Selbsteinstufungstest
Vorkurs Programmieren

Zeit 90 Minuten

Aufgabe 1: Einzigartig (10 Punkte)

Schreiben Sie eine Methode die angibt, ob ein String **`str1`** in einem zweiten String **`str2`** genau einmal vorkommt. Beim Vergleich wird Gross- und Kleinschreibung nicht unterschieden, wie das zweite Beispiel unten zeigt. Die beiden Strings **`str1`** und **`str2`** werden der Methode als Argumente übergeben, das Resultat (wahr oder falsch) gibt die Methode an den Aufrufer zurück. Überlappende Vorkommen von **`str1`** in **`str2`** werden nur einmal gezählt, wie die letzten beiden Beispiele zeigen.

Beispiele:

str1	str2	**Resultat**
"ei"	**"hallo"**	**false**
"ei"	**"Eine"**	**true**
"ei"	**"eiei"**	**false**
"ee"	**"eee"**	**true**
"ee"	**"eeee"**	**false**

Lösung:

```
public boolean kommtGenauEinmalVor(String str1, String str2){
  int idx;

  str1 = str1.toLowerCase();
  str2 = str2.toLowerCase();
  idx = str2.indexOf(str1);
  if (idx >= 0) {
    idx = str2.indexOf(str1,idx+str1.length());
    return idx == -1;
  }
  else{
    return false;
  }
}
```

Aufgabe 2: Ablaufkonstrukte (10 Punkte)

Finden Sie heraus, was das untenstehende Programm macht. Geben Sie dazu an, was das Programm auf die Konsole ausgibt. (5 Punkte)

Vereinfachen Sie dann den Inhalt der main-Methode soweit möglich, ohne dass sich die Ausgabe ändert. **Hinweis:** Die vereinfachte main-Methode sollte <15 Zeilen Code enthalten.

```
public class Ausgabe {
  public static void main(String[] args) {
    int col = 0;
    String color = "";
    String[] rows = { "", "", "" };
        while (col < 3){
      col++;
      for (int row = 1; row <= rows.length; row++){
        if ((row == 1 || row == 3) && (col != 2)){
          color = "blue ";
        }
        else{
          if (row == col){
            color = "blue ";
          }
          else{
            color = "red  ";
          }
        }
        rows[row-1] += color;
      }
    }
    for (int row = 0; row < rows.length; row++){
      System.out.println(rows[row]);
    }
  }
}
```

Lösung:

```
blue red  blue
red  blue red
blue red  blue
```

Vereinfachung:

```
public class Ausgabe {
    public static void main(String[] args) {
            for (int row = 0; row < 3; row++){
                  for( int col = 0; col<3; col++){
                        if ((row+col)%2 == 0){
                              System.out.print("blue ");
                        }
                        else {
                              System.out.print("red  ");
                        }
                  }
                  System.out.println();
            }
    }
}
```

Aufgabe 3: Primfaktor (15 Punkte)

1. (10 Punkte) Schreiben Sie zuerst eine Methode **`istPrimzahl`**, die für eine natürliche Zahl **`n`** (d.h. **`n > 0`**) feststellt, ob sie eine Primzahl ist. Die Zahl **`n`** ist dann eine Primzahl, wenn sie >1 und nur durch sich selbst und durch 1 teilbar ist (1 selbst zählt nicht zu den Primzahlen). Die Methode erhält die Zahl **`n`** als Argument und soll ihre Antwort mit einem geeigneten Rückgabewert zurückgeben.

   Hinweis:
   Um zu prüfen, ob eine Zahl **`n`** eine Primzahl ist, probieren Sie einfach alle Zahlen **`2 . . .`** $\sqrt{}$**`n`** als Teiler von **`n`** durch und schauen Sie, ob die Division einen Rest ergibt oder nicht.

