Fachhochschule Nordwestschweiz

Zwischenprüfung OOP2

Dauer: 90 Minuten

Name	
Vorname	
Punkte	
Note	

	Punkte	**Erreichte Punkte**
Aufgabe 1: Collections	8	
Aufgabe 2: anonyme Klassen, Lambdas, Streams	10	
Aufgabe 3: Generische Klasse	19	
Aufgabe 4: generische Methoden	9	
Total	**46**	

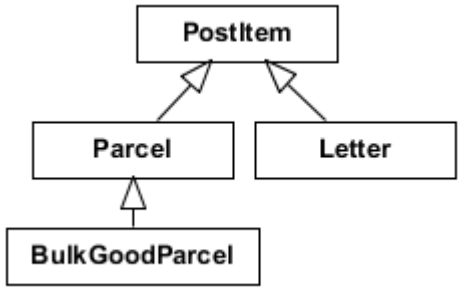
- Versuchen Sie wenn möglich all ihre Antworten in dem vorgegebenen Feld anzugeben. Falls dies nicht möglich ist, verwenden Sie für jede Aufgabe ein Blatt, welches Sie mit der Nummer der Aufgabe klar anschreiben, und dann auch immer noch die Nummer der Teilaufgabe (x.x)
- Als Hilfsmittel erlaubt sind **sämtliche schriftlichen Unterlagen und ihr Computer**.
- **Nicht erlaubt** ist jede Art von Netzzugriff. Das **WLAN muss ausgeschaltet sein**. Insbesondere sind alle Programme, die die Kommunikation mit anderen Personen ermöglichen**,** wie beispielsweise Mail, Twitter, Skype, WhatsApp nicht erlaubt.
- Bei unerlaubter Nutzung des Computers wird die Prüfung mit der Note 1 bewertet. Für die anderen Studierenden wird der weitere Einsatz des Computers untersagt.
- Bei jeder Teilaufgabe ist angegeben, wie viele Punkte erreicht werden können.
- Abschreiben oder Betrügen in irgendeiner Weise resultiert in der Note 1.

Gegeben ist die Klasse **`PostItem`**. Diese Klasse wird in allen Aufgaben verwendet.

```
public class PostItem {
   private final String receiver;
   private final double weight;

   //Versandart (A = A-post, B = B-post, E = Express)
   private final char dispatchMode;

   public PostItem(String receiver, double weight, char dispatchMode){
      this.receiver     = receiver;
      this.weight       = weight;
      this.dispatchMode = dispatchMode;
   }

   public PostItem() {
      this("Santa Clause", 0.2, 'E');
   }

   @Override
   public String toString() {
      return "PI[" + receiver + ", "+ weight + ", " + dispatchMode + "]";
   }

   // alle notwendigen Setter und Getter
}
```

Die Klasse `PostItem` (Poststück) hat die folgenden Unterklassen

- `Letter`: Brief
- `Parcel`: Paket
- `BulkGoodParcel`: Sperrgut-Paket



Aufgabe 1: Collections 8 Punkte

In dieser Aufgabe sollen Sie angeben, welche Ausgabe vom gegebenen Code erzeugt wird. Bei Collections bei denen die Reihenfolge nicht genau bestimmbar ist z.B. `HashSet`, muss keine bestimmte Reihenfolge bei der Angabe der Elemente beachtet werden.

a) Was ist die Ausgabe des folgenden Programms? (2 Punkte)

```
public static void main(String[] args) {
   List<PostItem> listA= new ArrayList<>();
   listA.add(new PostItem("Paul",  4.3,  'A'));
   listA.add(new PostItem("Anna",  0.5,  'B'));
   listA.add(new PostItem("Karl",  0.91, 'E'));
   listA.add(new PostItem("Peter", 3.5,  'B'));
   for(PostItem item : listA){
      System.out.println(item);
   }
}
```

Ausgabe:

b) Was ist die Ausgabe des folgenden Programms? (2 Punkte)

```
public static void main(String[] args) {
   List<PostItem> listB= new ArrayList<>();
   listB.add(new PostItem("Paul", 4.3,  'A'));
   listB.add(new PostItem("Anna", 0.5,  'B'));
   listB.add(new PostItem("Karl", 0.91, 'E'));
   listB.add(new PostItem("Peter", 3.5, 'B'));
   for (int i = listB.size() - 1; i > 0; i--) {
      System.out.println(listB.get(i));
   }
}
```

