Selbsttest 5

1. Umwandlung von Schleifen

Geben Sie die Ausgabe bei der Ausführung der folgenden Code-Stücke an. Geben Sie eine Assertion nach der Schleife an und wandeln Sie anschließend die Schleife in eine do-while- und in eine for-Schleife um, die dieselben Werte berechnen.

```
public class Selbsttest5 extends basic {

  public static void Aufgabe1a () {
    int i, j, k;

    i = 0;
    j = 10;
    k = 0;
    while (i++ != j) {
      k = i + j;
      if (i % 3 == 0)
        j--;
      out.writeln("i: " + i + ", j: " + j + ", k: " + k);
    }
    // ???
    out.writeln("i: " + i + ", j: " + j + ", k: " + k);
  }

  public static void Aufgabe1b () {
    int i, j, k;

    i = 0;
    j = 10;
    k = 0;
    if (i++ != j) {
      do {
        k = i + j;
        if (i % 3 == 0)
          j--;
        out.writeln("i: " + i + ", j: " + j + ", k: " + k);
      } while (i++ != j);
    }
    // ???
    out.writeln("i: " + i + ", j: " + j + ", k: " + k);
  }
```

```
    public static void Aufgabe1c () {
      int i, j, k;

      i = 0;
      j = 10;
      k = 0;
      for(; i++ != j;) {
        k = i + j;
        if (i % 3 == 0)
          j--;
        out.writeln("i: " + i + ", j: " + j + ", k: " + k);
      }
      // ???
      out.writeln("i: " + i + ", j: " + j + ", k: " + k);
    }

    public static void main (String[] arg) {
      Aufgabe1a();
      out.writeln();
      Aufgabe1b();
      out.writeln();
      Aufgabe1c();

      in.readln();
    }
  }

  Output:
  i: 1, j: 10, k: 11
  i: 2, j: 10, k: 12
  i: 3, j: 9, k: 13
  i: 4, j: 9, k: 13
  i: 5, j: 9, k: 14
  i: 6, j: 8, k: 15
  i: 7, j: 8, k: 15
  i: 8, j: 8, k: 16
  i: 9, j: 8, k: 16
```

2. Wissensfragen – Parameter

2.1 Formale Parameter stehen _____ (im Methodenkopf / beim Aufruf) .
2.2 Formale Parameter können _____ (nicht / schon) Ausdrücke sein.
2.3 Formale Parameter müssen an aktuelle Parameter _____ (nicht / schon) zuweisbar sein.
2.4 Aktuelle Parameter werden _____ (nach / vor) dem Aufruf ausgewertet.

3. Wissensfragen – lokale und globale Variablen

3.1 Lokale Variablen werden _____ (nach dem Methodenaufruf / am Programmende) wieder freigegeben.
3.2 Globale Variablen stehen _____ (nach / vor / während) einem Methodenaufruf zur Verfügung (wenn man mögliche Überdeckung unbeachtet lässt).
3.3 Lokale Variablen stehen _____ (nach / vor / während) eines Methodenaufrufs zur Verfügung.
3.4 Auf lokale Variablen der Methode X kann die Methode Y nur zugreifen, wenn die Methode X gerade aktiv ist. (richtig / falsch)

4. Wissensfragen – Definitionen

4.1 Der Sichtbarkeitsbereich einer Variable ist das Programmstück, in dem auf diese Variable zugegriffen werden kann.
4.2 Der Gültigkeitsbereich einer Variable ist gleich dem Sichtbarkeitsbereich.
4.3 Die Lebensdauer einer Variable ist die Zeitspanne zwischen Anlegen des Speichers und Freigabe des Speichers für die Variable.
4.4 Ein Name überdeckt einen anderen, wenn derselben Namen im Sichtbarkeitsbereich des ursprünglichen Namens nochmals deklariert wird.