Objektorientierte Programmierung: Aufgabenblatt 9

Florian Ludewig (185722)

13. Juli 2020

Aufgabe 1

Pair.java

```java
class Pair<T, U> {
  private final T first;
  private final U second;

  Pair(T fst, U snd) {
    first = fst;
    second = snd;
  }

  T getFirst() {
    return first;
  }

  U getSecond() {
    return second;
  }

  public boolean equals(Pair<T, U> other) {
    return other.getFirst().equals(getFirst()) && other.getSecond().equals(
        getSecond());
  }
}
```

Relation.java

```java
import java.util.HashSet;
import java.util.Set;
import java.util.ArrayList;

public class Relation<T> {
  private Set<T> set = new HashSet<>();
  private Set<Pair<T, T>> relation = new HashSet<>();

  public void add(T item) {
    set.add(item);
  }

  public void addRelation(Pair<T, T> pair) {
    relation.add(pair);
  }

  public boolean isReflexive() {
    for (T item : set) {
      Pair<T, T> search = new Pair<T, T>(item, item);
      if (!containsPair(search))
        return false;
    }
    return true;
  }

```

```
26    public boolean isSymmetric() {
27      for (Pair<T, T> pair : relation) {
28        Pair<T, T> search = new Pair<T, T>(pair.getSecond(), pair.getFirst());
29        if (!containsPair(search))
30          return false;
31      }
32      return true;
33    }
34
35    public boolean isTransitive() {
36      for (Pair<Pair<T, T>, Pair<T, T>> candiate : getTransitiveCandidates())
          {
37        Pair<T, T> search = new Pair<T, T>(candiate.getFirst().getFirst(),
            candiate.getSecond().getSecond());
38        if (!containsPair(search))
39          return false;
40      }
41      return true;
42    }
43
44    public boolean isEquivalent() {
45      return isReflexive() && isSymmetric() && isTransitive();
46    }
47
48    public void reflexiveShell() {
49      for (T item : set) {
50        Pair<T, T> search = new Pair<T, T>(item, item);
51        if (!containsPair(search)) {
52          addRelation(search);
53        }
54      }
55    }
56
57    public void symmetricShell() {
58      for (Pair<T, T> pair : relation) {
59        Pair<T, T> search = new Pair<T, T>(pair.getSecond(), pair.getFirst());
60        if (!containsPair(search)) {
61          addRelation(search);
62        }
63      }
64    }
65
66    public void transitiveShell() {
67      for (Pair<Pair<T, T>, Pair<T, T>> candiate : getTransitiveCandidates())
          {
68        Pair<T, T> search = new Pair<T, T>(candiate.getFirst().getFirst(),
            candiate.getSecond().getSecond());
69        if (!containsPair(search)) {
70          addRelation(search);
71        }
72      }
73    }
74
75    public String toString() {
76      String output = "  ";
77      for (T item : set) {
78        output += " " + item;
79      }
80      output += "\n";
81      for (T item : set) {
82        output += item + " ";
83        for (T item2 : set) {
84          Pair<T, T> search = new Pair<>(item, item2);
85          output += containsPair(search) ? " x" : "  ";
86        }
```

```
87        output += "\n";
88      }
89      return output;
90    }
91
92    private ArrayList<Pair<Pair<T, T>, Pair<T, T>>> getTransitiveCandidates()
          {
93      ArrayList<Pair<Pair<T, T>, Pair<T, T>>> candiates = new ArrayList<>();
94      for (Pair<T, T> pair : relation) {
95        for (Pair<T, T> pair2 : relation) {
96          if (pair.getSecond().equals(pair2.getFirst()))
97            candiates.add(new Pair<Pair<T, T>, Pair<T, T>>(pair, pair2));
98        }
99      }
100     return candiates;
101   }
102
103   private boolean containsPair(Pair<T, T> search) {
104     for (Pair<T, T> pair : relation) {
105       if (pair.equals(search))
106         return true;
107     }
108     return false;
109   }
110 }
```

App.java

```
1  public class App {
2    public static void main(String[] args) throws Exception {
3      Relation<Integer> r = new Relation<>();
4      r.add(1);
5      r.add(2);
6      r.add(3);
7      r.add(4);
8
9      r.addRelation(new Pair<>(1, 2));
10     r.addRelation(new Pair<>(1, 1));
11     r.addRelation(new Pair<>(2, 2));
12     r.addRelation(new Pair<>(3, 3));
13     r.addRelation(new Pair<>(4, 4));
14     r.addRelation(new Pair<>(2, 4));
15     r.addRelation(new Pair<>(1, 4));
16
17     System.out.println("Ist reflexiv: " + (r.isReflexive() ? "true" : "false
           "));
18     System.out.println("Ist symmetrisch: " + (r.isSymmetric() ? "true" : "
           false"));
19     System.out.println("Ist transitiv: " + (r.isTransitive() ? "true" : "
           false"));
20     System.out.println("Ist Aequivalenzrelation: " + (r.isEquivalent() ? "
           true" : "false"));
21     System.out.println("Matrixdarstellung:\n" + r.toString());
22   }
23 }
```