Programmiermethodik, Termin 1 SS21 (online) – Teil 2

Prüfung war geteilt in 2 Teile. Teil 1 mit vielen kurzen Fragen und Teil 2 mit wenigen längeren Fragen.

Zeit: 60min

Gleiche Punktewertung wie in Teil 1.

Sorry für Bilder mit Toaster-Qualität :/

Frage 1 (Lückentext)

4 Punkte

Gegeben seien die folgenden Klassen.

```java
public class Top {
    public int variable = 1;

    public void method() {
        System.out.print("Top-method()");
    }

    public void method(double d) {
        System.out.print("Top-method(double)");
    }
}
```

```java
public class Middle extends Top {
    public void method() {
        System.out.print("Middle-method()");
    }

    public void method(int d) {
        System.out.print("Middle-method(int)");
    }
}
```

```java
public class Bottom extends Middle {
    public int variable = 3;

    public void method(double i) {
        System.out.print("Bottom-method(double)");
    }
}
```

```java
public class Main {
    public static void main(String[] args) {
        Top o1 = new Middle();
        Top o2 = new Bottom();
        int i1 = o1.variable;
        int i2 = o2.variable;
    }
}
```

Füllen Sie den Lückentext aus! Die im Lückentext verwendeten Variablen beziehen sich auf die Variablen der main-Methode der Klasse Main.

Durch den Aufruf von o1.method(); wird der Text _____ ausgegeben.
Durch den Aufruf von o2.method(); wird der Text _____ ausgegeben.

Durch den Aufruf von o1.method(2); wird der Text _____ ausgegeben.
Durch den Aufruf von o2.method(2); wird der Text _____ ausgegeben.

Durch den Aufruf von o1.method(5.5); wird der Text _____ ausgegeben.
Durch den Aufruf von o2.method(5.5); wird der Text _____ ausgegeben.

Die Variable i1 weist den Wert _____ auf.
Die Variable i2 weist den Wert _____ auf.

Frage 2 (wahr/falsch)

2,5 Punkte

Gegeben sei die folgende Klasse Rectangle.

```java
import java.util.Objects;

public final class Rectangle {
    private final int length;
    private final int width;

    public Rectangle(int length, int width) {
        this.length = length;
        this.width = width;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof Rectangle)) return false;
        Rectangle r = (Rectangle) o;
        return length == r.length && width == r.width;
    }

    @Override
    public int hashCode() {
        return Objects.hash(length, width);
    }

    public static void main(String[] args) {
        var rectangle = new Rectangle(5, 4);
        System.out.println(rectangle.equals(null));
    }
}

```

- Die equals Implementierung der Klasse Rectangle ist nicht transitiv.
- Die equals Implementierung der Klasse Rectangle ist symmetrisch.
- Die equals Implementierung der Klasse Rectangle ist nicht reflexiv.
- Die Klasse Rectangle kann ohne Fehler kompiliert werden.
- Bei der Ausführung der main-Methode der Klasse Rectangle kommt es zu einem Laufzeitfehler.

Frage 3 (multiple choice)

2 Punkte

Gegeben sei die folgende Klasse MyClass.

```
1  public class MyClass {
2      protected int x;
3      private double y;
4
5      protected final void methodA() {}
6      private int methodB() { return 1; }
7      public double methodC() { return 0.0; }
8      private static void methodD() {}
9      public static float methodE() { return 0.0f; }
10 }
```

Kreuzen Sie alle Methoden an, welche in eine Subklasse von MyClass vererbt werden! Nehmen Sie an, dass keine der angeführten Methoden in der Subklasse überschrieben wurden.

- methodA()
- methodB()
- methodC()
- methodD()
- methodE()

Frage 4 (wahr/falsch)

3,6 Punkte

Gegeben sei die Klasse OuterClass.

```
1  public class OuterClass {
2      private static int staticVariable = 1;
3      private int variable = 2;
4      private String s;
5
6      public void foo() {}
7
8      public static class InnerClass {
9          public void foo() {
10             System.out.println(staticVariable);
11             System.out.println(variable);
12             OuterClass outer = new OuterClass();
13             System.out.println(outer.s.toUpperCase());
14         }
15     }
16
17     public static void main(String[] args) {
18         OuterClass.InnerClass inner = new OuterClass.InnerClass();
19         inner.foo();
20     }
21 }
```

Kreuzen Sie jeweils an ob die angegebene Aussage wahr oder falsch ist! Nehmen Sie an, dass Zeilen, welche einen Kompilierfehler produzieren, auskommentiert sind, wenn nach Laufzeitfehlern gefragt wird.

- Die Methode foo, aus Zeile 9 der InnerClass, ist eine überladene Methode.
- Die Anweisung System.out.println(variable); in Zeile 11 führt zu einem Kompilierfehler.
- Die Anweisung System.out.println(outer.s.toUpperCase()); in Zeile 13 führt zu einem Kompilierfehler.
- Die Anweisung System.out.println(outer.s.toUpperCase()); in Zeile 13 führt zu einem Laufzeitfehler.
- Die Anweisung OuterClass.InnerClass inner = new OuterClass.InnerClass(); in Zeile 18 führt zu einem Kompilierfehler.
- Die Anweisung OuterClass.InnerClass inner = new OuterClass.InnerClass(); in Zeile 18 führt zu einem Laufzeitfehler.

