

Prog Fragen

1) Welche Anweisung funktioniert ohne Compiler Fehler?

- `int i = 42; long l = i`
- `Byte b = 2; long l = b`
- `Char c = 'c'; int i = c`
- `Int i = 42; doubled d = I`
- `Int i = 13`
- `Int i = 42_001`
- `Char c = '1'`
- `Double d = 12.1D`

2) Was muss an der Stelle ? Stehen, damit man den String „ENGEL erhält?
„BENGEL“.substring(?)

- `Substring(1)`

3) Welche Ausgabe erzeugt `System.out.printf(„Gewicht %2f %s“, 24.788, „Gramm“)`?

- Gewicht 24,79 Gramm

4) Was ist äquivalent zu `x = x > 17 ? 2 : -1`

- `if (x > 17) { x = 2; } else { x = -1; }`

5) Was Stimmt? (Frage zu if-Anweisung; keine Schleifen!!!)

- `(1==1 ergibt true)`

6) Arrays was stimmt?

- Ein Array speichert verschiedene Werte, die alle vom gleichen Typ sein müssen

- `Int [] numbers = new int[5]` erzeugt einen Array vom Typ `int` mit Platz für 5 Elemente

- `String [] appointments = {„Essen“, „Schlafen“}; appointments [0] = „Essen“;` erzeugt keinen Fehler

- `String [] appointments = {„Essen“, „Schlafen“};`

- `appointments.length = 2`

7) `int[] numbers = new int[5];` steht bereits im Code. Welche der folgenden Anweisungen erzeugt _keinen_ Fehler?

- numbers[0] = 5

- numbers[4] = 5

8) int[] ValuesA = {1,2,3}; int[] valuesB = new int [2]; System.arraycopy(valuesA , 1 , valuesB, 0 ,2); Was steht in ValuesB

- {2, 3}

9) int[] valuesA={1,2}; int[] valuesB = {1,2}; Was ergibt true?

- valuesA.length == valuesB.length

- valuesA[0] == valuesB[0]

- valuesA[0] == valuesB[0] && valuesA[1] == valuesB[1]

10) String[][] Kalender = new String [31][24]; Was stimmt?

- calender ist ein mehrdimensionales Array

- calender enthält Elemente vom Typ String

- calender[0] ist ein eindimensionales Array

- calender[2] ist ein eindimensionales Array

- calender[13][25] erzeugt einen Fehler

11) Was erzeugt (int) (Math.random()*16) - 10 ?

- Zufällige Ganzzahlen im Bereich -10 bis +5

12) Was gilt für int[] numbers = {1,2,3}; for (n:numbers) {System.out.println(n); n = n*2;} ? 13 Wählen sie alle richtigen Antwortmöglichkeiten aus:

- Es handelt sich um eine for-each Schleife

- Die Zahlen 1,2,3 werden ausgegeben

-

13) Welche Aussagen sind wahr?

- Paths.get(„test.txt“) erzeugt ein Path-Objekt für die Datei test.txt

- newScanner(path, „UTF-8) kann eine IOException werfen

- scanner.close() sollte man nicht vergessen, damit Ressourcen korrekt freigegeben werden

- scanner.useDelimiter(„;“) weist den Scanner an, anhand des Semikolons zu trennen

- „abc;def“.split(„;“) liefert ein Array {„abc“, „def“}

14) Java quellcode wird kompiliert? WAHR

15) Einen Array speichert verschiedene Werte genau eines typen, zb. int-Array (typ int) Oder String array (typ String)
WAHR

16) float[] numbers = new float [3]; erzeugt ein Array für float mit platz für 3 Elemente WAHR

17) Was ist richtig?

- = ist ein Zuweisungsoperator und ein binär Operator

- (25 == 25 && true) ist wahr

- Mit x = 5 ist (++x == 6) wahr

- System.out.println(5 / (double) 2) gibt 2.5 aus

- System.out.println((double) 5 / 2) gibt 2.5 aus

18) Switch

- switch(x) arbeitet mit x = int, char, String und weiteren Typen

- Switch-Statements sind fehleranfällig, weil man gerne mal ein ‚break‘ vergisst

19) public class Animal {

public String name;

public Animal(String name) {

this.name = name;

} }

Wenn man das this. Im consturktor weglässt, macht der Code nicht mehr das gleiche wie vorher WAHR

20) Eine abstrakte Klasse kann man nicht instanziiieren. FALSCH (nur Klassen, die davon erben)

21) Wenn immer eine Zahl einen größeren Wert als 1.000.000 sollte man eine Exeption werfen FALSCH (kein logischer Grund dafür)

22) Es gibt checked und unchecked exeptions. WAHR

23) Die NullPointerException ist eine unchecked exeption. WAHR

24) Die Methode void print(int... ints); kann man mit : print (new int []{123, 32}); aufrufen WAHR(aber auch möglich mit print (4 , 65 , 1);)

25) Eine Methode, die sich selbst aufruft bezeichnet man als iterativ FALSCH (als rekursiv)

26) Die Berechnung der Fakultät einer Zahl lässt sich nur mit iteration durchführen FALSCH (auch mit Rekursion)

27) Man sagt, dass B von A erbt, wenn A extends B im code Steht.
FALSCH(„class A extends B“ , d.h die Klasse A erbt von B)

28) NullPointerException

```
String s = null;  
System.out.println (s.length());
```

erzeugt eine NullPointerException

29) Klassen A und B mit „B extends A“. Beide implementieren eine Methode „printName()“, die den jeweiligen Klassennamen ausgibt (also „A“ oder „B“).

A x = new A(); x.printName(); gibt „A“ aus.

B x = new B(); x.printName() gibt „B“ aus.

A x = new B(); x.printName() gibt „B“ aus. → A ist die Oberklasse (B erbt von A), d.h es wird erst in B (spezifische Klasse) nach einer PrintName Methode gesucht und dann in A.

30) Objekterzeugung

Konstruktoren können über this(...) andere Konstruktoren der gleichen Klasse aufrufen.

Instanzen einer Klasse werden mit dem Schlüsselwort „new“ erzeugt!