Programmiermethodik
LV-Nr.: 703017-_
Sommersemester 2013
Übungstest
3.6.2013
Dauer: 60 Minuten

Name: _____
Matrikelnummer: _____

Lösungen

Dieser Test enthält 10 Seiten (inklusive Deckblatt) und 4 Probleme. Bitte überprüfen Sie die Seitenzahl. Notieren Sie Name und Matrikelnummer auf dem Deckblatt.

Es sind *keine* Unterlagen erlaubt!

Sollte Ihnen der Platz für Antworten zu knapp werden, verwenden Sie bitte die Rückseite.

Bitte tragen Sie *nichts* in die Tabelle rechts ein!

Problem	Punkte	Erzielte Punkte
1	5	
2	10	
3	10	
4	15	
Total:	40	

1. (5 Punkte)

```
1  import java.io.*;
2
3  class Foo {
4
5    public static void main(String [] args) {
6      Foo p = new Foo();
7      int result = p.max(9,3);
8      System.out.println(result);
9      result = max(9,3);
10     System.out.println(result);
11     int sum = p.add(5,5);
12   }
13
14   static int max(int...xs) {
15     int temp = 0;
16     for(int x : xs) {
17       if(x > temp) temp = x;
18     }
19     return temp;
20   }
21
22   int add(int x, int y) {
23     int result = x + y;
24     System.out.println(result);
25     return result;
26   }
27
28   double add(double x, double y) {
29     double result = x + y;
30     return result;
31   }
32 }
```

Listing 1: Foo.java

2. (10 Punkte) Kisten, Kisten, Kisten.

   Implementieren Sie ein Verwaltungssystem, das Gegenstände in Boxen verwalten kann. Eine Box kann dabei eine beliebige Anzahl an Gegenständen und kleineren Boxen beinhalten. Implementieren Sie zwei Klassen `Box` und `Item`, die das gegebene Interface `Boxable` implementieren. Die Funktionen `getCount` und `getDepth` geben die Anzahl der Gegenstände (ohne Boxen!), bzw. die maximale Verschachtelungstiefe zurück. Ein Gegenstand in einer Box in einer Box hat dabei die Verschachtelungstiefe 2. Die Funktion `listItems` gibt eine Datenstruktur nach Wahl zurück, die alle Items enthält. Stellen Sie sicher, dass die von `listItems` zurückgegebene Datenstruktur keine Duplikate enthält. Sie können davon ausgehen, dass `equals` entsprechend implementiert ist.

```
1  import java.util.Collection;
2
3
4  public interface Boxable {
5
6    public int getCount();
7
8    public int getDepth();
9
10   public Collection<Item> listItems();
11
12 }
```

Listing 2: Boxable.java

```
1  import java.util.Collection;
2  import java.util.TreeSet;
3  import java.util.Set;
4
5
6  public class Box implements Boxable {
7
8    Set<Boxable> content = new TreeSet<Boxable>();
9
10   public int getCount() {
11     return listItems().size();
12   }
13
14   public int getDepth() {
15     int maxDepth = 0;
16     for (Boxable boxable : content) {
17       maxDepth = Math.max(maxDepth, boxable.getDepth());
18     }
19     return maxDepth + 1;
20   }
21
22   public Collection<Item> listItems() {
23     Set<Item> allItems = new TreeSet<Item>();
24     for (Boxable boxable : content) {
25       allItems.addAll(boxable.listItems());
26     }
27     return allItems;
28   }
29
30 }
```

Listing 3: Box.java

```java
import java.util.Collection;
import java.util.LinkedList;
import java.util.List;


public class Item implements Boxable {

  public int getCount() {
    return 1;
  }

  public int getDepth() {
    return 0;
  }

  public Collection<Item> listItems() {
    List<Item> ret = new LinkedList<Item>();
    ret.add(this);
    return ret;
  }

}
```

Listing 4: Item.java

3. (10 Punkte) Überladen, Überschreiben, ...

   Betrachten Sie das Programm `Talk.java` - was gibt dieses Programm aus? Notieren Sie bitte jeweils die Zeilennummer und die entsprechende Ausgabe. Sie können ihre Antworten auch abkürzen (etwa Samantha: Carry wird zu S:C).

   17 Carry: Carry

   18 Samantha: Samantha

   19 Miranda: Carry

   20 Samantha: Miranda

   21 Samantha: Miranda

   22 Carry: Carry

   23 Samantha: Miranda

   24 Samantha: Carry

   25 Miranda: Carry

   26 Exception in thread "main"java.lang.ClassCastException: Miranda cannot be cast to Samantha at Talk.main(Talk.java:26)

4. Generics

(a) (5 Punkte)

```
class Pair {

  Object getKey() { return key; }

  void setKey(Object key) { this.key = key; }

  Object key;
  Object value;
}

class Test {

  void foo(Pair p) {
    String s = (String)p.getKey();
    p.setKey("foo");
  }
}
```

Listing 5: Pair.java

(b) (5 Punkte) Betrachten Sie das Progamm aus Listing 6.

```
1  class Wildcard {
2
3      //Note that we couldn't declare list as List<List<Object>>
4
5      public static List flatten2DList(List<List<?>> list) {
6          List flatList = new LinkedList();
7          for(List sublist : list) {
8              for( Object e : sublist) {
9                  flatList.add(e);
10             }
11         }
12         return flatList;
13     }
14
15     public static void main(String [] args) {
16         List<List<?>> list1 = new LinkedList<List<?>>();
17         LinkedList<Integer> list2 = new LinkedList();
18         list2.add(3);
19         list2.add(null);
20         LinkedList<Double> list3 = new LinkedList();
21         list3.add(5.5);
22         list1.add(list2);
23         list1.add(list3);
24         System.out.println(flatten2DList(list1));
25     }
26 }
```

Listing 6: Wildcard.java

- Warum der Kommentar? Warum kann `list` nicht als Liste von `Object` deklariert werden?
  Aufgrund des Aufrufs von

```
flatten2DList(list1)
```

- Unter der Annahme, das Programm kompiliert, welches Problem könnte auftreten?
  Das Ergebnis von *flatten(...)* könnte eine Liste mit inkompatiblen Typen (z.B String, Integer) sein.
- Unter der Annahme, dass der Parameter `list` in der Methode `flatten2DList` den Typ `List<List<? extends Number>>` bekommt, was müsste in der `main` Method angepasst werden, damit das Programm kompiliert?

```
1 List<List<?>>
```

⇓

```
1 List<List<? extends Number>>
```

(c) (5 Punkte) Gegeben Sei das Programm aus Listing 7.

```
1  class A { }
2
3  class B extends A { }
4
5  class C extends B { }
6
7
8  public class Carpet<V extends B> {
9
10   public Carpet() {
11
12   }
13
14   public <X extends V> Carpet<? extends V> method() {
15     // insert code here
16   }
17 }
```

Listing 7: Generics.java

Welche(s) der folgenden Statements kann/können in Zeile 15 eingefügt werden? Begründen Sie Ihre Antwort, insbesondere, warum ein Statement nicht erlaubt ist!

A `return new Carpet<X>();`

B `return new Carpet<V>();`

C `return new Carpet<A>();`

D `return new Carpet<B>();`

E `return new Carpet<C>();`