1. Aufgabe

Lernziel

Java-Schnittstellen problemgerecht bei eigenen Programmen verwenden können.

Aufgabenstellung

Eine Klassenbibliothek wurde im Entwurf u.a. folgendermaßen definiert:

Ein GUI-Objekt kann beliebig viele der folgenden Eigenschaften besitzen: druckbar, selbstzeichnend, mehrfarbig.

a. Erstellen Sie die Java-Schnittstellen
   PrintableI mit der Operation print(),
   DrawableI mit der Operation draw(),
   ColorizedI mit den Operationen setColor(Color c) und getColor().
b. Erstellen Sie ein Java-Klasse MiniMenu, die nur zeichenbar ist und die entsprechende(n) Operation(en) syntaktisch korrekt leer implementiert.
c. Erstellen Sie ein Java-Klasse SuperButton, die alle obigen Eigenschaften besitzt und die entsprechende(n) Operation(en) syntaktisch korrekt leer implementiert.

2. Aufgabe

Definieren Sie das Interface *FahrzeugCockpit* mit folgenden Operationen:

- public void bewegeLenkrad();
- public void gibGas();
- public void bremse();

Realisieren Sie dazu ein Programmsystem, das folgendes Programmfragment umsetzt:

```
FahrzeugCockpit fc[] = {new VW(), new BMW(), new Audi()};
for(int i=0; i<fc.length; i++){
     fc[i].bewegeLenkrad();
     fc[i].gibGas();
     fc[i].bremse();}
```

Die Operationsaufrufe sollen folgende Meldungen erzeugen:

„der VW bzw. BMW bzw. Audi lenkt bzw. gibt Gas bzw. bremst“.

3. Aufgabe

Nehmen Sie die Klassen *OberKlasse* und *UnterKlasse* und setzen Sie die Methoden der Oberklasse auf *public*. Diskutieren und Erklären Sie sich die Probleme, die auftreten, wenn Sie die *UnterKlasse* ausführen wollen.

4. Aufgabe

Nehmen Sie die Klassen *OberKlasse* und *UnterKlasse* und setzen Sie diese in unterschiedliche Pakete. Diskutieren und Erklären Sie sich die Probleme, die auftreten, wenn Sie die *UnterKlasse* ausführen wollen.

5. Aufgabe

Es gilt folgendes

class A **extends** B **implements** C - sowie **class B** und die folgenden syntaktisch richtigen Anweisungen:

```
A x = new A();
B y = new B();
B z = new A();
C c = new A();
```

Geben Sie für jede der folgenden Anweisungen jeweils an, ob kein Fehler, ein Fehler bei der Übersetzung oder ein Fehler bei der Ausführung auftritt. Jede Anweisung ist hier isoliert zu betrachten.

Gehen Sie folgendermaßen vor: versuchen Sie die Fragen erst theoretisch zu beantworten und schaffen Sie sich dann unter Eclipse ein Testszenario, um die Richtigkeit Ihrer Antworten zu testen

Anweisung	Kein Fehler	Fehler bei Übersetzung	Fehler bei Ausführung
x = y;	☐	☐	☐
x = (A)y;	☐	☐	☐
x = (C)y;	☐	☐	☐
y = x;	☐	☐	☐
y = (B)x;	☐	☐	☐
y = (C)x;	☐	☐	☐
c = x;	☐	☐	☐
c = y;	☐	☐	☐
c = z;	☐	☐	☐
c = (C)y;	☐	☐	☐
x = c;	☐	☐	☐
x = (A)c	☐	☐	☐
y = (A)c;	☐	☐	☐
y = (B)c;	☐	☐	☐
c = (C)z;	☐	☐	☐

6. Aufgabe

Es sollen mehrere Klassen geschrieben werden, um Messwerte zu speichern und auszugeben. Entwickeln Sie die Klassen `Messwert`, `Messreihe` und `TemperaturMessreihe`. Die Klasse `Messwert` soll folgende Kriterien erfüllen:

- Eine Klassenvariable `anzahlMesswerte` vom Typ `int` soll die Anzahl der Messwerte festhalten.
- Die Klasse soll die Attribute `wert` vom Typ `double`, `messDatum` vom Typ `LocalDate` sowie `messwertID` vom Typ `int` enthalten.
- Die Klasse soll sich im Paket `messdaten` befinden.
- Es dürfen nur Klassen im selben Paket auf die Klasse `Messwert` zugreifen und sie verwenden.
- Der Konstruktor soll nur für Klassen im Paket `messdaten` aufrufbar sein. Der Konstruktor soll als Übergabeparameter `messwert` vom Typ `double` und `messDatum` vom Typ `LocalDate` erwarten.

