Check und Prepare

Vorbereitende Übungen

TECHNISCHE UNIVERSITÄT DARMSTADT

Autoren: Nhan Huynh und Darya Nikitina
Fachbereich: Informatik
Übungsblatt: 04
Version: 20. November 2021
Semesterübergreifend

V1 Grundbegriffe

★★★

Erklären Sie kurz in eigenen Worten die Unterschiede der folgenden Konzepte zueinander:

1. Klasse vs. Objekt
2. Objekt- vs. Klassenmethoden
3. Abstrakte Klassen vs. Interfaces
4. Überladen von Methoden vs. Überschreiben von Methoden

V2 Brumm, Brumm, Brumm

★★★

Schreiben Sie eine Klasse `Car` zur Repräsentation von Autos, die folgende Anforderungen erfüllen soll:

- Ein Auto hat einen Namen vom Typ `String` und einen Kilometerstand (`mileage`) vom Typ `double`. Beide Attribute sollen `private`, nicht `public`, sein.
- Der Konstruktor soll einen `String` als Parameter erhalten, der den Namen des Autos angibt. Der Konstruktor soll den Namen des Autos setzen und den Kilometerstand auf `0.0` setzen.
- Schreiben Sie die Methoden `public double getMileage()` und `public String getName()`. Diese liefern die entsprechenden Attribute der Klasse `Car` zurück.
- Schreiben Sie die Methode `public void drive(double distance)`, die eine Distanz in Kilometern als Argument erhält und auf den alten Kilometerstand addiert.

V3 Interfaces

★★★

Gegeben sei folgendes Interface

```
1 public interface I1 {
2
3   void m1();
4
5   int[] m2(double param1);
6 }
```

Schreiben Sie nun eine Klasse `C1`, die das Interface `I1` implementiert. Sofern im Body einer Methode eine Rückgabe erwartet wird, geben sie `null` zurück.

Übungsblatt 04: Vorbereitende Übungen von Nhan Huynh und Darya Nikitina

V4 Vererbung

Gegeben seien folgende zwei Klasse, die sich im gleichen Package befinden:

```
1  public class B1 {
2
3    public float f;
4    private boolean b;
5    protected byte by;
6
7    int m1() {
8      return -1;
9    }
10
11   private int m2() {
12     return -2;
13   }
14 }
15
16 public class B2 extends B1 {
17
18   int i;
19   public double d;
20
21   protected int m2() {
22     return 2;
23   }
24
25   public static void main(String[] args) {
26     B2 obj = new B2();
27   }
28 }
```

Betrachten Sie die `main`-Methode der Klasse `B2`. Auf welche Attribute können Sie mit dem Objekt `obj` zugreifen? Welche Methoden können Sie aufrufen und welchen Wert geben die Methoden zurück?

V5 Gleicher Abstand

Schreiben Sie eine Methode `static boolean evenlySpaced(int a, int b, int c)`, welche genau dann `true` zurückliefert, wenn der Abstand zwischen dem kleinsten und dem mittleren Element genauso groß ist wie der Abstand zwischen dem mittleren und dem größten Element. Dabei kann jeder der Parameter `a`, `b` oder `c` das kleinste, mittlere oder größte Element sein. Die Klasse, zu der die Methode gehört, muss nicht implementiert werden

V6 Zahlen aneinanderreihen

Schreiben Sie eine Methode `static int appendIntegers(int[] a)`. Die Methode bekommt ein `int`-Array übergeben und liefert eine Zahl zurück, die entsteht wenn man alle Zahlen des übergebenen Arrays aneinanderreiht. Der Aufruf `appendIntegers (1,2,3)` liefert `123` zurück. Der Aufruf `appendIntegers(43,2,7777)` liefert `4327777` zurück. Sie dürfen nur Variablen von Typ `int` in ihrer Implementation verwenden, keine `Strings` oder Ähnliches. Schleifen sind natürlich erlaubt.

Übungsblatt 04: Vorbereitende Übungen von Nhan Huynh und Darya Nikitina

V7 Zahlen einsortieren

Gegeben sei folgende Klasse

```
1 public class ArrayTuple {
2
3   public int[] iArr;
4   public double[] dArr;
5 }
```

Erweitern Sie diese Klasse um eine `public`-Klassenmethode `ArrayTuple split(double[] a)`. Die Methode liefert ein neues Objekt von Typ `ArrayTuple` zurück, in dessen `int`-Array sich alle ganzen Zahlen aus dem übergebenen Array `A` befinden. Im `double`-Array befinden sich die restlichen Zahlen aus dem übergebenen Array.

