Programmieren in Java
-Eingangstest-

Nummer: _____

1. Studiengang: ◯ Informatik B.Sc. ◯ Informatik M.Sc. ◯ ESE B.Sc. ◯ ESE M.Sc. ◯ Sonstiges:

Fachsemester: _____

Bitte Fragen, die Sie nicht beantworten können unbedingt mit "Weiß ich nicht" beantworten. Bitte raten Sie nicht! Dieser Test dient nur zur Einschätzung, was Sie schon können und wird nicht in Ihre Bewertung einfließen.

1 Selbsteinschätzung

1. Wie gut können Sie programmieren:
   ◯ Gar nicht ◯ Anfänger ◯ Mittelmäßig ◯ Gut ◯ Experte
2. Wie lange programmieren Sie schon: _____ Jahre
3. Welche Programmiersprachen können Sie (Mehrfachnennungen möglich)?
   ◯ Gar keine ◯ Java/C# ◯ Haskell/Scheme/Lisp/OCaml/F# ◯ Python/Ruby/Php ◯ JavaScript ◯ C/C++ ◯ Andere
4. Wo haben Sie hauptsächlich Programmieren gelernt?
   ◯ Noch gar nicht ◯ Studium ◯ Schule ◯ Selbststudium ◯ Beruf ◯ Sonstiges
5. Was war Ihr bislang größtes Programm, das Sie geschrieben haben (ca. Anzahl Zeilen) und in welcher Programmiersprache?

   ---

6. Ihre bisherigen Noten (`nb` für "nicht bestanden", `nt` für "nicht teilgenommen")?
   - Einführung in die Informatik: _____
   - Informatik 2 (Algorithmen und Datenstrukturen): _____

2 Java Syntax und Semantik

In den nun folgenden Fragen geht es ausschließlich um *Java*!

1. Welches sind Schlüsselwörter in Java?

   √ **static** √ **public** ◯ method √ **import** ◯ function √ **return** ◯ friend
   √ **volatile** √ **implements** √ **try** ◯ Weiß ich nicht

2. Was ist der Unterschied zwischen `int` und `Integer`?

   ◯ Ein `Integer` kann größere Zahlen als ein `int` speichern

   √ **Ein `int` ist ein nativer Typ, ein `Integer` eine Klasse**

   ◯ `Integer` ist ein Spezialfall von `int` und erlaubt z.B. schnellere Schleifen

   ◯ Weiß ich nicht

3. Erzeugen Sie ein Array `myarray` das Zahlen (`int`) aufnimmt und füllen Sie es mit den Zahlen 1,2 und 3.

   ```
   int[] myarray = new int[] { 1, 2, 3 };
   ```

   oder

   ```
   int[] myarray = new int[3];
   for (int i = 0; i < 3; i++) {
           myarray[i] = i+1;
   }
   ```

4. Sie haben wie folgt ein Array erzeugt und möchten dies um ein Element verlängern:

   ```
   public void bar() {
     int[] myarray = new int[10];
   }
   ```

   ◯ Kompiliert nicht.

   ◯ myarray[] = 1;

   √ **Geht nicht, die Arraysgröße ist nach Erstellung nicht mehr veränderbar.**

   ◯ myarray += 1;

   ◯ myarray.add(1);

   ◯ Weiß ich nicht

5. Was wird ausgegeben?

   ```
   if (0) {
     System.out.println("Not equal");
   } else {
     System.out.println("Equal");
   }
   ```

   √ **Kompiliert nicht.**

   ◯ `Not equal`

   ◯ `Equal`

   ◯ Weiß ich nicht

6. Initialisierungsreihenfolge

```
class MyWonderfulClass {
  int x = 10;
  boolean flag = x > 20;

  MyWonderfulClass() {
    x = 999;
  }
}
```

Welchen Wert hat **new** MyWonderfulClass().flag?

