines, determine whether or not they give a runtime error and/or a compilation error and give your explanation.
   (a) `t.addBook(book);`
   (b) `Bookcase bc = b;`
   (c) `((Table) b).addItem(book);`

b. *(6 pts)* Imagine we now want chairs to have a color. We give three methods to accomplish this. For each of these methods, explain when it would be appropriate to use that method.
   (a) Add a `color` field to the `Appliance` class and appropriate getters and setters.
   (b) Add a `color` field to the `Chair` class and appropriate getters and setters.
   (c) Extend `Chair` by a `ColoredChair` subclass which has a `color` field and appropriate getters and setters.

c. *(3 pts)* Your colleague has implemented colored chairs using (empty) interfaces `Red`, `Blue`, etc, and subclasses **`class`** `RedChair` **`extends`** `Chair` **`implements`** `Red`, etc. Your colleague explains that you can check the color of a chair using the **`instanceof`** keyword. There is however an important difference between this solution and the other solution. What can you do with the solutions of subquestion b that you cannot do with this solution?

Question 7 (10 points)

A mathematical number sequence to rationally approximate $\sqrt{2}$ is the Pell sequence:
- $P_0 = 1$
- $P_1 = 2$
- $P_n = 2P_{n-1} + P_{n-2}$

The first Pell numbers are 1, 2, 5, 12, 29, 70, 169, ... The fraction $\frac{P_n + P_{n+1}}{P_{n+1}}$ approximates $\sqrt{2}$.

a. *(5 pts)* Implement a method **`long`** `pellRecursive(`**`int`** `n)` that computes and returns the Pell number n. This method must be **recursive**.

b. *(5 pts)* Implement a nonrecursive method **`long`** `pellArray(`**`int`** `n)` that computes and returns the Pell number n. This method **must not** call any methods. Instead, you must use an array of **`long`**s to compute the Pell numbers, up to and including number n, then return Pell number n.

Question 8 (25 points)

In this question, you are going to program a class `TetrisCompetition` that collects the results of a Tetris competition. In the competition, players try to get as many points as they can playing Tetris. Of all their attempts, we use the **median** score to determine who wins the competition.

The median score is any of the values such that at most half of the population is less than the proposed median and at most half is greater than the proposed median. For example, the median of $\{1,3,3,6,7,8,9\}$ is 6. When the number of values is *even*, the median is the average of the middle two numbers. For example, the median of $\{1,2,3,4,5,6,8,9\}$ is 4.5.

a. *(4 pts)* Create and implement a Java class `TetrisCompetition` with a method `int median(List<Integer> numbers)` that computes the median number of a given list of numbers. Hint: you can use Java's `Collections.sort` method.

b. *(4 pts)* A `TetrisCompetition` object has to keep track of all obtained scores in the competition. Add an appropriate field or appropriate fields to your `TetrisCompetition` class and implement a method `addResult(String name, int score)` which adds one score to the scores recorded by the `TetrisCompetition` object.

c. *(4 pts)* Implement a method `String getWinner()` which returns the winner of the competition.

d. *(2 pts)* Explain your design decisions regarding how you store the participants and their scores.

e. *(2 pts)* Explain your design decisions regarding how you compute and return the winner of the competition.

f. *(2 pts)* Explain your design decisions of using **`public`**, **`protected`** and **`private`** for the fields and methods of your class.

g. *(2 pt)* Explain your design decisions regarding how you use initializers and/or constructors to initialize objects of your class.

h. *(5 pts)* Define appropriate preconditions and postconditions for `addResult`. Pay attention that your postcondition should describe the change to the state of the `TetrisCompetition` object after a call to `addResult`.

Question 9 (10 points)

The results from the competition of the previous question are stored in a file, which is a sequence of names and scores, separated by whitespace (one or more spaces, tabs, newlines), for example:

```
linda 1523   jane 145  carl
41 thomas 331 thomas 366 john 321  annet 142 linda 1034
thomas 399 carl 99 thomas 901

anon 911 annet 420 linda 9999
```

a. *(5 pts)* Implement a static method `TetrisCompetition processFile(String filename)` which reads a file (given by the filename) with competition results, and adds the results to a new `TetrisCompetition` object which is returned by the method. You may assume that a given file follows this file format. Any exceptions should be thrown (not caught in `processFile`).

b. *(5 pts)* Implement a `main` method that expects a command line argument with a filename of a file with competition results. Read the file, then print the winner of the competition to standard output. Add appropriate error messages for cases such as files not existing, command line parameters are missing, etc.

