

Einleitung:

Aufgabe 1 (Dynamische Polymorphie)

Gegeben seien die folgenden drei Klassen:

```
public class Y1{
    public void test(){
        System.out.println("1\n");
    }
}

public class Y2 extends Y1{
    @Override
    public void test(){
        System.out.println("2\n");
    }

    public void do1(){
        System.out.println("y2:do1 ");
        test();
    }

    public void do2(){
        System.out.println("y2:do2 ");
        super.test();
    }

    public void do3(){
        System.out.println("y2:do3 ");
        do4();
    }

    public void do4(){
        System.out.println("3\n");
    }
}
```

```
public class Y3 extends Y2{
    @Override
    public void do1(){
        System.out.println("y3:do1 ");
        test();
    }

    @Override
    public void do2(){
        System.out.println("y3:do2 ");
        super.do3();
    }

    @Override
    public void do3(){
        System.out.println("y3:do3 ");
        super.do2();
    }

    public void do4(){
        System.out.println("3\n");
    }

    public void do4(String s){
        System.out.println(s + "y3:do4");
        do2();
    }
}
```

In einer weiteren Klasse Start.java werden in der Main-Methode folgende drei Objekte erzeugt:

```
Y1 x1 = new Y1();
Y2 x2 = new Y2();
Y3 x3 = new Y3();
```

Vererbung, Zugriffsrechte

Bearbeitung: KW 49
Testat: KW 51

Welche Ausgabe liefern die folgenden einzelnen Zeilen (falls sie keine Ausgabe angeben können, beschreiben Sie, was passiert)?

- a) `x1.test()`
- b) `x2.test()`
- c) `x2.do1()`
- d) `x2.do2()`
- e) `x2.do3()`
- f) `x2.do4()`
- g) `x3.test()`
- h) `x3.do1()`
- i) `x3.do2()`
- j) `x3.do3()`
- k) `x3.do4 („Programmierung“)`

Vererbung, Zugriffsrechte

Bearbeitung: KW 49
Testat: KW 51

Aufgabe 2 (dynamisches Binden, Sonderfälle)

Diese Aufgabe ist eine Abwandlung von Aufgabe 1. Änderungen sind fett markiert.

Gegeben seien die folgenden drei Klassen:

```
public class Y1{
    private void test(){
        System.out.println("1\n");
    }
}

public class Y2 extends Y1{
@Override
    protected void test(){
        System.out.println("2\n");
    }

    public void do1(){
        System.out.println("y2:do1 ");
        test();
    }

    void do2(){
        System.out.println("y2:do2 ");
super.test();
        do4();
    }

    public void do3(){
        System.out.println("y2:do3 ");
        do4();
    }

    public static void do4(){
        System.out.println("3\n");
    }
}
```

```
public class Y3 extends Y2{
    @Override
    public void do1(){
        System.out.println("y3:do1 ");
        test();
    }

    @Override
    public void do2(){
        System.out.println("y3:do2 ");
        super.do3();
    }

    @Override
    public void do3(){
        System.out.println("y3:do3 ");
        super.do3();
    }

    public static void do4(){
        System.out.println("3\n");
    }

    public void do4(String s){
        System.out.println(s + "y3:do4");
        do2();
    }
}
```

In einer weiteren Klasse Start.java werden in der Main-Methode folgende drei Objekte erzeugt:

```
Y1 x1 = new Y1();
Y2 x2 = new Y2();
Y3 x3 = new Y3();
Y1 x4 = new Y3();
Y2 x5 = new Y3();
```

Vererbung, Zugriffsrechte

Bearbeitung: KW 49
Testat: KW 51

Kommt es zu Fehlern? Warum? Welche Ausgabe liefern die folgenden einzelnen Zeilen (falls sie keine Ausgabe angeben können, beschreiben Sie, was passiert)?