- ◯ Kompiliert nicht.
- √ **false**
- ◯ true
- ◯ Weiß ich nicht

7. Was wird ausgegeben wenn die Methode foo(); aufgerufen wird?

```
class A {
  private static boolean b;
  void static foo() {
    if (b) {
      System.out.println("true");
    } else {
      System.out.println("false");
    }
  }
}
```

- ◯ Kompiliert nicht.
- ◯ `true`
- √ `false`
- ◯ Zufällig, da b nicht initialisiert wurde.
- ◯ Weiß ich nicht

8. Welche Variante(n) kompilieren **nicht**?

- √ **class A { public abstract void bar(); }**
- ◯ **class abstract** A { **public abstract void** bar(); }
- ◯ **class abstract** A { **public void** bar() {} }
- ◯ Alle obigen Varianten kompilieren
- ◯ Weiß ich nicht

9. Was passiert?

```
class A {
  private void foo() { /* ... */ }
  public static void bar() {
    foo();
  }
}
```

◯ Kompiliert und die Methode `foo` wird in `bar` aufgerufen.

√ **Kompiliert nicht da die Methode `foo` nicht aufgerufen werden kann, weil sie nicht `static` ist.**

◯ Kompiliert nicht, weil die Klasse auch das Attribut `static` haben muss wenn eine ihrer Methoden `static` ist.

◯ Weiß ich nicht

10. Betrachten Sie folgendes Codefragment aus einem compilierendem Programm.

```
1 ObjA a1 = new ObjA();
2 a1.x = 5;
3 ObjA a2 = a1;
4 a2.x = 6;
5 if (□) { System.out.prinln("true"); } else { System.out.println("false"); }
```

Was ist die Ausgabe, wenn folgende Ausdrücke für □ eingesetzt werden?

□	"true"	" false"	kommt darauf an	weiß nicht
a1.x == 6	√			
a1.x == 5		√		
a1 == a2	√			
a1.equals(a2)	√		(√)	

11. Was gilt?

```
import java.util.Date;
```

√ **`import` erlaubt abkürzende Schreibweisen. Ohne `import` muss man `java.util.Date` schreiben, mit `import` kann man einfach nur `Date` schreiben.**

◯ Import kopiert die entsprechende Klasse in die aktuelle Datei (vgl. C/C++ `include`) damit man sie einfacher verwenden kann.

◯ Weiß ich nicht

12. Es gibt zwei Methoden `String`s zu vergleichen. Markieren Sie die „Richtige“!

◯ mystring == "abc"

√ **mystring.equals("abc")**

◯ Weiß ich nicht

13. Was wird ausgegeben wenn **new** B().foo(1); aufgerufen wird?

```
class A { void foo(int x) { System.out.println("A.foo"); }}
class B extends A { void foo(double x) { } }
```

◯ `A.foo`

√ **Nichts**

◯ Weiß ich nicht

14. Vervollständigen Sie die Klasse `ForrestGump` sodass sie die folgende Schnittstelle implementiert!

```
interface Run {
  public void run();
}
```

```
public class ForrestGump implements Run {
        public void run() {
                System.out.println("running...");
        }
}
```

15. Mit welcher Deklaration erzwingen Sie die Implementierung der Methode `foo` in einer nicht-abstrakten Subklasse?

    ◯ `public void foo();`

    ◯ `static void foo();`

    ◯ `public native void foo();`

    √ `abstract public void foo();`

    ◯ `protected void foo();`

    ◯ Weiß ich nicht

16. Sie möchten, dass Subklassen in beliebigen Paketen Zugriff auf eine Superklasse haben. Welches ist das restriktivste Attribut welches dieses Ziel erfüllt?

    ◯ `public`

    √ `protected`

    ◯ `transient`

    ◯ `native`

    ◯ `private`

    ◯ Das ist nicht möglich

    ◯ Weiß ich nicht

3 Programmierkenntnisse

1. Was ist die geeignetste Datenstruktur um das folgende Problem zu lösen?
   Sie wollen einen Druckauftragsverteiler implementieren, der folgendes tut:

   - Wenn ein neuer Druckauftrag kommt, dann wird er gedruckt, wenn er der erste ist.
   - Werden gerade andere Druckaufträge gedruckt, dann muss er warten, bis alle früheren fertig sind.