Ausgabe:

c) Was ist die Ausgabe des folgenden Programms? (2 Punkte)

```
public static void main(String[] args) {
   HashMap<Character, PostItem> map= new HashMap<>();
   PostItem i1= new PostItem("Paul",  4.3,  'A');
   PostItem i2= new PostItem("Anna",  0.5,  'B');
   PostItem i3= new PostItem("Karl",  0.91, 'E');
   PostItem i4= new PostItem("Peter", 3.5,  'B');
   map.put(i1.getDispatchMode(), i1);
   map.put(i2.getDispatchMode(), i2);
   map.put(i3.getDispatchMode(), i3);
   map.put(i4.getDispatchMode(), i4);

   for(char k : map.keySet()){
     System.out.println(map.get(k).getReceiver());
   }
}
```

Ausgabe:

d) Was ist die Ausgabe des folgenden Programms? (2 Punkte)

```
public static void main(String[] args) {
   Set<PostItem> set= new TreeSet<>(new Comparator<PostItem>() {

      @Override
      public int compare(PostItem o1, PostItem o2) {
         return Character.compare(o1.getDispatchMode(),
                                  o2.getDispatchMode());
      }
   });

   set.add(new PostItem("Karl",  0.91, 'E'));
   set.add(new PostItem("Anna",  0.5,  'B'));
   set.add(new PostItem("Paul",  4.3,  'A'));
   set.add(new PostItem("Peter", 3.5,  'B'));

   for(PostItem item : set){
      System.out.println(item);
   }
}
```

Ausgabe:

Aufgabe 2: Anonyme Klassen und Lambdas 10 Punkte

a) anonyme Klassen (5 Punkte)

Ergänzen Sie die Methode `printSortedList`, welche eine Liste von PostItems zuerst nach Gewicht (`weight`) **sortiert** und danach **ausgibt**. Es spielt keine Rolle ob die Elemente aufsteigend oder absteigend sortiert sind.

- Verwenden Sie für die Sortierung `Collections.sort(...)` mit einer anonymen Inner Class.
- Verwenden Sie für die Ausgabe die `forEach`-Methode aus.

```
public static void printSortedList(List<PostItem> list){




}
```

b) Streams und Lambda Ausdrücke 1 (5 Punkte)

Schreiben Sie eine Methode `getExpressWeight`, welche für eine Liste von PostItems die Summe der Gewichte (`weight`) aller Express-Elemente (mit `dispachMode` 'E') berechnet und zurückgibt.

Verwenden Sie dafür die Methode `stream()` von `Collection` sowie Lambda-Ausdrücke.

```
  public static double getExpressWeight(List<PostItem> list){




  }
```

Aufgabe 3: Generische Klasse 19 Punkte

a) Generische Klasse schreiben (13 Punkte)

Schreiben Sie hier die generische Klasse `MeanOfTransport`, die PostItems (oder Subklassen von PostItems) verwalten kann. Die Klasse soll die folgenden Methoden anbieten:

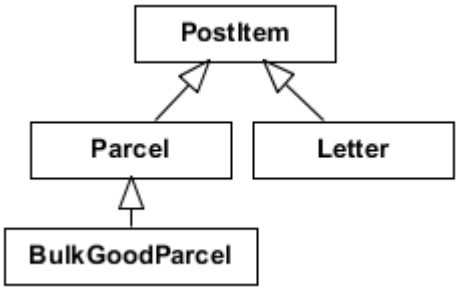
- **add**: diese Methode fügt ein neu zu verwaltendes Element in eine interne Datenstruktur ein. Die Datenstruktur können Sie selbst bestimmen.
- **getNumberOfItems**: diese Methode soll zurückgeben wie viele Poststücke per A-Post verschickt werden.
- **getMaxElement**: diese Methode gibt das Element mit dem grössten Gewicht zurück.
- **getPostItems**: diese Methode erhält als Parameter einen Empfänger (`receiver`) und gibt die Liste von Elementen zurück, welche diesen Empfänger haben.