Folgende Methoden sollen implementiert werden:

- `double getWert()`
- `LocalDate getMessDatum()`
- `int getMesswertID()`

Die Klasse Messreihe befindet sich ebenfalls im Paket messdaten, soll aber von Klassen in anderen Paketen verwendet werden können. Die Klasse erhält folgende Attribute und Methoden:

- `protected Messwert[] messwerte`
  Die Messwerte werden in diesem Array gespeichert.
- `public Messreihe (int messwertAnzahl)`
  Dem Konstruktor wird die Größe des Messwert-Arrays übergeben.
- `public void addMesswert (double messwert, LocalDate datum)`
  Fügt dem Array ein neues Messwert-/Datum-Paar hinzu.
- `public double getMesswert (LocalDate datum)`
  Ermittelt den Messwert, der zum übergebenen Datum gehört.
- `public void print()`
  Gibt alle gespeicherten Messwerte auf der Konsole aus.

Die Klasse TemperaturMessreihe wird von der Klasse Messreihe abgeleitet und befindet sich im Paket temperaturmessung. Sie soll folgende Attribute und Methoden erhalten:

- `private String temperaturEinheit`
  Gibt die verwendete Temperaturskala an, z. B. °C.
- `public TemperaturMessreihe (int messwertAnzahl, String temperaturEinheit)`
  Der Konstruktor soll die Anzahl der zu speichernden Messwerte und die zu verwendende Temperaturskala entgegennehmen.
- `public void print()`
  Die Methode soll die verwendete Temperaturskala (z. B. °C) und alle gespeicherten Messwerte auf der Konsole ausgeben.

- `public static double CelsiusToFahrenheit (double celsiusTemp)`
  Die Methode konvertiert eine Temperaturangabe von Celsius nach Fahrenheit.

Die entwickelten Klassen können mit folgender Testklasse, die sich im Default-Paket befindet, getestet werden.

```
// Datei: TestMesswerte.java

package aufgabe6;
import temperaturmessung.TemperaturMessreihe;
import java.time.LocalDate;
public class TestMesswerte
{
      public static void main (String[] args)
      {
         double fahrenheit;
         TemperaturMessreihe temperaturMessungen =
            new TemperaturMessreihe (5, "°C");
         LocalDate datum1 = LocalDate.of(2000,5,10);
         temperaturMessungen.addMesswert (25.3, datum1);

         LocalDate datum2 = LocalDate.of(2001,5,10);
         temperaturMessungen.addMesswert (23.0, datum2);

         LocalDate datum3 = LocalDate.of(2002,5,10);
         temperaturMessungen.addMesswert (18.4, datum3);

         LocalDate datum4 = LocalDate.of(2003,5,10);
         temperaturMessungen.addMesswert (26.9, datum4);

         LocalDate datum5 = LocalDate.of(2004,5,10);
         temperaturMessungen.addMesswert (28.0, datum5);

         fahrenheit = TemperaturMessreihe.CelsiusToFahrenheit (25.0);
         System.out.println("25.0 °C entsprechen " +
            fahrenheit + "° F.");
         System.out.println();
         temperaturMessungen.print();
      }
}
```

7. Aufgabe

Machen Sie eine Kopie der 2. Aufgabe. Ergänzen Sie das Interface `FahrzeugCockpit` um die (`default`-)Operation:

- public void hatEDrive()
- public void metaDaten()

Modifizieren Sie das Programmsystem folgendermaßen:

FahrzeugCockpit fc[] = {new VW(), new BMW(), new Audi(), new Teslar()};

Dabei soll nur die Klasse `Teslar` neu erstellt werden(im Paket aufgabe7emobil). Alle anderen Auto-Klassen bleiben unverändert. Die `for`-Schleife wird um die Aufrufe `fc[i].hatEDrive(); fc[i].metaDaten();` ergänzt.

