# Name:

# Quiz-Fragen zu Java:

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

1. Welche der folgenden Aussagen sind für Java richtig?

* Ein Interface enthält *keine* privaten Methoden. **X**
* Ein Interface kann keine Konstanten deklarieren.
* Ein Interface definiert einen Referenztypen. **X**
* Ein Interface kann Exemplarvariablen deklarieren.

1. Welche der folgenden Aussagen sind richtig?

* Mit gutem Testem erhöhen wir unser Vertrauen darin, dass eine Software-Einheit ihre Aufgabe in angemessener Weise erfüllt. **X**
* Positiv-Tests sind wichtiger als Negativ-Tests. **X**
* Ein Black-Box-Test gilt als vollständig, wenn er alle Operationen des zu testenden Objektes testet.
* Beim Testen sollte auf Grenzwerte und Äquivalenzklassen in den erwarteten Ausgabewerten geachtet werden.

1. Welche der folgenden Aussagen sind üblicherweise erst zur Laufzeit eines Java-Programms beantwortbar?

* Welche Lebensdauer haben die erzeugten Objekte? **X**
* Welche Syntaxregeln gelten für einen Ausdruck?
* Ist diese Variable an dieser Stelle sichtbar?
* Wie oft wird diese Methode aufgerufen? **X**
* Wie viele Objekte werden in diesem Programm erzeugt? **X**
* Sind die vorab vereinbarten Quelltextkonventionen hier eingehalten?

1. Welche der folgenden regulären Ausdrücke (in Java-Notation als Zeichenkette angegeben) beschreiben Mengen, in denen u.a. die Zahlen von 0 bis 255 enthalten sind?

* „[0..255]“
* „[0-255]”
* „(0..255)”
* „\d+” **X**
* „.\*”**X**

1. Was liefert die Methode **addMul**?

Antwort:      **„328“(String)**

1. Welche Ausgabe erfolgt auf System.out, wenn die Variable i zu Anfang der Ausführung des folgenden Java-Quelltextes den Wert 2 hat?

switch(i) {

case 1: System.out.println(„Eins“);

case 2: System.out.println(„Zwei“);

default: System.out.println(„Viele“);

}

Antwort:        **Zwei** **(Neue Zeile) Viele**

1. Welches Ergebnis liefert die Methode **m**, wenn bei einem Aufruf als aktueller Parameter der Wert 3 übergeben wird?

* 2
* 3 **X**
* 4
* 6

1. Wie muss der Quelltext (an der mit Unterstrichen markierten Stelle) ergänzt werden, damit der Schleifenrumpf genau einmal ausgeführt wird?

int p = 9;

int result = 0;

for(\_\_\_\_\_; i<p; ++i) {

result = result + 2\*i

}

Antwort:    **int** **i = 8**

1. Wie muss der Quelltext (an der mit Unterstrichen markierten Stelle) ergänzt werden, damit die Methode bei der Übergabe des richtigen Losungswortes *true* liefert?

public boolean sesamOeffneDich(String losungswort) {

boolean result = false;

String passwort = “Abrakadabra”;

if(\_\_\_\_\_) {

result = true;

}

return result;

}

Antwort:    **passwort.equals(losungswort)**

1. Welches Ergebnis liefert die Methode **m2**, wenn bei einem Aufruf als aktueller Parameter der Wert 4 übergeben wird?

* 10
* 15
* 20 **X**
* 30

1. Welche der folgenden Aussagen zum Testen sind richtig?

* Mit Testen kann nur die Anwesenheit von Fehlern nachgewiesen werden, aber im Allgemeinen nicht deren Abwesenheit garantiert werden. **X**
* Da vollständige Tests meist nicht praktikabel sind, empfiehlt es sich Grenzwert und Äquivalenzklassen in den möglichen Eingabewerten zu ermitteln und zu verwenden. **X**
* Bei einem Positivtest dürfen nur positive Ganzzahlen als Testparameter übergeben werden.
* Mit Negativ-Tests erhöhen wir unser Vertrauen in die Korrektheit einer Software-Einheit.
* Ein Black-Box-Test ist aus der Sicht eines Klienten geschrieben, der lediglich die Schnittstelle des zu testenden Objektes kennt. **X**
* Statische haben gegenüber dynamischen Tetss den Vorteil, dass die keinen Quelltext erforden.
* Ein vollständiger Test überprüft für alle gültigen Eingabewerte, ob die korrekten Ausgabewerte geliefert werden. **X**
* Bei einem Code-Review wird der zu testende Code aus einem Test-Rechner ausgeführt.

# Codeanhang:

## Methode „addMul“:

public String addMul() {

String a = „3“;

int b = 7;

int c = 4;

return a + b \* c;

}

## Methode „m“:

public int m(int p) {

int i = 0;

for(int k = 0; i<p; ++i) {

k = k + 2\*i

}

return i;

}

## Methode „m2“:

public int m2(int p) {

int result = 0;

int i = 1;

while(i<=p) {

result = result + i \* 2;

i++;

}

return result;

}