- a) `x1.test()`
- b) `x2.test()`
- c) `x2.do1()`
- d) `x2.do2()`
- e) `x2.do3()`
- f) `x2.do4()`
- g) `x3.test()`
- h) `x3.do1()`
- i) `x3.do2()`
- j) `x3.do3()`
- k) `x3.do4 („Programmierung“)`
- l) `x4.do3();`
- m) `x5.do3();`

Vererbung, Zugriffsrechte

Bearbeitung: KW 49
Testat: KW 51

Aufgabe 3 (Zugriffsrechte)

Gegeben sei folgende Klasse:

```
//Verzeichnis: zugriff
//Datei: Zugriffsrechte.java
package zugriff;
public class Zugriffsrechte {
    float attribut1;
    private float attribut2;
    public float attribut3;
    protected float getAttribut2() {
        return this.attribut2;
    }
}
```

und folgendes *Klassen-Gerüst*:

```
//Verzeichnis: <verzeichnis>
//Datei: Zugriffstest.java
package <verzeichnis>;
import zugriff.Zugriffsrechte;

public class Zugriffstest extends <oberklasse> {
    Zugriffsrechte obj;
    public void testen() {

        this.attribut1 = 3.0F;
        this.attribut2 = 5.6F;
        this.attribut3 = obj.getAttribut2();

        obj.attribut1 = 3.0F;
        obj.attribut2 = 5.6F;
        obj.attribut3 = this.getAttribut2();
    }
}
```

Ersetzen Sie dabei wechselseitig

<verzeichnis>	durch (a) zugriff	(b) zugriffstest
<oberklasse>	durch (1) Zugriffsrechte	(2) Object

und überlegen Sie: Wo liefert der Compiler Fehlermeldungen und wieso?

Aufgabe 3 (Klassen und Methoden)

Implementieren Sie in Java eine Klasse `Konto`, die ein Bankkonto realisiert. Wählen Sie geeignete Zugriffsrechte für die Attribute und Methoden der Klasse. Ein Konto wird dabei repräsentiert durch einen Kontostand sowie einen eingeräumten Kreditrahmen. Die Klasse soll folgende Methoden zur Verfügung stellen:

- Einen Konstruktor zum Initialisieren eines neuen (leeren) Kontos.
- Einen Copy-Konstruktor zum Initialisieren eines Kontos mit einem bereits existierenden Konto.
- Eine Methode zum Klonieren eines Konto-Objektes.
- Eine Methode zum Überprüfen der Wertgleichheit zweier Konto-Objekte.
- Eine Methode, die den aktuellen Kontostand als String-Objekt zurückliefert.
- Eine Methode zum Einzahlen eines bestimmten Geldbetrages auf ein Konto. Dabei soll gelten: Wenn der Kontostand einmal den Wert von 10000 überschreitet, wird dem Konto im Folgenden ein Kreditrahmen von 3000 eingeräumt.
- Eine Methode zum Abheben eines bestimmten Geldbetrages von einem Konto.
- Eine Methode, die den aktuellen Kontostand als Wert liefert
- Eine Methode zum Überweisen eines bestimmten Geldbetrages von einem Konto auf ein anderes

Schreiben Sie weiterhin ein Programm zum Testen der Klasse.

Aufgabe 4 (komplexeres Programm)

Rushhour ist ein Spiel für einen einzelnen Spieler. Auf einem rechteckigen Spielbrett, das aus einzelnen Kacheln besteht, wird initial eine Menge an Autos platziert. Jedes Auto überdeckt mindestens zwei Kacheln. Ein Auto mit mehr als zwei Kacheln darf dabei keine Ecken aufweisen. Liegen die Kacheln, die ein Auto überdeckt, nebeneinander, ist es ein horizontal verschiebbares Auto. Liegen die Kacheln übereinander, ist es ein vertikal verschiebbares Auto. Ein Auto wird als Hauptauto gekennzeichnet. Ziel des Spiels ist es nun, die Autos so zu verschieben, bis irgendwann das Hauptauto (im Bild bspw. das rote Auto) den rechten Rand berührt.

Horizontale Autos dürfen dabei nur nach links und rechts, vertikale Autos nur nach oben und unten verschoben werden. Ein Auto darf nicht über den Rand hinweg oder wenn ein anderes Auto im Wege steht, verschoben werden.