V8 Statischer und dynamischer Typ

```
1 public class Alpha {
2
3   protected int v;
4
5   public Alpha(int a) {
6     v = a;
7   }
8 }
9
10 public class Beta extends Alpha {
11
12   public Beta(int b, int c) {
13     super(b);
14     v = c;
15   }
16
17   public Alpha x1() {
18     super.v++;
19     return new Beta(0, v);
20   }
21
22   public int x2(int x) {
23     return x + ++v + v++;
24   }
25 }
```

```
26
27 public class Gamma extends Beta {
28
29   private short y;
30
31   public Gamma(int d, int e) {
32     super(d, e);
33     y = (short) d;
34   }
35
36   public int x2(int x) {
37     return x - y;
38   }
39
40   public static void main(String[] args) {
41     Alpha a = new Alpha(7);
42     Beta b = new Beta(0, 1);
43     Gamma g = new Gamma(9, 2);
44     a = b.x1();
45     int t = b.x2(5);
46     a = new Beta(10, 12).x1();
47     b = g;
48     int r = g.x2(50);
49   }
50 }
```

Hinweis: Nach Zeile X heißt unmittelbar nach X, noch vor Zeile $X + 1$.

(1) Welchen statischen und dynamischen Typ haben `a`, `b` und `g` nach Zeile 40?

(2) Welchen statischen und dynamischen Typ hat `a` und welchen Wert hat `a.v` nach Zeile 41?

(3) Welchen Wert haben `t` und `b.v` nach Zeile 42?

(4) Welchen statischen und dynamischen Typ haben `a`, `b` und welchen Wert hat `a  v` nach Zeile 44?

(5) Welchen Wert haben `r` und `b.v` nach Zeile 45?

Übungsblatt 04: Vorbereitende Übungen von Nhan Huynh und Darya Nikitina

V9 Klassen, Interfaces und Methoden ☆☆☆

V9.1

Schreiben Sie ein `public`-Interface `A` mit einer Objektmethode `m1`, die Rückgabetyp `double`, einen `int`-Parameter `n` und einen char-Parameter `c` hat.

V9.2

Schreiben Sie ein `public`-Interface `B`, das von `A` erbt und zusätzlich eine Objektmethode `m2` hat, die keine Parameter hat und einen `String` zurückliefert.

V9.3

Schreiben Sie eine `public`-Klasse `XY`, die `A` implementiert, aber `m1` nicht. Klasse `XY` soll ein `protected`-Attribut `p` vom Typ `long` haben sowie einen `public`-Konstruktor mit Parameter `q` vom Typ `long`. Der Konstruktor soll `p` auf den Wert von `q` setzen. Weiter soll `XY` eine `public`-Objektmethode `m3` mit Rückgabetyp `void` und Parameter `XY` vom Typ `XY` haben, aber nicht implementieren.

V9.4

Schreiben Sie eine `public`-Klasse `YZ`, die von `XY` erbt und `B` implementiert. Die Methode `m1` soll `n+c+p` zurückliefern und `m2` den String `"Hallo"`. `m3` soll den Wert `p` von `XY`auf den Wert `p` des eigenen Objektes addieren. Der Konstruktor von `YZ` ist `public`, hat einen `long`-Parameter `r` und ruft damit den Konstruktor von `XY` auf.

V10 Jedes dritte Element ☆☆☆

Gegeben sei eine Klasse `X`. Schreiben Sie für diese Klasse die `public`-Objektmethode `foo`. Diese hat ein Array `a` von Typ `int` als formalen Parameter und liefert ein anderes Array `b` vom Typ `int` zurück, das aus `a` entsteht, indem jedes dritte Element gelöscht wird, das heißt, die Elemente von `a` an den Indizes $0, 3, 6, 9, \ldots$ werden nicht nach `b` kopiert, alle anderen Elemente von `a` werden in derselben Reihenfolge, wie sie in `A` stehen, nach `b` kopiert. Weitere Elemente hat `b` nicht. Sie dürfen voraussetzen, dass `A` mindestens Länge 2 hat und ungleich `null` ist. Sie dürfen einfach Operator `=` für das Kopieren von Elementen verwenden.

> **Hinweis:** Überlegen Sie sich die Gesetzmäßigkeit, nach der die Indizes $1, 2, 4, 5, 7, 8, \ldots$ in `a` auf die Indizes $0, 1, 2, 3, 4, 5, \ldots$ in `b` abzubilden sind. Für die Länge von `b` werden Sie eine Fallunterscheidung benötigen, je nachdem, welchen Rest `a.length` dividiert durch 3 ergibt. Denken Sie auch an die letzten beiden Elemente von `a`.