Übung zur VL „Grundlagen der Programmierung“

10. Übung

Dr. Zubow

Quine

- Ein **Quine** ist ein Computerprogramm, welches eine Kopie seiner selbst (üblicherweise seines Quelltextes) als Ausgabe schreibt. Es handelt sich somit um eine Form der Selbstbezüglichkeit.
- Schreiben Sie ein Quine in Java.

```
public class Quine {
  public static void main(String[] args) {
    char c = 34;
    System.out.println(s + c + s + c + ';' + '}');
  }

  static String s = "public class Quine { public static void main(String[] args) { char c=34; System.out.println(s+c+s+c+';'+'}'); } static String s=";
}
```

Welche Ausgabe erzeugt dieses Programm?

```
class X {
   int a = 4;
   int get() { return a; }
}
class Y extends X {
   static int a = 7;
   int get() { return a; }
   static void set(int x) { a = x; }
   static void set(char c) { a = 2 * c; }
}
public class Z extends Y {
   static int b = 3;
   int get() { return b + a; }
   static int get(X x) { return x.a; }
   static void set(int i) { a = 3 * i; }
   static void set(X x, int i) { a = i; }
   public static void main(String[] a) {
      test();
   }
}
```

```
static void test() {
      Z z = new Z();
      System.out.println(z.a);
      System.out.println(get(z));
      System.out.println(((X) z).get());
      z.set('c' - 'a' - 1);
      System.out.println(get(z));
      System.out.println(z.get());
      Y y = z;
      y.set(2);
      System.out.println(z.get());
      z.set(y, 0);
      System.out.println(y.get());
   }
```

Exceptions

```
class TestException extends Exception {
  TestException() {
    super();
  }

  TestException(String s) {
    super(s);
  }
}
```

Ausführung:
>java Test divide null test foo

```
public class Test {
  public static void main(String[] args) {
    for (int i = 0; i < args.length; i++) {
      try {
        thrower(args[i]);
        System.out.println("Test \"" + args[i] + "\" didn't throw an exception");
      } catch (Exception e) {
        System.out.println("Test \"" + args[i] + "\" threw a " + e.getClass()
            + "\n         with message: " + e.getMessage());
      }
    }
  }

  static int thrower(String s) throws TestException {
    try {
      if (s.equals("divide")) {
        int i = 0;
        return i / i;
      }
      if (s.equals("null")) {
        s = null;
        return s.length();
      }
      if (s.equals("test"))
        throw new TestException("Test message");
      return 0;
    } finally {
      System.out.println("[thrower(\"" + s + "\") done]");
    }
  }
}
```

Threads - ATM-Beispiel