Frage 5 (wahr/falsch)

2,5 Punkte

Gegeben sei die folgende Klasse Filter.

```java
public class Filter {
    private final List<Integer> values;

    public Filter(List<Integer> values) {
        if (values == null) {
            this.values = new ArrayList<>();
        } else {
            this.values = new ArrayList<>(values);
        }
    }

    public List<Integer> getValues() {
        return values;
    }

    public void filter1() {
        this.values.removeIf(value -> value < 0);
    }

    public void filter2() {
        for (Integer value : this.values) {
            if (value < 0) {
                this.values.remove(value);
            }
        }
    }

    public static void main(String[] args) {
        List<Integer> list = Stream.iterate(10, i -> i - 2)
                .limit(3)
                .collect(Collectors.toList());
        Filter f1 = new Filter(list);
        f1.filter1();

        Filter f2 = new Filter(list);
        f2.filter2();
    }
}
```

Kreuzen Sie jeweils an ob die angegebene Aussage wahr oder falsch ist!

- Durch die Methode getValues wird die Datenkapselung nicht verletzt.
- Die Methode filter1 verhält sich exakt gleich wie die Methode filter2.
- In der Methode filter2 kommt es bei der Ausführung der main-Methode zu einem Laufzeitfehler.
- Die Klasse Filter kann erfolgreich kompilieren.
- In der Methode filter2 kann es, abhängig vom Inhalt der Liste values, zu einem Laufzeitfehler kommen.

Frage 6

6,4 Punkte

Gegeben seien die folgenden Klassen ClassA und ClassB. Betrachtet werden jeweils die Deklarationen in ClassB.

```java
public class ClassA {
    private int method1() { return 1; }
    Number method2() { return 2.0; }
    public void method3() {}
    void method4(int p1) {}
    public void method5(double p1, long p2) {}
    protected void method6(String p1) {}
    public final void method7() {}
    public void method8(Number p1) {}
}
```

```java
public class ClassB extends ClassA {
    int method1() { return 1; }
    Double method2() { return 2.0; }
    protected void method3() {}
    void method4(int p1) throws IllegalArgumentException {}
    public final void method5(double p1, long p2) {}
    public void method6(String p1) {}
    public void method7() {}
    protected void method8(Double p1) {}
}
```

Kreuzen Sie jeweils an, ob es aufgrund dieser Definition zu einem **Kompilierfehler** kommt. Falls dies nicht der Fall ist, kreuzen Sie an, ob eine **überschriebene** oder **überladene** Methode vorliegt oder ob **keines dieser Kriterien** erfüllt wird.

- method1()
- method2()
- method3()
- method4(int)
- method5(double, long)
- method6(String)
- method7()
- method8(Double)

Frage 7 (Lückentext)

2 Punkte

Gegeben sei folgende Anweisung.

```
var collection = IntStream.iterate(1, i -> i + 2)
                          .limit(50)
                          .filter(i -> 100 % i == 0)
                          .boxed()
                          .collect(Collectors.toSet());
```

Hinweis: Auszug aus der Javadoc der Methode boxed():
„Returns a Stream consisting oft he elements of this stream, each boxed to an Integer“

Füllen Sie den Lückentext aus!

Das größte Element im Stream vor dem Filtern mit filter(i -> 100 % i == 0) ist _____

Die Summe der Elemente in collection ist _____

Die Anzahl der Elemente in collection ist _____

Der Rawtype von collection ist _____

Frage 8

2,5 Punkte

Gegeben seien die folgenden Klassen.

```
public class Class1<T> {}

public class Class2<T> extends Class1<T> {}

public class Class3<T> extends Class2<T> {}

public class Class4<T> extends Class2<T> {}

public class A {}

public class B extends A {}
```

Kreuzen Sie jeweils an, ob es bei den Zuweisungen zu einem **Kompilierfehler** kommt. Falls dies nicht der Fall ist, kreuzen Sie an, ob es beim Ausführen zu einem **Laufzeitfehler** kommt. Sollten beide Fälle nicht eintreten, wird von einer **erfolgreichen Zuweisung** gesprochen

- Class1<A> o1 = new Class1<A>();
- Class1<A> o2 = new Class1<B>();
- Class1<A> o3 = new Class3<A>();
- Class2<B> o4 = new Class2<A>();
- Class3<A> o5 = new Class4<A>();

Frage 9

2,5 Punkte

Gegeben seien die folgenden Klassen.

```
public class Class1<T> {}

public class Class2<T> extends Class1<T> {}

public class Class3<T> extends Class2<T> {}

public class A {}

public class B extends A {}
```

Kreuzen Sie jeweils an, ob es bei den Zuweisungen zu einem **Kompilierfehler** kommt. Falls dies nicht der Fall ist, kreuzen Sie an, ob es beim Ausführen zu einem **Laufzeitfehler** kommt. Sollten beide Fälle nicht eintreten, wird von einer **erfolgreichen Zuweisung** gesprochen

- Class1<? super B> o1 = new Class1<A>();
- Class3<? super B> o2 = new Class2<A>();
- Class2<? extends B> o3 = new Class3<B>();
- Class3<? extends A> o4 = new Class3<B>();
- Class2<? extends B> o5 = new Class2<A>();