Hinweis: In dieser Aufgabe ist es Ihnen freigestellt ob Sie "External Iteration" oder "Internal Iteration" verwenden.

b) Generische Klasse verwenden (6 Punkte)

Geben Sie für die folgenden Code-Abschnitte an, ob sie fehlerfrei sind. Falls sie nicht fehlerfrei sind, geben Sie noch an, in welcher Zeile **der erste Fehler** auftritt.

Der Konstruktor für `PostItem` ist auf Seite 1 gegeben. Die Unterklassen besitzen jeweils einen Konstruktor, der die gleichen Parameter bekommt.



1 MeanOfTransport<Parcel> m1= new MeanOfTransport<>(); 2 m1.add(new Parcel("Peter",3.2,'A')); 3 PostItem p= m1.getMaxElement();	☒ funktioniert □ funktioniert nicht Zeile _____
1 MeanOfTransport<Parcel> m2= new MeanOfTransport<>(); 2 m2.add(new PostItem("Peter",3.2,'A')); 3 Parcel p2= m2.getMaxElement();	□ funktioniert ☒ funktioniert nicht Zeile 2
1 MeanOfTransport<PostItem> m3= new MeanOfTransport<Letter>(); 2 m3.add(new Letter("Peter",3.2,'A')); 3 Letter l3= m3.getMaxElement();	☒ funktioniert □ funktioniert nicht Zeile _____
1 MeanOfTransport<? extends PostItem> m4= new MeanOfTransport<PostItem>(); 2 m4.add(new Letter("Peter", 3.2, 'A')); 3 PostItem p4= m4.getMaxElement();	□ funktioniert ☒ funktioniert nicht Zeile 2
1 MeanOfTransport<PostItem> m5 = new MeanOfTransport<>(); 2 m5.add(new Letter("Peter", 3.2, 'A')); 3 List<Letter> letters = m5.getPostItems("Peter");	□ funktioniert ☒ funktioniert nicht Zeile 3
1 MeanOfTransport<PostItem> m6= new MeanOfTransport<>(); 2 m6.add(new Parcel("Peter",3.2,'A')); 3 m6.add(new Letter("Peter", 0.2, 'B')); 4 Object o1 = m6.getMaxElement();	☒ funktioniert □ funktioniert nicht Zeile _____

Aufgabe 4: Generische Methoden 9 Punkte

Hinweis: In dieser Aufgabe ist es Ihnen freigestellt ob Sie "External Iteration" oder "Internal Iteration" verwenden.

a) Generische Methode 2 (5 Punkte)

Schreiben Sie hier die generische Methode `extractExpress`. Diese Methode erhält als Parameter ein Array von Elementen (`PostItem` oder Subklasse von `PostItem`) und gibt die Liste aller Elemente zurück, welche mit Express bezeichnet wurden.

Beispiele für die Verwendung der Methode.

```
Letter[] letters= new Letter[4];
letters[0]= new Letter("Paul", 4.3, 'A');
letters[1]= new Letter("Anna", 0.5, 'B');
letters[2]= new Letter("Karl", 0.91, 'E');
letters[3]= new Letter("Peter", 3.5, 'B');
List<Letter> expressLetter= extractExpress(letters);
// Element in expressLetter:  [PostItem [Karl, 0.91, E]]

Parcel[] parcels= new Parcel[2];
parcels[0]= new Parcel("Karl", 0.91, 'E');
parcels[1]= new Parcel("Peter", 3.5, 'E');
List<Parcel> expressParcels= extractExpress(parcels);
// Elemente in expressParcels :
// [PostItem [Karl, 0.91, E]], PostItem [Peter, 3.5, E]]
```

b) Generische Methode 2 (4 Punkte)

Schreiben Sie hier die generische Methode `removeDoubles`, welche eine Liste erhält und eine neue Liste zurückgibt, bei der alle Elemente nur einmal vorkommen.

Beispiele für die Verwendung der Methode.

```
List<Integer> intList= new ArrayList<>();
intList.add(1);
intList.add(4);
intList.add(4);
List<Integer> intOhneDoubles= removeDoubles(intList);  // [1, 4]

List<String> stringList= new LinkedList<>();
stringList.add("Hallo");
stringList.add("Hallo");
stringList.add("Hallo");
List<String> stringOhneDoubles= removeDoubles(stringList); // [Hallo]
```