

Übung 1: *Enum* und *Generics*

Aufgabe 1: Enum

a) Folgende Frage soll beantwortet werden:

Warum ist bei einem **enum** ein Vergleich der Art (`meineEnumFarbVariable == Farbe.GELB`) so ohne weiteres möglich ?

b) Bei einem unserer Kunden wird viel mit Münzen gearbeitet.

Er will in den Programmen mit Ausdrücken in der Art `"FUENFER"`, `"ZEHNER"`, `"ZWANZIGER"`, `"FUENFZIGER"`, `"EIN_FRAENKLER"` und `"ZWEI_FRAENKLER"` arbeiten können ;-)

Des Weiteren haben die Münzen neben der Eigenschaft des Wertes in Rappen noch die Eigenschaft des Gewichtes in Gramm.

Es soll ein entsprechender **enum** definiert werden, welcher diesen Anforderungen genügt.

c) Im Weiteren soll auch die Münz-Farbe mit den Werten für das Farb-Tripel "Rot-Grün-Blau" pro Münze definiert sein.

Der Kunde will, dass wenn man eine Münze via `System.out` auf die Konsole ausgibt, der Output für einen Fünfer wie folgt aussieht:

```
"FUENFER: Gewicht=2 Farbe:GOLD=154-114-50"
```

Es sollen nun die dazu nötigen *enum*'s erstellt werden, sodass folgende Test-Sequenz funktioniert:

```
Muenze meineMuenzVariable = Muenze.FUENFER;
if (meineMuenzVariable == Muenze.FUENFER) {
    System.out.println("Das ist ein FUENFER ;-);
}
System.out.println(meineMuenzVariable);
```

```
/* Session-Log:
```

```
Das ist ein FUENFER ;-);
FUENFER: Gewicht=2 Farbe:GOLD=154-114-50
```

```
*/
```

Java – Advanced Concepts

Aufgabe 2: Generics

a) Generics und Collections

Die Klassen *Person*, *Student* und *PersonenListe* sollen erstellt werden, sodass nachfolgende Sequenz fehlerlos funktioniert gemäss dem *Session-Log*.

Wobei:

- *add()* die Personen nach Namen sortiert in die Liste einfügt.
- *merge()* die eigene mit der übergebenen Liste merged.
- die Klasse *Person* das Interface *java.lang.Comparable* implementiert (um nach Namen sortieren zu können).

```

PersonenListe<Person> personenListe = new PersonenListe<Person>();

personenListe.add(new Person("Hans"));
personenListe.add(new Person("Annelis"));
personenListe.print("Personen-Liste ");

PersonenListe<Student> studentenListe = new PersonenListe<Student>();
studentenListe.add(new Student("Max", 4711));
studentenListe.add(new Student("Anna", 4712));
studentenListe.add(new Student("Clara", 4713));
studentenListe.print("Studenten-Liste ");

PersonenListe<Person> mergeListe = new PersonenListe<Person>();
int length = personenListe.merge(studentenListe, mergeListe);
System.out.print("Number of merged Persons : " + length);
mergeListe.print("Merge-Liste ");

/* Session-Log:
Personen-Liste           : Annelis Hans
Studenten-Liste         : Anna:4712 Clara:4713 Max:4711
Number of merged Persons : 5
Merge-Liste             : Anna:4712 Annelis Clara:4713 Hans Max:4711
*/

```

Hinweise:

- Die Klassen sollen nur minimal implementiert werden. Sie sollen nur soweit funktionieren was gemäss *Session-Log* benutzt wird.
- Mit *java.util.Collections.sort(List<T> list)* können Listen sortiert werden.
- Auf dem Skripte-Server befindet sich die Ausgangslage (*Personen.java*). Daran sollen die nötigen Anpassungen und Erweiterungen durchgeführt werden.

b) Legacy -> Generic

Das File *BinarySearchTree.java* (auf dem Skripte-Server) soll von Java 1.4 nach Java 1.5 portiert werden.

Ziel: in *BinarySearchTree.main()* soll der der Binary-Tree deklariert werden mit:

```
BinarySearchTree<Integer, String> intTree
```

Am Schluss sollen keine Warnungen mehr vorkommen (ohne `@SuppressWarnings` ;-).