- Der Aufruf `hatEDrive()` erzeugt beim Teslar die Meldung: `der Teslar hat EDrive`. Bei allen anderen Fahrzeugen: `der XXX hat keinen EDrive`.
- Der Aufruf `metaDaten()` gibt Klasse und Paket des jeweiligen Objektes durch Komma getrennt aus.

8. Aufgabe

Betrachten Sie folgendes Interface und Klassen:

```
public class Meldung {
	private String typ, inhalt;
	public Meldung(String typ, String inhalt){
		this.typ = typ;
		this.inhalt=inhalt;
	}
	public String getTyp() {
		return typ;
	}
	public String getInhalt() {
		return inhalt;
	}
}

public interface Melden {
	void ausgabe(Meldung m);
}

public class TestMeldung {
	Melden melden;
	public TestMeldung(){

		 melden = new Melden(){

			@Override
			public void ausgabe(Meldung m) {
				System.out.println(m.getTyp()+ ": " + m.getInhalt());
			}
		};
	}
	Melden getMelden(){return melden;}

	public static void main(String[] args) {
		TestMeldung tm = new TestMeldung();
		Meldung m = new Meldung("Status", "gerade gestartet");
		tm.getMelden().ausgabe(m);
	}
}
```

Testen Sie diese. Ersetzen Sie dann die anonyme Klassenkonstruktion durch einen Lambda-Ausdruck. Reimplementieren Sie dazu die Klasse `TestMeldung` in einem neuen Paket (z.B. aufgabe8Lambda).

9. Aufgabe

Testen und ersetzen Sie die anonymen Konstruktionen der folgenden zwei Klassen durch Lambda-Ausdrücke.

```
public class LambdaHallo {
        public static void main(String[] args) {
                Printer p = new Printer() {
                        @Override
                        public void print() {
                                System.out.println("Hallo Welt");
                        }
                };
                p.print();
        }
}
interface Printer { void print();}

public class LambdaCalculateTest {
   public static void main(String[] args) {
      Rectangle rect = new Rectangle() {
                        @Override
                        public double getArea(double l, double w) {
                                l*=2; w*=3;
                                return l * w;
                        }
                };
      System.out.println("Flaeche: " + rect.getArea(4, 3));
   }
}
interface Rectangle {
      public double getArea(double length, double width);
}
```

10. Aufgabe

Betrachten sie folgende (nicht ganz unkomplizierte) Konstruktion mit zwei anonymen Klassen.

```
public class ReferenzHallo {
	public static void main(String[] args) {
		berechneUndGibAus(new IFPar3() {
			@Override
			public String berechne(int p1, double p2, double p3) {
				return ToolKlasse.flaeche(p1, p2, p3);
			}
		}, new IFPar1() {
			@Override
			public void machWas(String s) {
				System.out.println(s);
			}

		}, 2, 3.5, 4.3);
	}

	static void berechneUndGibAus(IFPar3 p3, IFPar1 p1, int typ, double a, double b) {
			p1.machWas(p3.berechne(typ, a, b));
	}
}
interface IFPar3 {
	public String berechne(int p1, double p2, double p3);
}
interface IFPar1{
	public void machWas(String s);
}
class ToolKlasse{
	static String flaeche(int typ, double a, double b){
		if(typ == 1) return "Fläche Rechteck: " + a*b;
				else return "Fläche Ellipse: " + Math.PI*a*b;
 }
}
```

a) Ersetzen Sie in einem ersten Schritt diese anonymen Klassen durch Lambda-Ausdrücke
b) Ersetzen Sie in einem zweiten Schritt die Lambda-Ausdrücke durch Methodenreferenzen
c) Welche weiteren Veränderungen der durch b) erhalten Konstruktion sind notwendig wenn die Methode `flaeche` in `ToolKlasse` nicht mehr statisch ist
d) Ersetzen Sie in der nun nochmal modifizierten Ausgangsklasse unten, die erzeugten anonymen Klassen direkt durch Methodenreferenzen.

```
public class ReferenzHallo {
	public ReferenzHallo(){
		berechneUndGibAus(new IFPar3() {
		@Override
		public String berechne(int p1, double p2, double p3) {
			return flaeche(p1, p2, p3);
		}
	}, new IFPar1() {
		@Override
		public void machWas(String s) {
			System.out.println(s);
		}

	}, 2, 3.5, 4.3);
	}
	public static void main(String[] args) {
		new ReferenzHallo();
	}

	void berechneUndGibAus(IFPar3 p3, IFPar1 p1, int typ, double a, double b) {
		p1.machWas(p3.berechne(typ, a, b));
	}
	String flaeche(int typ, double a, double b) {
		if (typ == 1)
			return "Fläche Rechteck: " + a * b;
		else
			return "Fläche Ellipse: " + Math.PI * a * b;
	}
}
interface IFPar3 {
	public String berechne(int p1, double p2, double p3);
}
interface IFPar1 {
	public void machWas(String s);
}
```

