

4 Punkte

Erstellen Sie eine Methode mit Rückgabewert `int[][]`, welche eine Matrix (2-Dimensionales Array) mit parametrisierter Grösse erstellt und diese mit Zufallszahlen zwischen 0-9 abfüllt. Die Methode soll `MakeMatrix` heissen und folgendem Kopf entsprechen:

```
public static int[][] MakeMatrix(int i1, int i2){
```

A large grid of graph paper with 20 columns and 10 rows. The grid is composed of small squares, with a slightly larger margin at the top for writing.

5 Punkte

Schreiben Sie eine Methode ohne Rückgabewert, die eine Matrix ausdrückt, die als Parameter der Methode übergeben wird. Zwischen den Matrixelementen einer Zeile soll ein Leerschlag sein. Nach jeder gedruckten Zeile aus der Matrix soll ein Zeilenumbruch eingefügt werden. Die Methode soll `PrintMatrix` heissen und folgendem Kopf entsprechen:

```
public static void PrintMatrix(int[][] matrix){
```

A large grid of graph paper with 20 columns and 10 rows. The grid is composed of small squares, with a slightly larger square at the top left corner, likely for a title or header. The grid is empty and ready for use.

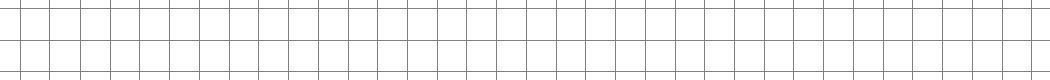
Vervollständigen Sie folgende `main`-Methode, damit eine neuerstellte Matrix mit den Dimensionen 10x10 mit der Methode `MakeMatrix` über die Methode `PrintMatrix` gedruckt wird. Alles auf einer Zeile!

```
public static void main(String[] args) {
```

[illegible]

2 Punkte

Sie möchten wissen wie viel Zeit folgender Funktionsaufruf `MakeMatrix(10000, 10000)` beansprucht. Ergänzen Sie den Funktionsaufruf mit einer Zeitmessung und geben Sie die benötigte Zeit in Millisekunden aus.



5. Algorithmen implementieren (Collatz-Folge oder Ulam-Sequenz)

9 Punkte

Das Gewinnspiel

Das Spielfeld des Gewinnspiel besteht aus beliebig vielen, ab Eins (1) indexierten Feldern.

Auf jedem Feld gibt es einen Gewinn, welcher entsprechend des Feldindexes ist.

Auf dem Feld 8 liegen 8.-- Fr. usw.

Der Betrag von jedem Feld, das man besucht, darf als Gewinn mitgenommen werden.

Die Spielvorschrift besteht aus den folgenden Regeln:

1. Wählen Sie ein Feld zwischen 1 und 99.
2. Nehmen Sie das Geld und gehen Sie nach folgender Vorschrift zum nächsten Feld:
 - 2.1 Ist die Feldnummer ungerade? Gehen Sie zu Feld $\cdot 3 + 1$
 - 2.2 Ist die Feldnummer gerade? Gehen Sie zu Feld $/ 2$

Hinweis: Die Spielvorschrift dieser Aufgabe ist als Collatz-Folge oder Ulam-Sequenz bekannt.

3. Wiederholen Sie Schritt 2, bis Sie auf dem Feld 1 landen. Die Gewinne aller Felder werden aufsummiert

Ergänzen Sie das Unterprogramm `gewinn(int i)`, welches zu einem gewählten Startwert den Gewinn ausgibt.

In einer gegebenen `for`-Schleife werden alle 1-99 Startwerte in der Methode `gewinn(int i)` ausgeführt.

Ergänzen Sie die Methode `gewinn(int i)` so, dass der maximale Gewinn dieses Gewinnspiels ausgegeben

werden kann.

Ausgabe:

Gewinn für Feld 1 ist 1

Gewinn für Feld 2 ist 3

Ge
...

Gewinn für Feld 97 ist 103254

Gewinn für Feld 98 ist 904

Gewinn für Feld 99 ist 1716

Der Maximalgewinn beträgt 105098 Fr.

```
public static void main(String[] args) {
    StartGewinnSpiel();
}

public static int maximalgewinn = 0;

public static void StartGewinnSpiel() {
    for(int i = 1; i <= 99; i++) {
        System.out.println("Gewinn für Feld " + i + " ist " + gewinn(i));
        System.out.println("Der Maximalgewinn beträgt " + maximalgewinn + " Fr.");
    }
}

public static int gewinn(int i) {
```

A full page of blank graph paper with a uniform grid of small squares. The grid covers the entire area of the page, leaving no margins or other markings.