2. (5 Punkte) Schreiben Sie aufbauend auf der Methode **`istPrimzahl`** die Methode **`istPrimfaktor`**, die feststellt, ob eine Zahl **`p`** Primfaktor einer anderen Zahl **`n`** ist. Die Zahl **`p`** ist Primfaktor der Zahl **`n`**, falls **`n`** durch **`p`** teilbar und **`p`** eine Primzahl ist. Die Zahlen **`p`** und **`n`** werden der Methode als Argumente übergeben, das Resultat gibt die Methode als Rückgabewert zurück.

Lösung:

```
Aufgabe 1):
  public boolean istPrimzahl(int n){
    int teiler = 2;
    int rootN = (int)(Math.sqrt(n)+0.5);  // runden

    if (n <= 1) {
      return false;
    }
    while(n % teiler != 0 && teiler <= rootN){
      teiler++;
    }
    return !(teiler <= rootN);
  }

Aufgabe 2)
  public boolean istPrimfaktor(int n, int p){
    return (n%p == 0) && istPrimzahl(p);
  }
```

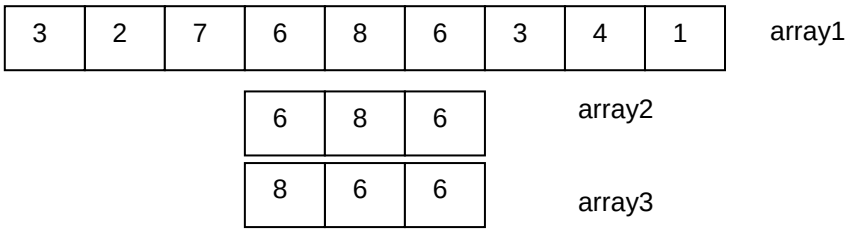
Aufgabe 4 : Eindimensionale Arrays (15 Punkte)

Schreiben Sie eine Methode

`private boolean contains(int[] arr1, int[] arr2)`

die prüft, ob der Array **`arr2`** in **`arr1`** enthalten ist. Die Methode soll für beliebige Array-Argumente ein korrektes Resultat zurückgeben.

Beispiel:



`array2` ist in **`array1`** enthalten, **`array3`** hingegen nicht.

Lösung:

```
  private boolean contains(int[] arr1, int[] arr2) {
    if (arr1 == null || arr2 == null) return false;
    if (arr2.length > arr1.length) return false;
    boolean contains = false;
    for (int i = 0; i <= arr1.length - arr2.length; i++){
      contains = true; // if arr2.length = 0, contains = true
      for (int j = 0; j < arr2.length && contains; j++) {
        contains = arr1[i + j] == arr2[j];
      }
      if (contains) return contains;
    }
    return contains;
 }
```

Aufgabe 5: Bezeichnungen (10 Punkte)

In der untenstehenden Tabelle stehen in der 1. Spalte die Namen von verschiedenen Java-Konstrukten und in den restlichen Spalten die dazugehörigen Codebeispiele. Streichen Sie alle Codebeispiele, die nicht zum Java-Konstrukt in der ersten Spalte gehören können.

Beispiel **Konstrukt**	a	B	c	d	e
Datentyp	`3.5` (X)	`real` (X)	`int`	`float`	`type` (X)
Argumentliste	`(int x, z)` (X)	`(a+b,c/d)` (X)	`()`	`paint` (X)	`args()` (X)
Kommentar	`\\` (X)	`{ xxx }` (X)	`//* xx`	`/* xxx */`	`"xxx"` (X)
Variablen-deklaration	`(int b, c)` (X)	`z = 3*a;` (X)	`int x= y;`	`x++;` (X)	`int (x,y)` (X)
Objekt	`d40`	`Graphics` (X)	`2.f` (X)	`30` (X)	`int x` (X)
Zuweisung	`int v = 10;`	`x++;`	`x = y;`	`x();` (X)	`return;` (X)
Operator	`/*` (X)	`(float)`	`;` (X)	`**` (X)	`-`
Double-Zahl	`1d`	`2E5D`	`(double)` (X)	`5.`	`d2` (X)
Methodenaufruf	`v.w();`	`x = 3;` (X)	`(int);` (X)	`(x,y);` (X)	`run();`
Anweisung	`extends` (X)	`app.init();`	`draw();`	`x= -y;`	`(a + b)` (X)

Aufgabe 6 Kurzaufgaben (14 Punkte)

In den folgenden zwei Teilaufgaben ist jeweils ein lauffähiges Java-Programm (ohne Import-Anweisungen) angegeben. Geben Sie zu jedem Programm an, welches die Ausgabe auf dem Bildschirm ist, wenn es ausgeführt wird.

