Sample tasks of the exam paper

Task 1
Assume that all required classes are imported, the program has the main method which works correctly. Draw the output of the program. Add dimensions in numbers and the colors in words next to the figures.

```java
public void start(Stage st) {
    GridPane p1 = new GridPane();

    String s = "emma";
    StringBuilder sb = new StringBuilder(s);
    if (s.equals(sb.reverse().toString())) {
        BorderPane p2 = new BorderPane();
        Button n1 = new Button("N1");
        Button n2 = new Button("N2");
        p2.setLeft(n1);
        p2.setCenter(n2);
        p1.add(p2, 1, 1);
    }

    String s2 = "tartu";
    StringBuilder sb2 = new StringBuilder(s2);
    if (sb2.equals(sb2.reverse())) {
        Circle k1 = new Circle(300, 30, 30, Color.RED);
        p1.add(k1, 1, 2);
    }

    String s3 = "OOP";
    StringBuilder sb3 = new StringBuilder(s3);
    if (sb3.toString().equals(sb3.reverse().toString())) {
        Rectangle k2 = new Rectangle(50, 50, 100, 100);
        p1.add(k2, 2, 2);
    }

    Scene s1 = new Scene(p1, 400, 300);
    st.setTitle("Exam");
    st.setScene(s1);
    st.show();
}
```
Info about StringBuilder class

<table>
<thead>
<tr>
<th>StringBuilder</th>
<th>reverse()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Causes this character sequence to be replaced by the reverse of the sequence.</td>
</tr>
</tbody>
</table>

In the StringBuilder class, the equals method is not overridden.

Info about Circle class

<table>
<thead>
<tr>
<th>Constructor and Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Circle(double centerX, double centerY, double radius, Paint fill)</td>
</tr>
<tr>
<td>Creates a new instance of Circle with a specified position, radius and fill.</td>
</tr>
</tbody>
</table>

Info about Rectangle class

<table>
<thead>
<tr>
<th>Constructor and Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rectangle(double width, double height, Paint fill)</td>
</tr>
<tr>
<td>Creates a new instance of Rectangle with the given size and fill.</td>
</tr>
</tbody>
</table>

Info about GridPane class

GridPane lays out its children within a flexible grid of rows and columns.

add(Node child, int columnIndex, int rowIndex)
Task 2

```java
import java.util.*;
public class Structures {
    public static void main(String[] args) {
        HashSet<Integer> mySet = new HashSet<Integer>();
        mySet.add(3);
        mySet.add(1);
        mySet.add(1);
        ArrayList<Integer> myList = new ArrayList<Integer>();
        myList.add(1);
        myList.add(2);
        myList.add(1, 1);
        System.out.println(myList);
        System.out.println(mySet);
        System.out.println(mySet.addAll(myList));
        System.out.println(mySet.addAll(myList));
        System.out.println(mySet);
    }
}
```

What will the program output? Why?

What will change if line
`ArrayList<Integer> myList = new ArrayList<Integer>();`
is replaced by
`List<Integer> myList = new ArrayList<Integer>();` ?

What will change if line
`ArrayList<Integer> myList = new ArrayList<Integer>();`
is replaced by
`List<Integer> myList = new List<Integer>();` ?
## addAll

```java
boolean addAll(Collection<? extends E> c)
```

Adds all of the elements in the specified collection to this set if they're not already present (optional operation). If the specified collection is also a set, the `addAll` operation effectively modifies this set so that its value is the union of the two sets. The behavior of this operation is undefined if the specified collection is modified while the operation is in progress.

**Specified by:**

```
addAll in interface Collection<E>
```

**Parameters:**
- `c` - collection containing elements to be added to this set

**Returns:**
- `true` if this set changed as a result of the call
public class TestException {

    public static void main(String[] args) {

        int[] a = {1, 0};
        try {
            System.out.println(1/a[1]);
            System.out.println("Between");
            System.out.println(a[3]);
        } catch (ArithmeticException e) {
            System.out.println("First catch");
        } catch (RuntimeException e) {
            System.out.println("Second catch");
        } finally {
            System.out.println("Finally");
            System.out.println(a[3]);
        }

        System.out.println("After");
    }
}

What is the output of the program? Explain your answer.

Info about ArithmeticException class

ArithmeticException class is a subclass of RuntimeException class.
ArrayIndexOutOfBoundsException class is (indirect) subclass of RuntimeException class.
Task 4

class D {
    D() {
        System.out.print("d");
    }
    D(boolean large) {
        if (large) {
            System.out.print("D");
        }
    }
}

class E extends D {
    E() {
        System.out.print("e");
    }
    E(boolean large) {
        super(large);
        if (large) {
            System.out.print("E");
        }
    }
}

class F extends E {
    F(boolean large) {
        if (large) {
            System.out.print("F");
        }
    }
}

Task 4.1

What is the output of the program if an instance of the F class is created using the following statement:
new F(true);? Explain your answer.

