Code aus der ersten Übungsstunde

Beispiel 1 / Ergänzungen zum Assignment 1

```
public class Beispiel1 {
//Array Index Fehler und Input Fehler auffangen, Funktion von
finally.

     public static void main(String[] args) {
          try {
               for (int i = 0; i <= args.length; i++) {

     System.out.println(Integer.parseInt(args[i]) + 1);
               }
          } catch (ArrayIndexOutOfBoundsException e) {
               System.err.println("Out of Bounds: " +
e.getMessage());
          } catch (NumberFormatException e) {
               System.out.println("You have to enter Integers
as arguments. "
                         + e.getMessage());
          } finally {
               System.out.println("finally wird ausgeführt.");
          }
     }
}
```

Beispiel 2 / try und catch

```
class Example2 {
     public static void main(String[] args) {
          int i;
          int tmp;
          /* iterate through the argument vector */
          for (i = 0; i < args.length; i++) {
               try {
                    tmp = Integer.parseInt(args[i]);
                    if (tmp < 0) {
                         throw(new MyException("less than
zero"));
                    }
                    System.out.println(tmp);
               } catch (MyException e) {
                    System.out.println("Exception caught ...");
                    System.out.println(e.getMessage());
                 }
          }
          System.out.println("done.");
     }
}
```

```
class MyException extends Exception {
    MyException(String text) {
     super(text);
    }
}
```

Beispiel 3 / try und catch (nested)

```
class Example5 {

       public static void main(String[] args) {

         int i;
         int tmp;
         int count = 0;

          try {
              /* iterate through the argument vector */
              for (i = 0; i < args.length; i++) {
               try {
                        tmp = Integer.parseInt(args[i]);
                        if (tmp < 0) {
                         throw(new MyException1("less than
zero"));
                        }
                        System.out.println(tmp);
                    } catch (MyException1 e) {
                        System.out.println("Exception1 caught
...");
                        System.out.println(e.getMessage());
                        count++;
                        if (count >= 3) {
                         throw(new MyException2 ());
                        };
                    }
                    finally {
                        System.out.println("Processed one
integer");
                    }
                 }
            }
                 catch (MyException2 e) {
                     System.out.println("Exception2 caught");
                     System.out.println("Enough.");
                 }
                 finally {
                     System.out.println("done.");
                 }
      }
}
```

```
class MyException1 extends Exception {
     MyException1(String text) {
     super(text);
     }
}

public class MyException2 extends Exception{}
```

VirtualCafe-Beispiel

```
class BadTasteException extends Exception {
     //constructor
     public BadTasteException(){
     }
}

class TemperatureException extends Exception{
     private int temperature;

     //constructors
     public TemperatureException(){
     }

     //getter
     public int getTemperature() {
          return temperature;
     }
}

public class TooColdException extends TemperatureException {
     //constructors
     public TooColdException(){
     }
}

public class TooHotException extends TemperatureException{
     //constructors
     public TooHotException(){
     }
}

public class VirtualCoffee {

     public static void main(String[] args) throws
TemperatureException{
          //create a new coffee cup
          Coffee cup = new Coffee();
          //create and serve to a customer
          try {
               Customer customer = new Customer();
               Cafe.serveCustomer(customer, cup);
          } catch (BadTasteException e) {
               System.out.println("This coffee tastes bad.");
          } finally {
               System.out.println("I'm sitting in a cafe
drinking coffee.");
          }
     }
}
```

```
public class Coffee {
     private int temperature = 75;
     //constructor
     public Coffee(){
     }

     //setters and getters
     public void setTemperature(int temperature) {
          this.temperature = temperature;
     }

     public int getTemperature() {
          return temperature;
     }
}

public class Cafe {
     public static void serveCustomer(Customer cust, Coffee
cup)throws
     TemperatureException, BadTasteException{
          try{
               cust.drinkCoffee(cup);
               System.out.println("Coffee is great!");
          } catch (TooColdException e) {
               System.out.println("This coffee is too cold.");
          }
     }
}

public class Customer {
     public void drinkCoffee(Coffee cup) throws
TooColdException, TemperatureException, BadTasteException{
          try{
               int i = (int)(Math.random()*5.0);
               System.out.println(i);
               switch(i){
               case 0:
                    throw new TooHotException();
               case 1:
                    throw new TooColdException();
               case 2:
                    throw new BadTasteException();
               case 3:
                    throw new TemperatureException();
               default:
                    System.out.println("everything is o.k.");
               }
          } catch (TooHotException e){
               System.out.println("This coffee is too hot.");
          }
     }
}
```