Question 10 (10 points)

Consider the following Java class.

```
1  public class ProducerConsumerExample {
2      public static class StackOfNumbers {
3          private final StackOfNumbers rest;
4          private final int number;
5
6          StackOfNumbers(StackOfNumbers rest, int number) {
7              this.rest = rest;
8              this.number = number;
9          }
10
11         public StackOfNumbers getRest() {
12             return rest;
13         }
14
15         public int getNumber() {
16             return number;
17         }
18     }
19
20     private static volatile StackOfNumbers top = null;
21
22     public static class Producer implements Runnable {
23         private int from, to;
24
25         public Producer(int from, int to) {
26             this.from = from;
27             this.to = to;
28         }
29
30         @Override
31         public void run() {
32             for (int i = from; i < to; i++) {
33                 top = new StackOfNumbers(top, i);
34             }
35         }
36     }
37
38     public static class Consumer implements Runnable {
39         private int count;
40
41         public Consumer(int count) {
42             this.count = count;
43         }
44
45         @Override
46         public void run() {
47             for (int i = 0; i < count; i++) {
48                 while (top == null) continue; // wait until we have something
49                 System.out.println("next␣item:␣" + top.getNumber());
50                 top = top.getRest();
51             }
52         }
53     }
54
```

```
55      public static void main(String[] args) {
56          Thread prod1 = new Thread(new Producer(0, 10));
57          Thread prod2 = new Thread(new Producer(10, 20));
58          Thread prod3 = new Thread(new Producer(20, 30));
59          Thread cons = new Thread(new Consumer(30));
60
61          prod1.start();
62          prod2.start();
63          prod3.start();
64          cons.start();
65          try {
66              prod1.join();
67              prod2.join();
68              prod3.join();
69              cons.join();
70          } catch (InterruptedException ignored) {
71          }
72      }
73  }
```

The **volatile** keyword (see the book by Eck, 12.1.4) indicates to Java that a variable or field might be changed by other threads and forbids Java from storing a "local copy". Without this keyword, the **while** loop of line 48 will loop forever, even if another thread changes the value of `top`.

a. *(3 pts.)* Someone runs the above `main` method and sees the numbers 0 to 29 in a seemingly random order. However the numbers are not completely random: there are constraints on the order that numbers appear. Describe these constraints.

b. *(3 pts.)* The next time they run the `main` method, they find that the program does not terminate. Explain how this is possible.

c. *(4 pts.)* Make the code thread-safe, ensuring that it always terminates and does not crash.

Question 11 (3 points)

Consider the following headline: "Australian tech unicorn Canva suffers security breach - Hacker claims to have stolen the data of 139 million Canva users."

a. Which of the three important security properties was violated here?

b. What are the other two security properties?

Question 12 (2 points)

What is multi-factor authentication and what advantage does it have? Give an example of multi-factor authentication.

Question 13 (5 points)

Next to a 3-digit (0-9) combination lock you find a piece of paper containing the following text:

```
5d8f6cce532a7aeb57196be62344095936793400b3aeb3580d248b17d5518a86: 338
fc71f2d6d38dbfc752ecaf2262916dc8ad99a34243d47b34691f9f8a3afaeffd:
```

You suspect that these are hexadecimal representations of the output of the application of the SHA-256 hash to a string representing the code for the lock. That is, the SHA-256 hash of the string `"338"` is `"5d8f6c.."`. You try 338, but the lock does not open. That was probably the old combination. Write a small program that tries to find which 3-digit combination matches the digest "fc71f2...". What is the current combination of the lock?

To get you started, we provide you with the following two snippets of code:

```
public static String hex(byte[] bytes) {
    StringBuilder result = new StringBuilder();
    for (byte aByte : bytes) {
         result.append(String.format("%02x", aByte));
    }
    return result.toString();
}

MessageDigest md = MessageDigest.getInstance("SHA-256");
byte[] hash = md.digest(text.getBytes(StandardCharsets.UTF_8));
```

In your answer, give both the current combination of the lock as well as the source code of your solution.