Task 4.2

What is the output of the program if an instance of the E class is created using the following statement:
new E(true);? Explain your answer.

Task 4.3

What is the output of the program if an instance of the D class is created using the following statement:
new F(false);? Explain your answer.
Task 5

The Random class has the nextBoolean method to generate a random value of the boolean type. The following class is created to calculate the percentage of the generated boolean values.

```java
import java.util.Random;
public class BooleanPercent extends Random {
    double bolProcent;
    BooleanPercent(int percent) {
        bolProcent = percent / 100.0;
    }

    public boolean nextBoolean() {
        if (nextDouble() > bolProcent)
            return true;
        return false;
    }
}
```

In the client class, random boolean values have to be generated and the percent calculated.

```java
BooleanPercent bool1 = new BooleanPercent(40);
System.out.println(bool1.nextBoolean());
Random bool2 = new BooleanPercent(40);
System.out.println(bool2.nextBoolean());
Random bool3 = new Random(40);
System.out.println(bool3.nextBoolean());
BooleanPercent bool4 = new Random(40);
System.out.println(bool4.nextBoolean());
```

Will the program compile? If yes - what is the output; if not – why?
Java API. Random

If two instances of Random are created with the same seed, and the same sequence of method calls is made for each, they will generate and return identical sequences of numbers.

**Random()**
- Creates a new random number generator.

**Random(long seed)**
- Creates a new random number generator using a single long seed.

**nextBoolean()**
- Returns the next pseudorandom, uniformly distributed boolean value from this random number generator’s sequence.

**nextDouble()**
- Returns the next pseudorandom, uniformly distributed double value between 0.0 and 1.0 from this random number generator’s sequence.
Task 6

There are four files below. Fill in the gaps so that the rest of the code does not have to be changed. The gap can be filled with several words or none. If there are dependencies between the gaps, add a comment “if the first gap is ..., then in the third gap there has to be ...”.

BlockOfFlats.java:
```java
public class BlockOfFlats _______Building _______InUse _______Measuring {
    //1., 2., 3. gap
    private static int maxHeight;
    private int numberOfFlats;
}
```

Building.java:
```java
public abstract class Building {
    public _______String buildingOwner(); //4. gap
    public _______String toString() {
        //5. gap
        return "Building - owner: " + buildingOwner();
    }
}
```

InUse.java:
```java
public interface InUse {
    public int numberOfPeople();
}
```

Measuring.java:
```java
public _______Measuring { //6. gap
    public int height();
}
```
Fill in the gaps. In sentences 4. and 5. underline the correct word.

1. **Class** BlockOfFlats is a _______________________ of Building **class**.

2. **Class** BlockOfFlats is a _______________________ of InUse **class**.

3. **Class** BlockOfFlats _______________________ the Measuring ________________.

4. **Variable** maxHeight is a(n) (instance field/class field) of BlockOfFlats **class**.

5. **Variable** numberOfFlats is a(n) (instance field/class field) of the **BlockOfFlats class**.

6. To make the program compile, the **BlockOfFlats class** has to have at least the following methods: ______________________________________________________________.
Task 7

The arguments of the main method are 1 and 2. It is expected that the program outputs the value of 0.5. Unfortunately, the program does not meet the requirement. There are several possible reasons listed below. Add explanations to each of the statements below and note if it is the case.

1. A public method cannot invoke a private method.

2. Using the arguments 1 and 2, the f1 method will not return the value of 0.5.

3. There has to be throws ArithmeticException in the main method header.

4. An exception of type ArrayIndexOutOfBoundsException occurs.

5. An exception of type ArithmeticException occurs.

6. The output of the program is written into the file, not to the screen.

What happens if the program arguments are 1 and 0?