11. Aufgabe

a) Weise den folgenden Lambda-Ausdrücken jeweils eine funktionale Schnittstelle zu. (Die Methoden stammen aus der Klasse String). Testen Sie Ausdrücke und Schnittstellen.
   - () -> System.out.println("lambda!");
   - () -> "lambda!"
   - s -> s.toLowerCase()
   - s -> s.contains("-")
   - (s,t) -> s.contains(t)
   - (s,t) -> s.concat(t)
   - (s,ch) -> s.indexOf(ch)
   - (s,ch,fromIndex) -> s.indexOf(ch,fromIndex)

b) Suche zu dem folgenden Lambda-Ausdruck eine passende funktionale Schnittstelle und schreibe den Lambda-Ausdruck als Methodenreferenz
   - s -> System.out.println(s)

Fragen zum Wiederholen/ Selbststudium/ Ausprobieren

Interfaces, abstrakte Klassen, Modifier

1. Frage

Welche der untenstehenden Begriffe können in Interfaces vor Methoden stehen?

a) protected
b) private
c) final
d) abstract
e) public
f) package local
g) static
h) default
i) Keine dieser Möglichkeiten.

2. Frage

Welche der untenstehenden Aussagen sind korrekt?

a) Ein Interface darf keine Konstanten enthalten.
b) Ein Interface darf Methoden mit Methodenkörper enthalten.
c) Methoden in Interfaces müssen leer sein.
d) Eine abstrakte Klasse darf Methoden mit Methodenkörper enthalten.
e) Es heißt Interface A „implements“ Interface B.
f) Methoden in Interfaces sind implizit private.
g) Enthält eine Klasse eine abstrakte Methode, muss die Klasse nicht abstrakt sein.
h) Methoden in Interfaces sind nie static.

3. Frage

Welche der untenstehenden Aussagen sind korrekt?

a) Ein Interface darf eine extends-Beziehung zu mehreren Interfaces haben.
b) Eine abstrakte Klasse darf Methoden mit Methodenkörper enthalten.
c) Methoden in abstrakten Klassen müssen leer sein.
d) Eine Klasse darf Methoden ohne Methodenkörper enthalten.
e) Es heißt interface A extends interface B.
f) Methoden in Interfaces sind implizit public.
g) Methoden in Interfaces sind implizit abstract.
h) Enthält eine Klasse eine abstrakte Methode, muss die Klasse abstrakt sein.
i) Es heißt class A extends interface B.

4. Frage

In welcher Zeile tritt ein Kompilierfehler auf?

```
interface A {
      public byte a = 1;                  Zeile 1
      public static final byte b = 1;     Zeile 2
      byte c = 1;                         Zeile 3
      long d = 5;                         Zeile 4
      static int e = 1;                   Zeile 5
}
```

a) Zeile 1
b) Zeile 2
c) Zeile 3
d) Zeile 4
e) Zeile 5
f) Keine dieser Möglichkeiten.

5. Frage

In welcher Zeile tritt ein Kompilierfehler auf?

```
interface A {
      protected short a = 2;        Zeile 1
      private byte b = 2;           Zeile 2
      short c = 2;                  Zeile 3
      long d = 2;                   Zeile 4
      int e = 2;                    Zeile 5
}
```

a) Zeile 1
b) Zeile 2
c) Zeile 3
d) Zeile 4
e) Zeile 5
f) Keine dieser Möglichkeiten.

6. Frage

In welcher Zeile tritt ein Kompilierfehler auf?

```
interface A {
      public short a = 25;          Zeile 1
      public final byte b = 25;     Zeile 2
      static int c = 25;            Zeile 3
      long d = 25;                  Zeile 4
      final int e = 21;             Zeile 5
}
```

a) Zeile 1
b) Zeile 2
c) Zeile 3
d) Zeile 4
e) Zeile 5
f) Keine dieser Möglichkeiten.

