# Name:

# Quiz-Fragen zu Java:

**Hinweis: Wenn auf Klassen oder Methoden mit Namen verwiesen wird, dann sind diese im Codeanhang zu finden.**

1. Welchen Wert haben die Variablen nach Ausführung des folgenden Java-Quelltextes?

String s1 “Ein String”;

String s2 “Ein “;

String s3 = “String”;

s3 = s2.concat(s3);

boolean b1 = (s3 == s1);

boolean b2 = (s3.equals(s1));

boolean b3 = (s2.equals(s1));

s3.toLowerCase();

boolean b4 = (s3.equals(“ein string”));

b1:                                 b2:                                 b3:                                 b4:

1. Wie sieht im Rumpf eines Konstruktors mit einem oder mehr formalen Parametern der Aufruf des parameterlosen Konstruktors derselben Klasse aus?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Welchen Wert hat die Variable **x** nach der Ausführung des folgenden Java-Quelltextes?

int x = 0;

while(x > 0) {

x = x +1;

}

1. Worauf sollte in jeder rekursiven Methode, die sich selbst direkt aufruft, geachtet werden?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Wie viele lokale Variablen hat die Methode „**gibAus**“in der Klasse „**B**“?
2. Wie viele Methoden hat die Java Klasse „**B**“?
3. Wie viele aktuelle Parameter enthält die Java Methode „**output**“ in der Klasse „**B**“?
4. Welche der folgenden Aussagen gelten für die folgende Java-Methode?

public boolean istName(String anrede)

{

if(anrede.equals(\_name))

…

}

* Der Rückgabetyp der Methode ist **String**
* **anrede** ist der Name des formalen Parameters
* **String** ist der Typ des formalen Parameters
* Der Wert des formalen Parameters wird als aktueller Parameter weitergegeben

1. Welche der folgenden Aussagen sind für Java richtig?

* Ein formaler Parameter kann wie eine lokale Variable nur innerhalb der deklarierenden Methode benutzt werden.
* Die Typen und die Reihenfolge der formalen Parameter einer Java-Methode legen zusammen mit dem Namen der Methode ihre Signatur fest.
* Wenn bei einem Aufruf ein formaler Parameter keinen Wert bekommt, wird der Default-Wert gesetzt.
* Der Wert eines formalen Parameters wird beim Methodenaufruf an den entsprechenden aktuellen Parameter gebunden.
* Der Wert eines formalen Parameters kann im Rumpf der gerufenen Methode beim Aufruf einer anderen Methode als aktueller Parameter übergeben werden.
* Beim Aufruf einer Methode müssen an der Aufrufstelle die formalen Parameter angegeben werden.

1. Welchen Ergebnistyp hat die folgende Java-Methode?

public boolean m(int a){ … }

1. Welche der folgenden Aussagen sind richtig?

* In einer Grammatik in EBNF gibt es genau ein Startsymbol.
* Auf der linken Seite einer EBNF-Regel steht immer ein Nichtterminal.
* In einer Grammatik in EBNF muss es für jedes Terminal mindestens eine Regel geben, in der das Terminal auf der linken Seite steht.
* Auf der rechten Seite einer EBNF-Regel muss mindestens ein Symbol(Terminal oder Nichtterminal) stehen.
* Eine Grammatik in EBNF legt die möglichen Typen eines Programms fest.
* Es gibt syntaktisch korrekte Java-Programme, die nicht von einem Java-Compiler akzeptiert werden.

1. Welche der folgenden Aussagen sind für Java richtig?

* In einer **switch**-Anweisung kann jeder **case**-Label mit einem **break** abgeschlossen werden.
* In dem Java-Quelltextfragment **switch(x)** kann das **x** ein beliebiger int-Ausdruck sein.
* Bedingte Anweisungen (mit **if**) dürfen in Java *nicht* geschachtelt werden.
* Bei der **switch**-Anweisung dürfen unter anderem die Literale aller primitiven Typen als **case**-Label verwendet werden.
* Die **switch**-Anweisung bietet auch die Möglichkeit, mehrere Fälle auszuwählen.
* Java bietet mehrere for-Schleifen an, aber *keine* while-Schleife.

1. Welche der folgenden Aussagen sind für Java richtig?

* Eine Referenzvariable hält entweder eine gültige Referenz auf ein Objekt oder den Wert **null**
* Eine Referenzvariable darf *nicht* in einem arithmetischen Ausdruck verwendet werden.
* Jede Referenzvariable hat einen Typ.
* Mit Referenzvariablen ist *keine* Zeiger-Arithmetik zulässig.
* Zwei verschiedene Referenzvariablen, die auf dasselbe Objekt verweisen, haben verschiedene Werte.
* Der Java-Ausruck „**String a = b;**“ kopiert das String-Objekt das in b referenziert wird.

1. Bei welchen Java-Konstrukten unterscheidet man zwischen Kopf und Rump?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Codeanhang:

## Klasse „B“:

class B {

private String \_name;

public int \_jahre;

public void setName(String name)

{

\_name = name;

}

public void gibAus()

{

System.out.println(„Name: “ + output());

}

private String output()

{

return \_name + „ - “ + \_jahre;

}

}