   √ **Schlange (queue)** ◯ Stapel (stack) ◯ Liste (list) ◯ Baum (tree) ◯ Weiß ich nicht

2. Sie wollen ein Biologiesystem nachbauen. Dabei wollen Sie Pflanzen, Bäume, Tannenbäume, Blumen und Rosen modellieren. Was für eine Implementierungsstrategie verwenden Sie?

   ◯ Ich verwende eine Konstante für jeden Typ (z.B. Pflanze=1, Baum=2,...)

   ◯ Ich verwende eine Klasse `Bioelement` die ein Feld `Typ` hat, welches mit dem jeweiligen Type gefüllt wird, z.B. `Typ=''Pflanze''` oder `Typ=''Rose''`

   √ **Ich verwende `Baum extends Pflanze` etc.**

   ◯ Weiß ich nicht

3. Schreiben Sie eine Methode die 4 Integer Zahlen als Parameter nimmt und den Durchschnitt der Zahlen zurückgibt (Klasse ist nicht notwendig)!

```
double avg(int i1, int i2, int i3, int i4) {
        return ((double)i1 + i2 + i3 + i4)/4;
}
```

4. Dynamischer vs. statischer Typ

```
A a = new A();
B b = new B();
B x = new A();
```

Geben Sie den statischen und dynamischen Typ für a,b und x an!

Dyn. Typ	A	B
a	√	
b		√
x	√	

Stat. Typ	A	B
a	√	
b		√
x		√

5. Sie haben eine veränderbare Liste (`implements List<T>`) von Zahlen und wollen diese jeweils um eins erhöhen. Was tun Sie?

○ `map (+1) list;`

○
```
for (int element : list) {
    element += 1;
}
```

√
```
for (int i=0;i<list.length();i++) {
    list.set(i,list.get(i) + 1);
}
```

○ Weiß ich nicht

4 Objektorientierte Konzepte

1. Ordnen sie den folgenden Codefragmenten eine Erklärung zu, die Ihnen passend erscheint:

```
interface A { ... }
interface B { void bAction(); }
class C implements B {
  private A a;
  public C (A a) { this.a = a; }
  void bAction() { ... }
}
```

√ **Die Schnittstelle `A` soll in eine Schnittstelle `B` mittels der Hilfsklasse `C` übersetzt werden.**

○ Es soll verhindert werden, dass mehrere Instanzen von `C` erzeugt werden können.

○ Der interne Zustand der Klasse `C` soll durch die Schnittstellen `A` und `B` repräsentiert werden.

○ Funktionalität von `C` Objekten soll erweitert werden.

○ Weiß ich nicht

2. Sie haben folgendes Codefragment:

```
class Rectangle {
  int width, height;
}
class Table {
  Rectangle tablePlate;
}
```

Sie möchten dem Tisch eine Methode plateArea, welche die Oberfläche des Tisches berechnet, hinzufügen. Was machen Sie?

- ◯ Table bekommt eine Methode
  `int area() { return tablePlate.width * tablePlate.height; }`
- ◯ Rectangle bekommt eine Methode
  `int area() { return width * height; }`
- √ **Rectangle bekommt eine Methode**
  `int area() { return width * height; }`
  und Table bekommt eine Methode `int area() { return tablePlate.area(); }`
- ◯ Weiß ich nicht

3. Geben Sie eine Klasse Pair an, die Paare von Werten speichern kann. Die Klasse Pair soll generisch sein!

```
public class Pair<X, Y> {
        private final X x;
        private final Y y;

        public Pair(X x, Y y) {
                this.x = x;
                this.y = y;
        }

        public X fst() {
                return x;
        }

        public Y snd() {
                return y;
        }

}
```