Autoren: Marius Birk
Pieter Vogt
Tutor: Florian Brandt

Abgabe: 19.05.2020, 12:00 Uhr

Smileys:

| A1 | A2 | A3 | A4 | $\sum$ |
|---|---|---|---|---|
| | | | | |

# Objektorientierte Modellierung und Programmierung

# Abgabe Uebungsblatt Nr.04

(Alle allgemeinen Definitionen aus der Vorlesung haben in diesem Dokument bestand, es sei den sie erhalten eine explizit andere Definition.)

## Aufgabe 1

```
1       import java.lang.Math;
2 //Dieser Code läuft für das explizite Beispiel aus der
     Aufgabe.
3 public class Functions {
4     public static void main(String[]  args){
5         double i = 2;
6         Function chain = new SineFunction(new SquareFunction
              ());
7         chain.calculate(i);
8     }
9
10 }
11 interface Function{
12     void calculate(double i);
13     double  result=0;
14 }
15 class SineFunction implements Function{
16     private String function =" ";
17     public SineFunction(SquareFunction squareFunction) {
18         if(squareFunction.getClass()== SquareFunction.class){
19             function="square";
20         }
21         else{
22             function="sine";
23         }
24         //Diese If Abfragen können bei beliebiger Reihenfolge
               der Funktionen entsprechend erweitert werden.
25         //Weitere Anpassungen am Code sind dann aber noch nö
              tig.
26     }
27     @Override public void calculate(double i ){
28         if(function.equals("sqaure")){
29             i=i*i;
30         }
31         i=Math.sin(i);
```

```
32          System.out.println(i);
33      }
34  }
35  class SquareFunction implements Function{
36      @Override
37      public void calculate(double i) {
38
39      }
40  }
```

# 1 Aufgabe 2

## Teilaufgabe a

```
1     interface Sequence{
2        public int current=0;
3    }
```

## Teilaufgabe b

```
1           public class Naturals implements Sequence {
2              int current = 0;
3
4              @Override
5              public int getNext() {
6                 current++;
7                 return current;
8              }
9           }
```

## Teilaufgabe c

```
1           abstract class Filter implements Sequence1{
2              public Filter(Sequence1 sequence) {
3                  Object s = sequence;
4              }
5          }
```

## Teilaufgabe d

```
1           class ZapMultiples extends Filter{
2              private int cur=0;
3              private int basic;
4              public ZapMultiples(int base$,$ Sequence1
                  sequence) {
```

```
5                   super(sequence);
6                   basic = base;
7               }
8
9               @Override public int current(){
10                  return cur;
11              }
12              @Override public int getNext(){
13                  cur++;
14                  if(cur%basic ==0){
15                      cur=cur+1;
16                      return cur;
17                  }else{
18                      return cur;
19                  }
20              }
21          }
```

## Teilaufgabe e

```
1           import java.util.ArrayList;
2
3   public class Primes implements Sequence {
4
5       Sequence sequence = new Naturals();
6       int next = 1;
7       ArrayList<Integer> primes = new ArrayList<>();
8
9       private void incNext() {
10          next++;
11      }
12
13      @Override
14      public int getNext() {
15          if (checkIfPrime()) {
16              primes.add(next);
17              incNext();
18              return sequence.getNext();
19          } else {
20              incNext();
21              sequence.getNext();
22              return getNext();
23          }
24          /*if (checkIfPrime(next) && next != 1) {
25              incNext();
26              return sequence.getNext();
27          } else {
28              incNext();
```

```
29            sequence.getNext();
30            return getNext();
31         }*/
32      }
33
34      private boolean checkIfPrime() {
35         if (next == 1) {
36            return false;
37         }
38         if (isDividableByPrim(next)) {
39            return false;
40         } else return true;
41         /*
42         for (int i = 2; i < number; i++) {
43            if (next % i == 0) {
44               return false;
45            }
46         }
47         return true;
48         */
49      }
50
51      private boolean isDividableByPrim(int next) {
52         for (Integer i : primes) {
53            if (next % i == 0) {
54               return true;
55            }
56         }
57         return false;
58      }
59
60      public Primes() {
61      }
62   }
```

# 2 Aufgabe 3

```
1  import jdk.jshell.execution.Util;
2
3  import javax.print.DocFlavor;
4
5  public class Compare {
6      public static void main(String[] args){
7          Object[] objects = new Object[3];
8          Comparable one = new ComparableInteger(1);
9          Comparable four = new ComparableInteger(4);
10         Comparable seven = new ComparableInteger(7);
11
```

```
12          objects[0] = one;
13          objects[1] = four;
14          objects[2] = seven;
15
16          Utils util = new Utils();
17          Comparable getMinimum;
18      }
19  }
20  interface Comparable{
21      public int compareTo(Comparable obj);
22      public int getValue();
23      public void setValue(int i);
24  }
25
26  class Utils{
27      public static Comparable getMinimum(Comparable[] elements
            ){
28          ComparableInteger min= new ComparableInteger(0);
29          for(int i =0; i<elements.length;i++){
30
31              if (min.getValue() > elements[i].getValue()) {
32                  min.setValue(i);
33              }
34          }
35          return elements[min.getValue()];
36      }
37
38  }
39  class Integer{
40      protected int value;
41      public Integer(int value){
42          this.value=value;
43      }
44      public int getValue(){
45          return value;
46      }
47  }
48  class ComparableInteger implements Comparable{
49      protected int value;
50      public ComparableInteger(int value){
51          this.value=value;
52      }
53      public int getValue(){
54          return value;
55      }
56
57      @Override public void setValue(int value) {
58          this.value = value;
59      }
```

```
60
61      @Override
62      public int compareTo(Comparable obj) {
63          return 0;
64      }
65  }
```

# 3 Aufgabe 4

```
1     import java.util.ArrayList;
2
3   public class Rooms {
4       public static void main(String[] args){
5           Furniture chair = new Chair();
6           Desk desk = new Desk();
7           Chair chair1 = new Chair();
8
9           Office office = new Office(desk, chair1, chair);
10      }
11  }
12  interface Furniture{
13
14  }
15  class Table implements Furniture{
16      private static int legs = 4;
17      public int getLegs(){
18          return legs;
19      }
20  }
21  class Desk extends Table{
22
23  }
24  class Chair implements Furniture{
25      private static int legs = 4;
26      public int getLegs(){
27          return legs;
28      }
29  }
30  class Office{
31      private int i=0;
32      ArrayList<Chair> chairs = new ArrayList<Chair>();
33      ArrayList<Desk> desks = new ArrayList<Desk>();
34      ArrayList<Furniture> furniture = new ArrayList<Furniture
           >();
35
36      public Office( Desk desk, Chair chair, Furniture fur) {
37          desks.add(desk);
38          chairs.add(chair);
```

```
39          furniture.add(fur);
40      }
41  }
42
43  class Room extends Office{
44      public Room(Desk desk, Chair chair, Furniture fur) {
45          super(desk, chair, fur);
46      }
47  }
```