7. Frage

In welcher Zeile tritt ein Kompilierfehler auf?

```
interface A {
      public void a1 ( );     Zeile 1
      private void a2 ( );    Zeile 2
      protected void a3 ( );  Zeile 3
      abstract void a4 ( );   Zeile 4
      final void a5 ( );      Zeile 5
}
```

a) Zeile 1
b) Zeile 2
c) Zeile 3
d) Zeile 4
e) Zeile 5
f) Keine dieser Möglichkeiten.

8. Frage

In welcher Zeile tritt ein Kompilierfehler auf?

```
interface A {
      public void a1 ( );           Zeile 1
      void a2 ( );                  Zeile 2
      int a3 ( );                   Zeile 3
      abstract void a4 ( );         Zeile 4
      abstract public void a5();    Zeile 5
      static void a6( ){};          Zeile 6
}
```

a) Zeile 1
b) Zeile 2
c) Zeile 3
d) Zeile 4
e) Zeile 5
f) Zeile 6
g) Keine dieser Möglichkeiten.

9. Frage

In welcher Zeile tritt ein Kompilierfehler auf?

```
interface A {
      private void a1 ( );    Zeile 1
      void a2 ( );            Zeile 2
      int a3 ( );             Zeile 3
      protected void a4 ( );  Zeile 4
      public void a5 ( );     Zeile 5
      default void a6(){};    Zeile 6
}
```

a) Zeile 1
b) Zeile 2
c) Zeile 3
d) Zeile 4
e) Zeile 5
f) Zeile 6
g) Keine dieser Möglichkeiten.

10. Frage

Welche der untenstehenden Bemerkungen trifft nicht zu?

a) Eine abstrakte Klasse kann instanziiert werden.
b) Die Modifier abstrakt und final schließen sich gegenseitig aus.
c) Von einer final class darf es Subklassen geben.
d) Alle Methoden in einer final class sind implizit final.

11. Frage

Welche der untenstehenden Deklarationen für Klassen und Interfaces sind möglich?

a) abstract final class a { }
b) final public class a { }
c) private interface a { }
d) abstract class a { }
e) abstract public interface a { }
f) protected interface a { }
g) interface a{ }

12. Frage

Welche der untenstehenden Deklarationen für Methoden und Variablen sind nicht möglich?

a) abstract final void m ();
b) public void m();
c) abstract int a = 34;
d) private int a = 34;

Subtyping und Casting von Objekten

1. Frage

In welcher Zeile tritt ein Kompilierfehler auf?

```
class A { }

class B extends A{ }

class C extends B {
      public static void main(String[ ] args){
            A a = new A();    Zeile 1
            B b = new B();    Zeile 2
            C c = new C();    Zeile 3
            a = b;            Zeile 4
            b = (B) a;        Zeile 5
            b = c;            Zeile 6
      }
}
```

a) Zeile 1
b) Zeile 2
c) Zeile 3
d) Zeile 4
e) Zeile 5
f) Zeile 6
g) Keine dieser Möglichkeiten.

2. Frage

In welcher Zeile tritt ein Kompilierfehler auf?

```
class A {}

class B {}

class C {
      public static void main(String[ ] args){
            A a = new A();    Zeile 1
            B b = new B();    Zeile 2
            C c = new C();    Zeile 3
            a = b;            Zeile 4
            b = (B) a;        Zeile 5
            b = c;            Zeile 6
      }
}
```

a) Zeile 1
b) Zeile 2
c) Zeile 3
d) Zeile 4
e) Zeile 5
f) Zeile 6
g) Keine dieser Möglichkeiten.

3. Frage

In welcher Zeile tritt ein Kompilierfehler auf?

```
class A {}

class B extends A{}

class C extends B{
      public static void main(String[ ] args){
            A a = new A();    Zeile 1
            B b = new B();    Zeile 2
            C c = new C();    Zeile 3
            a = b;            Zeile 4
            b = a;            Zeile 5
            b = c;            Zeile 6
      }
}
```

a) Zeile 1
b) Zeile 2
c) Zeile 3
d) Zeile 4
e) Zeile 5
f) Zeile 6
g) Keine dieser Möglichkeiten.

