252-0027-00: Einführung in die Programmierung

Invarianten (G-09)

Bei allen folgenden Aufgaben sind die (Preconditions und) Postconditions gegeben. Finden Sie jeweils die Loop Invariante. Teilweise finden Sie weitere Bemerkungen zu Methoden in der Aufgabenstellung.

Aufgabe 1: Loop Invariante

```
public int compute(int n) {
  // Precondition:  n >= 0
  int x = 1;
  int factorial = 1;

  // Loop Invariante:

  while (x <= n) {
    factorial = factorial * x;
    x = x + 1;
  }

  // Postcondition:  factorial == n!
  return factorial;
}
```

Aufgabe 2: Loop Invariante

```
public int compute(int base, int exponent) {
  // Precondition:  exponent >= 0
  int x = 0;
  int res = 1;

  // Loop Invariante:

  while (x < exponent) {
    res = res * base;
    x = x + 1;
  }

  // Postcondition:  res == Math.pow(base, exponent)
  return res;
}
```

Aufgabe 3: Loop Invariante

Die Methode `average(int[] arr)` gibt den Durchschnitt der Elemente des Arrays `arr` zurück. Sie dürfen zusätzlich eine Methode `subarray(int i, int j)` auf ein Array `a` aufrufen, welches das Subarray von `a` von Index `i` (inklusive) bis `j` (exklusive) zurückgeben soll.

```
public int compute(int[] n) {
  // Precondition: n != null && n.length > 0
  int x = 0;
  double sum = 0;

  // Loop Invariante:

  while (x < n.length) {
    sum = sum + n[x];
    x = x + 1;
  }

  // Postcondition:  sum == average(n) * n.length
  return sum/n.length;
}
```

Aufgabe 4: Loop Invariante

Die Methode `fibonacci(int n)` gibt die `n`-te Fibonaccizahl zurück, wobei `fibonacci(0) = 0` und `fibonacci(1) = 1`.

```
public int compute(int n) {
  // Precondition:  n >= 1
  int a = 0; int b = 1; int x = 2;

  // Loop Invariante:

  while (x <= n) {
    int temp = b;
    b = a + b;
    a = temp;
    x = x + 1;
  }

  // Postcondition:  b == fibonacci(n)
  return res;
}
```

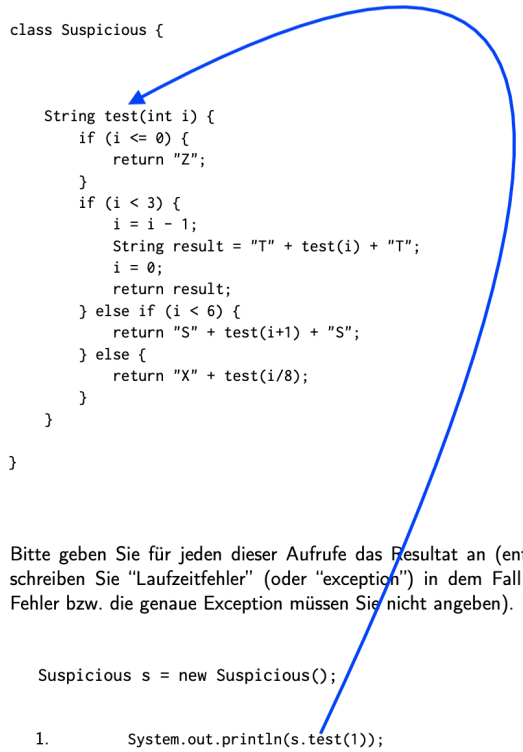
Aufgabe 5: Loop Invariante

Die Methode `maximum(int[] arr)` gibt den grössten Wert im Array `arr` zurück. Sie dürfen zusätzlich eine Methode `subarray(int i,int j)` auf ein Array `a` aufrufen, welches das Subarray von `a` von Index `i` (inklusive) bis `j` (exklusive) zurückgeben soll.

```
public int compute(int[] n) {
// Precondition: n != null && n.length > 0
  int max = Integer.MIN_VALUE;
  int x = 0;

  // Loop Invariante:

  while (x < n.length) {
    if (n[x] > max) {
      max = n[x];
    }
    x = x + 1;
  }

  // Postcondition: max = maximum(n)
  return max;
}
```

Aufgabe 6: Ausgabe

Gegeben sei eine Java Methode `test()`, die mit verschiedenen Argumenten für ein Exemplar `s` der Klasse `Suspicious` aufgerufen wird.



```
class Suspicious {

    String test(int i) {
        if (i <= 0) {
            return "Z";
        }
        if (i < 3) {
            i = i - 1;
            String result = "T" + test(i) + "T";
            i = 0;
            return result;
        } else if (i < 6) {
            return "S" + test(i+1) + "S";
        } else {
            return "X" + test(i/8);
        }
    }

}
```

Bitte geben Sie für jeden dieser Aufrufe das Resultat an (entweder dem String, der gedruckt wird, oder schreiben Sie "Laufzeitfehler" (oder "exception") in dem Fall dass eine Exception auftritt (den genauen Fehler bzw. die genaue Exception müssen Sie nicht angeben).

```
Suspicious s = new Suspicious();
```

1. `System.out.println(s.test(1));`

   _____

2. `System.out.println(s.test(3));`

   _____

3. `System.out.println(s.test(9));`

   _____