Teilaufgabe a (Arithmetische Ausdrücke, 7 Punkte)

```
public class Operatoren {
  public static void main(String[] args){
    int i1 = 4, i2 = 5;
    long l1 = 10;
    float f1 = 4.0f, f2 = 6f;
    double d1 = 5.0;
    System.out.println(l1*i1 + (int)f1/(int)f2);
    System.out.println(l1 % i1 + f2 % (float)d1 );
    System.out.println(f2/i1 + i2%i1 );
    System.out.println(i1 - (int)(f1/f2) );
    System.out.println(1/i1*2f*l1 );
    System.out.println(l1 - 3/2.0 );
    System.out.println(Math.sqrt(1/2));
  }
}
```

Lösung:

1	40
2	3.0f
3	2.5f
4	4
5	0.0f
6	8.5d
7	0.0d

Teilaufgabe b (Logische Ausdrücke, 7 Punkte)

```
public class LogischeAusdruecke {
  public static void main(String[] args){
    boolean ok = false;
    int y = 20;
    for (int a = 0; a < 10; a++){
      if ((a > 1) && (a <4)){
        ok = false;
      }
      ok = ok && !(a % 3 == 0);
      if (ok || (a % 4 != 0)){
        ok = !ok;
      }
      if (ok) {
        System.out.println("a= " + a);
      }
    }
  }
}
```

Lösung:

1	a= 1
2	a= 2
3	a= 3
4	a= 5
5	a= 6
6	a= 9

Aufgabe 7: Methode mit Arrays (15 Punkte)

Schreiben Sie zwei Methoden, denen jeweils ein beliebiger **`double`**-Array übergeben wird. Die Methoden sollen sodann die Reihenfolge der Elemente im übergebenen Array umdrehen. Wird den Methoden z.B. ein Array `a = [1,2,3,4]` übergeben, so geben die Methoden den Array `[4,3,2,1]` zurück. Die zwei Methoden unterscheiden sich wie folgt:

- Die Methode **`changeOrder`** erzeugt einen neuen Array und kopiert dann die Elemente des übergebenen Arrays in der gewünschten Reihenfolge in den neuen Array. Schliesslich gibt die Methode den neuen Array zurück.
- Die Methode changeOrder2 erzeugt keinen neuen Array, sondern kopiert die Elemente direkt im übergebenen Array um.

Aufgaben

1. Schreiben Sie eine Methode **`changeOrder`** gemäss obiger Beschreibung. (8 Punkte)
2. Schreiben Sie die Methode **`changeOrder2`** gemäss obiger Beschreibung. (5 Punkte)
3. Schreiben Sie für jede der obigen Methoden einen Methodenaufruf, um die Reihenfolge der Elemente im obigen Array **`a = [1,2,3,4]`** umzudrehen,
   so dass danach **`a = [4,3,2,1].`** (2 Punkte)

Lösung:

```
1.  public double[] changeOrder(double[] arr){          //2P
      double[] arrR = new double[arr.length];           //2P

      for (int i=0; i<arr.length; i++){                 //2P
        arrR[arr.length-i-1] = arr[i];                  //1P
      }
      return arrR;                                      //1P
    }

2.  public void changeOrder2(double[] arr){             //1P
      double tmp;

      for (int i=0; i<arr.length/2; i++){               //1P
        tmp = arr[arr.length-i-1];                      //1P
        arr[arr.length-i-1] = arr[i];                   //1P
        arr[i] = tmp;                                   //1P
      }
    }

3.  double[] a = {1.,2.,3.,4.};
    a = changeOrder(a);                                 //1P
    changeOrder2(a);                                    //1P
```