4. Frage

In welcher Zeile tritt ein Kompilierfehler auf?

```
class A {}

class B extends A{}

class C extends A{
      public static void main(String[ ] args){
            A a = new A();    Zeile 1
            B b = new B();    Zeile 2
            C c = new C();    Zeile 3
            a = b;            Zeile 4
            a = c;            Zeile 5
            b = c;            Zeile 6
      }
}
```

a) Zeile 1
b) Zeile 2
c) Zeile 3
d) Zeile 4
e) Zeile 5
f) Zeile 6
g) Keine dieser Möglichkeiten.

5. Frage

In welcher Zeile tritt ein Kompilierfehler auf?

```
class A {}

class B extends A{}

class C extends A{
      public static void main(String[ ] args){
            A a = new A();    Zeile 1
            B b = new B();    Zeile 2
            C c = new C();    Zeile 3
            a = b;            Zeile 4
            b = a;            Zeile 5
            a = c;            Zeile 6
      }
}
```

a) Zeile 1
b) Zeile 2
c) Zeile 3
d) Zeile 4
e) Zeile 5
f) Zeile 6
g) Keine dieser Möglichkeiten.

6. Frage

In welcher Zeile tritt ein Kompilierfehler auf?

```
class A {}

class B extends A{}

class C extends A{
      public static void main(String[ ] args){
            A a = new A();    Zeile 1
            B b = new B();    Zeile 2
            C c = new C();    Zeile 3
            b = (B)a;         Zeile 4
            a = c;            Zeile 5
            b = a;            Zeile 6
      }
}
```

a) Zeile 1
b) Zeile 2
c) Zeile 3
d) Zeile 4
e) Zeile 5
f) Zeile 6
g) Keine dieser Möglichkeiten.

7. Frage

In welcher Zeile tritt ein Kompilierfehler auf?

```
class A {}

class B extends A{}

class C extends A{
      public static void main(String[ ] args){
            A a = new A();    Zeile 1
            B b = new B();    Zeile 2
            C c = new B();    Zeile 3
            a = b;            Zeile 4
            b = c;            Zeile 5
            b = a;            Zeile 6
      }
}
```

a) Zeile 1
b) Zeile 2
c) Zeile 3
d) Zeile 4
e) Zeile 5
f) Zeile 6
g) Keine dieser Möglichkeiten.

8. Frage

In welcher Zeile tritt welcher Fehler auf?

```
class A {}

class B extends A{}

class C extends A{
      public static void main(String[ ] args){
            A a = new A();
            B b = new B();
            C c = new C();
            a = b;            Zeile 1
            c = (C)a;         Zeile 2
      }
}
```

a) Es tritt in Zeile 1 ein Kompilierfehler auf.
b) Es tritt in Zeile 1 ein Laufzeitfehler auf.
c) Es tritt in Zeile 2 ein Kompilierfehler auf.
d) Es tritt in Zeile 2 ein Laufzeitfehler auf.
e) Keine dieser Möglichkeiten.

9. Frage

In welcher Zeile tritt welcher Fehler auf?

```
class A {}

class B extends A{}

class C extends A{
      public static void main(String[ ] args){
            A a = new A();
            B b = new B();
            C c = new C();
            a = b;            Zeile 1
            c = a;            Zeile 2
      }
}
```

a) Es tritt in Zeile 1 ein Kompilierfehler auf.
b) Es tritt in Zeile 1 ein Laufzeitfehler auf.
c) Es tritt in Zeile 2 ein Kompilierfehler auf.
d) Es tritt in Zeile 2 ein Laufzeitfehler auf.

10. Frage

Welches Ergebnis erhält man, wenn man folgendes Programm kompiliert und startet?

```
class A {}

class B extends A{}

class C extends A{
      public static void main(String[ ] args){
            A a = new A();
            B b = new B();
            C c = new C();
            boolean b1 = a instanceof B;  Zeile 1
            boolean b2 = a instanceof C;  Zeile 2
            boolean b3 = c instanceof A;  Zeile 3
            System.out.println(b1);
            System.out.println(b2);
            System.out.println(b3);
      }
}
```

a) true, true, false
b) true, false, true
c) false, false, true
d) false, true, false
e) false, true, true
f) Es tritt in Zeile 1 ein Kompilierfehler auf.
g) Es tritt in Zeile 2 ein Kompilierfehler auf.
h) Es tritt in Zeile 3 ein Kompilierfehler auf.

