



Abgabe für Rückmeldungen bis zum 18.01.2021 (08:00 Uhr), Besprechung ab dem 18.01.2021.

Aufgabe 08-0 (Schlüsselwort `super` und Methodenüberschreibung)

Betrachten Sie folgende Klasse.

```
1 class Klasse {  
2     void ausgabe1() {  
3         System.out.println("Ausgabe 1");  
4     }  
5     void ausgabe2() {  
6         System.out.println("Ausgabe 2");  
7     }  
8  
9     void methode() {  
10        ausgabe1();  
11    }  
12 }
```

Betrachten Sie folgende Subklassen.

```
0. class Subklasse extends Klasse {  
    void methode() {  
        super.methode();  
    }  
}
```

```
3. class Subklasse extends Klasse {  
    void ausgabe1() {  
        System.out.println(  
            "Ausgabe 1 mod");  
    }  
}
```

```
1. class Subklasse extends Klasse {  
    void methode() {  
        methode();  
    }  
}
```

```
4. class Subklasse extends Klasse {  
    void ausgabe1() {  
        System.out.println(  
            "Ausgabe 1 mod");  
    }  
  
    void methode() {  
        super.methode();  
    }  
}
```

```
2. class Subklasse extends Klasse {  
    void methode() {  
        ausgabe2();  
    }  
}
```

Sei `s` ein Zeiger auf eine Instanz von `Subklasse`. Entscheiden, begründen und validieren Sie, welches Verhalten beziehungsweise welche Kommandozeilenausgabe von `s.methode()` jeweils zu erwarten ist.

Aufgabe 08-1 (Zugriff mit polymorphen Zeigern)

Betrachten Sie folgende Klassen.

```
1 class Klasse {  
2     int a;  
3     void methode0() { ... }  
4 }
```

```
1 class Subklasse0 extends Klasse {  
2     int b;  
3     void methode1() { ... }  
4 }
```

```
1 class Subklasse1 extends Klasse {  
2     int c;  
3     void methode2() { ... }  
4 }
```

Entscheiden, begründen und validieren Sie, ob folgende Listings jeweils fehlerfrei sind.

0. Klasse k = **new** Subklasse0();
k.methode1();

4. Subklasse1 k = **new** Klasse();
k.a = 17;

1. Klasse k = **new** Subklasse0();
k.a = k.b;

5. Klasse k = **new** Subklasse1();
k.a = ((Subklasse1)k).c;

2. Klasse k = **new** Subklasse1();
k.a = k.b;

6. Subklasse0 k = **new** Subklasse1();
k.a = 0;

3. Subklasse1 k = **new** Subklasse1();
k.c = 42;

7. Klasse k = **null**;
k.methode0();

Aufgabe 08-2 (Selection Sort)

Versehen Sie die Selection-Sort-Implementierung aus der Vorlesung mit einer Kommandozeilenausgabe derart, daß bei jeder Iteration der Array, der jeweilige Minimalwert rechts vom Markierer und die Markiererposition sinnvoll ausgegeben werden.

Präsenzaufgabe 08-3 (Mergesort)

Versehen Sie die Mergesort-Implementierung aus der Vorlesung mit einer Kommandozeilenausgabe derart, daß zu Beginn jedes `mergesort()`-Aufrufs der Eingabearray und am Ende jedes `mergesort()`-Aufrufs der sortierte Array ausgegeben werden.

Erweitern Sie die Parameterliste der `mergesort()`-Prozedur mit einem Parameter `tiefe` vom Typ `int`. Rufen Sie `mergesort()` in `main()` mit der `tiefe` 0 auf und innerhalb Ihrer `mergesort()`-Methode in beiden Fällen mit `tiefe + 1`. Dieses Idiom erlaubt es, die aktuelle Rekursionstiefe zu wissen. Nutzen Sie `tiefe`, um Ihre Kommandozeilenausgaben mit entsprechend vielen Tabulatorzeichen einzurücken. Ein solches Präfix mit `tiefe` vielen Tabulatorzeichen für Ihre Ausgaben können Sie beispielsweise mit folgendem Listing erzeugen.

```

1 String praefix = "";
2 for (int i = 0; i < tiefe; i++) {
3     praefix += "\t";
4 }

```

Präsenzaufgabe 08-4 (Zeitrechner)

Betrachten Sie folgendes lückenhafte Listing.

```

1 public class Uhrzeit {
2     private int h;
3     private int m;
4     private int s;
5
6     public Uhrzeit(int h, int m, int s) { ... }
7
8     public String to_string() {
9         return String.format("%02d:%02d:%02d", h, m, s);
10    }
11    public boolean ist_spaeter_oder_gleich(Uhrzeit z) { ... }
12    public boolean ist_frueher_oder_gleich(Uhrzeit z) { ... }
13    public boolean ist_gleich(Uhrzeit z) { ... }
14    public int abstand_sek(Uhrzeit z) { ... }
15 }

```

Diese Klasse beschreibt eine Uhrzeit (Stunde, Minute, Sekunde) an einem festen Tag, wobei alle ihre Instanzen als Uhrzeiten am selben Tag interpretiert werden sollen und wir Spezialfälle wie Schaltsekunden nicht beachten.

Beim Instanzieren der Klasse werden dem Konstruktor `Uhrzeit()` die drei Parameter `h`, `m`, `s`, die jeweils für Stunde, Minute und Sekunde stehen, übergeben. Es soll geprüft werden, ob die Eingabewerte im korrekten Wertebereich liegen ($\{0, \dots, 23\}$, $\{0, \dots, 59\}$ beziehungsweise $\{0, \dots, 59\}$). Liegen diese außerhalb, sollen sie auf 0 gesetzt werden, bevor sie in die Instanzvariablen geschrieben werden.

Die `ist_*`-Methoden dienen dem Vergleich zweier Uhrzeiten. Beachten Sie hier bei der Implementierung, daß es sich um zwei Uhrzeiten am *selben* Tag handelt und wir keine Übergänge über die Mitternachtsgrenze haben. Zum Beispiel soll die Uhrzeit 00:00:01 früher sein als die Uhrzeit 23:59:59.

Mit der Methode `abstand_sek()` soll der Abstand zwischen zwei Uhrzeiten in Sekunden berechnet werden. Beachten Sie auch hier die obige Bemerkung.

Vervollständigen Sie die Methodenrumpfe und testen Sie die Klasse entsprechend (mit einer sinnvollen Kommandozeilenausgabe) in Ihrer `main`-Methode. Nehmen Sie den Aufruf der Methode `String.format()` erst einmal nur als gegeben hin, falls Sie ihn nicht verstehen.