**PR1 – Formular für Lesenotizen**

**WS2020/21**

|  |  |  |  |
| --- | --- | --- | --- |
| Nachname  Abdel Kader | Vorname  Schehat | Matrikelnummer  1630110 | Abgabedatum:  22.10.2020 |

**Primitive Daten und definite Schleifen (L.2.6-L2.8)**

**Lernzielfragen:**

1. Gegeben sei folgender Code:

public class Test {

public static void main(String[] args) {

int a=5, b, c=6;

System.out.println(a+b+c);

}

}

Ist der Code übersetzbar? Falls ja: wie lautet die Ausgabe? Falls nein: wieso nicht?

* Der Code ist nicht übersetzbar, weil b nicht deklariert ist.

1. Schreiben Sie eine for-Schleife, die die Funktionswerte der folgenden Parabel im Definitionsbereich [-10,10] im Abstand von Schritten der Schrittweite 0,5 ausgibt. f(x) = 4x² - 3x + 5.

* for (double i = -10; i <= 10; i+= 0.5 ) {

System.out.println(4\*i\*i - 3\*i + 5);

}

1. Gegeben ist die folgende Schleife:

for (int count = 1; count <= 5; count++) {

...

}

Ergänzen Sie ein Statement, so dass die Ausgabe wie folgt lautet: 9 5 1 -3 -7

* System.out.println(-4\*count + 13);

Ergänzen Sie ein alternatives Statement, so dass die Ausgabe wie folgt lautet: 16 27 38 49 60

* System.out.println(11\*count + 5);

1. Was sind Vor- und Nachteile der Lokalität von Variablen?

* Vorteile: Vermeidung von Namenskollisionen & Zugriff auf Variablen steuerbar
* Nachteil: Aufwendiger auf Variablen anderer Methoden zuzugreifen durch Parameter und Rückgabewerten von Methoden

**Notizen:**

**Mehrfache Variablendefinition**

int a, b, c; bzw.: int a = 2, b = 3, c = -4;

**Typinferenz für lokale Variablen**

Den Typ der Variablen aus dem Initialwert ableiten: var x = 3;

Gibt es erst seit Java 10 x = 3.4; => geht nicht, Typen in Java sind statisch

var x; = > geht natürlich nicht

**Die for-Schleife**

Bsp.: for (int i = 1; i <= 6; i++) {

System.out.println(i + " zum Quadrat ist " + (i \* i));

}

* For-Schleife mit einem Statement kann man die geschweiften Klammern weglasse:

for (int i = 1; i <= 3; i++)

System.out.println("Dies wird 3x gedruckt");

System.out.println("Dies auch ..., oder?"); // eben nicht

**Degenerierte for-Schleifen**

Bezeichnet man Schleifen, die die Test-Bedingung in der Initialisierung nicht erfüllen oder Endlosschleifen

**Zahlenfolgen erzeugen**

Bsp.: 2 7 12 17 22 mit for (int count = 1; count <= 5; count++)

|  |  |  |  |
| --- | --- | --- | --- |
| Count | Ausgabe | Abstand bestimmen: (Count2 – Count1) \* Count | Anpassen: + oder - |
| 1 | 2 | (7 – 2) = 5 | 5 – **3** = 2 |
| 2 | 7 | 10 | 10 – **3** = 7 |
| 3 | 12 | 15 | 15 – **3** = 12 |
| 4 | 17 | 20 | 20 – **3** = 17 |
| 5 | 22 | 25 | 25 – **3** = 22 |

= > System.out.println(5\*count - 3);

**Geltungsbereich**

Variablen haben einen begrenzten Geltungsbereich. Der Geltungsbereich einer Variablen reicht von der Deklaration bis zum Ende der geschweiften { } Klammern, innerhalb derer sie deklariert wurde. Also existiert diese dann nur in der Methode oder in der for-Schleife, in der Sie deklariert wurde.

**Lokale Variablen**

Eine in einer Methode deklarierte Variable nennt man lokale Variable. Man soll immer so lokal wie möglich die Variablen deklarieren.