What happens if the program arguments are not given?
public class Arithmetics {

    public static void main(String[] args) throws FileNotFoundException {

        OutputStream output = new FileOutputStream("systemout.txt");
        PrintStream printOut = new PrintStream(output);
        System.setOut(printOut);

        System.out.println(f1(Integer.parseInt(args[0]), Integer.parseInt(args[1])));
    }

    private static double f1(int a, int b) throws ArithmeticException {

        return a / b;
    }
}

Info about the ArithmeticException class

public class ArithmeticException
    extends RuntimeException

Thrown when an exceptional arithmetic condition has occurred. For example, an integer "divide by zero" throws an instance of this class. ArithmeticException objects may be constructed by the virtual machine as if suppression were disabled and/or the stack trace was not writable.
Task 8

There are three files below. Assume that all classes are imported.

```java
public class A implements Comparable<A> {
    private int a;

    public A(int a) {
        this.a = a;
    }

    public int compareTo(A a) {
        return this.a - a.a;
    }

    public String toString() {
        return Integer.toString(a);
    }
}

public class B implements Comparable<B> {
    private int b;

    public B(int b) {
        this.b = b;
    }

    public int compareTo(B b) {
        return b.b - this.b;
    }

    public String toString() {
        return Integer.toString(b);
    }
}

public class TestAB {
    public static void main(String[] args) {
        Queue<A> a = new LinkedList<>();
        a.add(new A(3));
        a.add(new A(5));
        a.add(new A(2));
        printTheResult(a);

        Queue<B> b = new PriorityQueue<>();
        b.add(new B(3));
        b.add(new B(5));
        b.add(new B(2));
        printTheResult(b);

        Queue<A> a2 = new PriorityQueue<A>(a);
        printTheResult(a2);
    }

    public static <T> void printTheResult(Queue<T> c) {
        while (!c.isEmpty()) {
            System.out.println(c.poll());
        }
    }
}
```
What is the output of the program? Why?

What happens if the following line of the program is commented out: `printTheResult(a)`?

What happens if variable `a2` is instantiated as follows: `PriorityQueue<A> a2 = new PriorityQueue<A>(a);`?

What happens if `poll()` method is replaced with `peek()` method?

**Info about the constructor of PriorityQueue<E> class**

```java
public PriorityQueue(Collection<? extends E> c)
```

Creates a PriorityQueue containing the elements in the specified collection. If the specified collection is an instance of a SortedSet or is another PriorityQueue, this priority queue will be ordered according to the same ordering. Otherwise, this priority queue will be ordered according to the natural ordering of its elements.

**Parameters:**
- `c` - the collection whose elements are to be placed into this priority queue

**Throws:**
- ClassCastException - if elements of the specified collection cannot be compared to one another according to the priority queue's ordering
- NullPointerException - if the specified collection or any of its elements are null

**Info about the methods of Queue<E> class**

```java
E peek()
Retrieves, but does not remove, the head of this queue, or returns null if this queue is empty.

E poll()
Retrieves and removes the head of this queue, or returns null if this queue is empty.
```
Task 9
Students are standing in the queue. At some point, a few people start thinking that the second queue is shorter, and they go to the second queue. The following classes simulate the students and the queues.

```java
public class Person {
    String pName;
    Person previous;
    public Person(String pName) {
      this.pName = pName;
    }
    public String toString() {
      return pName;
    }
}

public class ShopQueue {
    private Person first;
    private Person last;
    private int nr = 1;
    ShopQueue() {}       // ShopQueue(ShopQueue k) {
          // first = k.first;
          // last = k.last;
          // nr = k.nr;
    }
    void addStudent(Person i) {
        if (first == null) {
            first = i;
            last = first;
            System.out.println("The first in the queue: " + i);
        } else {
            if (nr % 3 == 0)
                System.out.println("To the next queue: " + i);
            else {
                i.previous = last;
                last = i;
                System.out.println("Next came: " + i);
            }
        }
        nr++;
    }
    public String toString() {
        StringBuilder myOutput = new StringBuilder("ShopQueue: ");
        Person i = last;
        while (i != null) {
            myOutput = myOutput.append(i + " -> ");
            i = i.previous;
        }
        return myOutput.substring(0, myOutput.length()-3);
    }
    Person myMethod(Person i) {
        Person i1 = last;
        i.previous = last.previous. previous. previous; last. previous. previous. previous = i;
        last = last. previous;
        return i1;
    }
}
```
The client class:

```java
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

public class TestQueue {
    public static void main(String[] args) {
        Person i1 = new Person("Mark");
        Person i2 = new Person("Kate");
        Person i3 = new Person("Dan");
        Person i4 = new Person("Sven");

        List<Person> students = new ArrayList<>(Arrays.asList(i1, i2, i3, i4));
        ShopQueue k = new ShopQueue();

        for (Person i : students)
            k.addStudent(i);

        System.out.println(k);
        System.out.println("==========");
        ShopQueue k2 = new ShopQueue(k);
        k2.addStudent(new Person("Mart"));
        k2.addStudent(new Person("Andres"));
        System.out.println(k2);
        System.out.println("==========");
        System.out.println(k.myMethod(new Person("Lisa")));
        System.out.println(k);
    }
}
```

What is the output of the program? Explain your answer.