11. Frage

Welches Ergebnis erhält man, wenn man folgendes Programm kompiliert und startet?

```
class A {}
class B extends A{}
class C extends A{
      public static void main(String[ ] args){
            A a = new A();
            B b = new B();
            C c = new C();
            boolean b1 = a instanceof B; Zeile 1
            boolean b2 = a instanceof C; Zeile 2
            boolean b3 = c instanceof B; Zeile 3
            System.out.println(b1);
            System.out.println(b2);
            System.out.println(b3);
      }
}
```

a) true, true, false
b) true, false, true
c) false, true, false
e) false, true, trued, true, false, false
f) Es tritt in Zeile 1 ein Kompilierfehler auf.
g) Es tritt in Zeile 2 ein Kompilierfehler auf.
h) Es tritt in Zeile 3 ein Kompilierfehler auf.

12. Frage

Welches Ergebnis erhält man, wenn man folgendes Programm kompiliert und startet?

```
class A {}
class B extends A{}
class C extends A{
      public static void main(String[ ] args){
            A a = new A();
            B b = new B();
            C c = new C();
            boolean b1 = a instanceof B; Zeile 1
            boolean b2 = a instanceof C; Zeile 2
            boolean b3 = b instanceof A; Zeile 3
            System.out.println(b1);
            System.out.println(b2);
            System.out.println(b3);
      }
}
```

a) true, true, false
b) true, false, true
c) false, true, false
d) true, false, false
e) false, false, true
f) Es tritt in Zeile 1 ein Kompilierfehler auf.
g) Es tritt in Zeile 2 ein Kompilierfehler auf.
h) Es tritt in Zeile 3 ein Kompilierfehler auf.

13. Frage

Welches Ergebnis erhält man, wenn man folgendes Programm kompiliert und startet?

```
class A {}
class B extends A{}
class C extends A{
      public static void main(String[ ] args){
            A a = new A();
            B b = new B();
            C c = new C();
            boolean b1 = a instanceof B; Zeile 1
            boolean b2 = a instanceof C; Zeile 2
            boolean b3 = c instanceof Object; Zeile 3
            System.out.println(b1);
            System.out.println(b2);
            System.out.println(b3);
      }
}
```

a) true, true, false
b) true, false, true
c) false, false, true
d) false, true, false
e) true, false, false
f) Es tritt in Zeile 1 ein Kompilierfehler auf.
g) Es tritt in Zeile 2 ein Kompilierfehler auf.
h) Es tritt in Zeile 3 ein Kompilierfehler auf.

14. Frage

Welches Ergebnis erhält man, wenn man folgendes Programm kompiliert und startet?

```
class A {}
class B extends A{}
class C extends A{
      public static void main(String[ ] args){
            A a = new A();
            B b = new B();
            C c = new C();
            boolean b1 = a instanceof B; Zeile 1
            boolean b2 = c instanceof C; Zeile 2
            boolean b3 = b instanceof A; Zeile 3
            System.out.println(b1);
            System.out.println(b2);
            System.out.println(b3);
      }
}
```

a) true, true, false
b) true, false, true
c) false, false, true
d) false, true, false
e) false, true, true
f) Es tritt in Zeile 1 ein Kompilierfehler auf.
g) Es tritt in Zeile 2 ein Kompilierfehler auf.
h) Es tritt in Zeile 3 ein Kompilierfehler auf.

15. Frage

Welches Ergebnis erhält man, wenn man folgendes Programm kompiliert und startet?

```
class A {}
class Tisch {}
class B extends A{}
class C extends A{
      public static void main(String[ ] args){
            A a = new A();
            B b = new B();
            C c = new C();
            boolean b1 = a instanceof B;        Zeile 1
            boolean b2 = a instanceof Tisch;    Zeile 2
            boolean b3 = c instanceof A;        Zeile 3
            System.out.println(b1);
            System.out.println(b2);
            System.out.println(b3);
      }
}
```

a) true, true, false
b) true, false, true
c) false, false, true
d) true, false, false
e) false, true, true
f) Es tritt in Zeile 1 ein Kompilierfehler auf.
g) Es tritt in Zeile 2 ein Kompilierfehler auf.
h) Es tritt in Zeile 3 ein Kompilierfehler auf.