Arbeitsblatt 5

# Aufgabe 1

Mit Java 8 wurden mehrere neue funktionale Interfaces, darunter Consumer<T> und Predicate<T> , die die Arbeit mit Collections und Maps vereinfachen sollen, eingeführt. Das Interface Consumer  definiert die Methode void accept(T t) , in der Operationen auf das im Aufruf übergebene Objekt oder auf externe Objekte durchgeführt werden. Die Methode boolean test(T t)  des Predicate-Interface gibt true zurück, wenn das im Aufruf übergebene Objekt bestimmte Kriterien erfüllt. Beide Methoden definieren einen Parameter vom generischen Typ des Typparameters der Interfaces.

Eine Klasse BuchVerkauf  soll dazu dienen, neue Preise für die Kindle Editions von Büchern in Abhängigkeit von deren Seitenanzahl zu bestimmen. Hat ein Buch mehr als 750 Seiten, soll der ursprüngliche Preis um 12% herabgesetzt werden, und ansonsten um 9%.

Definieren Sie eine weitere Klasse KindleEdition  (deren Instanzen Bücher repräsentieren) mit den Instanzfeldern String titel , double preis  und int seitenanzahl und den zugehörigen Getter- und Setter-Methoden für den Zugriff darauf. Eine Instanzmethode void anzeigeBuch() soll den Titel und den Preis für ein bestimmtes Buch am Bildschirm ausgeben. Mittels der generischen Methode static <T> void anzeigeEigenschaft(T eigenschaft)  soll ermöglicht werden, eine Bucheigenschaft von einem beliebigen Typ anzuzeigen.

Erzeugen Sie in der Klasse BuchVerkauf String -, Double - und Integer -Arrays mit den Titeln, Preisen und der Seitenanzahl von Büchern, deren Elemente mit Hilfe von Zugriffsmethoden für ein Array KindleEdition buecher[] = new KindleEdition[4]  gesetzt werden, wie zum Beispiel mit:

```
String[] titelArray = {"Java 7 Übungsbuch Band I",
"Java 7 Übungsbuch Band II", "Android 4 Übungsbuch",
"Servlets und JavaServer Pages“};

Double[] preisArray = {29.95, 29.95, 24.95, 16.95};

Integer[] seitenArray = {806, 796, 454, 748};

for(int i=0; i < buecher.length; i++) {
      buecher[i] = new KindleEdition();
      buecher[i].setTitel(titelArray[i]);
      buecher[i].setPreis(preisArray[i]);
      buecher[i].setSeitenanzahl(seitenArray[i]);
}
```

Definieren Sie zwei Methoden

```
static void berechnePreisKindleEdition (KindleEdition buch ,
Predicate<KindleEdition> predicate , Consumer<KindleEdition> consumer)

static void anzeigeAttributemit-Consumer(List<KindleEdition> liste ,
Consumer<String> consumer1 , Consumer<Double> consumer2)
```

für die Kalkulation von neuen Preisen und die Anzeige der Bucheigenschaften titel und preis.

Die erste der Methoden benutzt einen Predicate, um zu entscheiden, wie der Preis abzuändern ist, und einen Consumer, um die Änderung durchzuführen. Rufen Sie daran ihre Methoden test() und accept() auf, in denen eine Instanz vom Typ des Schnittstellenparameters KindleEdition übergeben wird.

Die zweite Methode soll an den übergebenen Consumer -Objekten deren accept() - Methode aufrufen, in der die Bucheigenschaften für eine Anzeige übergeben werden. Benutzen Sie für die Anzeige von Titeln und Preisen vor und nach Änderungen alternativ die Methoden System.out.println() bzw. anzeigeBuch() als auszuführende Operation für die accept() - Methode.

Erzeugen Sie in Deklarationen mit der Angabe ihres parametrisierten Target-Typs Lambda-Ausdrücke wie zum Beispiel mit:

```
Consumer<String> consumer1 = titel ->
System.out.println("Titel: " + titel);

Predicate<KindleEdition> predicate2 = (KindleEdition buch) ->
buch.getSeitenanzahl() >= 750;

Consumer<KindleEdition> consumer3 = (KindleEdition buch) ->
buch.setPreis(buch.getPreis() – buch.getPreis()*9/100.)
```

um diese in den Methodenaufrufen per Referenz zu übergeben, oder weisen Sie diese direkt den Methodenparameter zu.

Hinweise für die Programmierung:
Die generischen Interfaces Predicate<T> und Consumer<T> befinden sich im Paket java.util.function . Weil es funktionale Interfaces sind, können diese als Target-Typen für Lambda-Expressions benutzt werden.