Entwickeln Sie ein Java-Programm, mit dem ein Benutzer Rushhour spielen kann. Die Ausgangsstellung der Autos sei dabei in einem von Ihnen zu wählenden Datenformat festgelegt. Die Autoteile werden dabei durch jeweils gleiche Zeichen repräsentiert; das Hauptauto durch das Zeichen `*`. Die Stellung der Autos in der oberen Abbildung entspräche z.B. folgendem Kodierung:

```
122_3_
145_36
145**6
7778_6
_98aa
bb9cc_
```

Ein `*` steht dabei für das Hauptauto; `_` kennzeichnen leere Felder.

Der Benutzer soll nun so lange Autos verschieben können (durch Angabe von Auto und Richtung), bis das Hauptauto den rechten Rand erreicht. Im Folgenden wird ein beispielhafter Programmablauf skizziert (Benutzereingaben in `<>`):

Vererbung, Zugriffsrechte

Bearbeitung: KW 49
Testat: KW 51

```
+-----+-----+-----+-----+-----+-----+
| 1 | 2 | 2 |   | 3 |   |   |
+-----+-----+-----+-----+-----+-----+
| 1 | 4 | 5 |   | 3 | 6 |   |
+-----+-----+-----+-----+-----+-----+
| 1 | 4 | 5 | * | * | 6 |   |
+-----+-----+-----+-----+-----+-----+
| 7 | 7 | 7 | 8 |   | 6 |   |
+-----+-----+-----+-----+-----+-----+
|   |   | 9 | 8 | a | a |   |
+-----+-----+-----+-----+-----+-----+
| b | b | 9 | c | c |   |   |
+-----+-----+-----+-----+-----+-----+
```

Wahl eines Autos (Buchstabe angeben): <c>

Richtung (l=links, r=rechts, u=hoch, d=runter, q=Autowechsel): <r>

```
+-----+-----+-----+-----+-----+-----+
| 1 | 2 | 2 |   | 3 |   |   |
+-----+-----+-----+-----+-----+-----+
| 1 | 4 | 5 |   | 3 | 6 |   |
+-----+-----+-----+-----+-----+-----+
| 1 | 4 | 5 | * | * | 6 |   |
+-----+-----+-----+-----+-----+-----+
| 7 | 7 | 7 | 8 |   | 6 |   |
+-----+-----+-----+-----+-----+-----+
|   |   | 9 | 8 | a | a |   |
+-----+-----+-----+-----+-----+-----+
| b | b | 9 |   | c | c |   |
+-----+-----+-----+-----+-----+-----+
```

Richtung (l=links, r=rechts, u=hoch, d=runter, q=Autowechsel): <q>

Wahl eines Autos (Buchstabe angeben): 8

Richtung (l=links, r=rechts, u=hoch, d=runter, q=Autowechsel): <d>

```
+-----+-----+-----+-----+-----+-----+
| 1 | 2 | 2 |   | 3 |   |   |
+-----+-----+-----+-----+-----+-----+
| 1 | 4 | 5 |   | 3 | 6 |   |
+-----+-----+-----+-----+-----+-----+
| 1 | 4 | 5 | * | * | 6 |   |
+-----+-----+-----+-----+-----+-----+
| 7 | 7 | 7 |   |   | 6 |   |
+-----+-----+-----+-----+-----+-----+
|   |   | 9 | 8 | a | a |   |
+-----+-----+-----+-----+-----+-----+
| b | b | 9 | 8 | c | c |   |
+-----+-----+-----+-----+-----+-----+
```

Richtung (l=links, r=rechts, u=hoch, d=runter, q=Autowechsel): <q>

Wahl eines Autos (Buchstabe angeben): 7

Richtung (l=links, r=rechts, u=hoch, d=runter, q=Autowechsel): <r>

Vererbung, Zugriffsrechte

Bearbeitung: KW 49
Testat: KW 51

+	+	+	+	+	+	+	+
	1		2		2		
					3		
+	+	+	+	+	+	+	+
	1		4		5		
					3		6
+	+	+	+	+	+	+	+
	1		4		5		*
					*		6
+	+	+	+	+	+	+	+
			7		7		
					7		6
+	+	+	+	+	+	+	+
					9		8
					a		a
+	+	+	+	+	+	+	+
	b		b		9		8
					c		c
+	+	+	+	+	+	+	+

...

bs das Auto rausgefahren wird.

Einen Eindruck können sie unter <https://www.playit-online.de/denkspiele-onlinespiele/rush-hour/> gewinnen.