Objektorientierte Programmierung I
freiwillige Übungsaufgaben
17.05.2022

Legende:

🌶 = einfach
🌶🌶 = mittel
🌶🌶🌶 = schwer

🌶 Aufgabe 1

Benutze die `next()`-Methode der `Scanner`-Klasse der Java-Standardbibliothek (siehe Übungsblatt 4), um einen String von der Tastatur einzulesen. Wurde der Text „dfg987345“ eingegeben, soll das Programm „Zugriff gestattet“ ausgeben; ansonsten soll es „Zugriff verweigert, falsches Passwort“ ausgeben.

Hinweis für die nächste Aufgabe:

Die `String`-Klasse verfügt über eine Methode `charAt()`, die einen Index (Datentyp: `int`) als Argument erwartet und den Buchstaben an der entsprechenden Position zurückgibt (Rückgabetyp: `char`).

Beispiel:

```
String text = "Test";
System.out.println(text.charAt(0));
System.out.println(text.charAt(2));
```

Ausgabe:

```
T
s
```

Darüber hinaus verfügt die Klasse `String` auch über eine `length()`-Methode, welche die Anzahl der Zeichen innerhalb des Strings zurückgibt. Damit ist es möglich, über die einzelnen Zeichen eines Strings zu iterieren.

Beispiel:

```
String text = "Test";
for (int i = 0; i < text.length(); ++i) {
    System.out.println(text.charAt(i));
}
```

Ausgabe:

```
T
e
s
t
```

Objektorientierte Programmierung I
freiwillige Übungsaufgaben
17.05.2022

🌶️🌶️ Aufgabe 2

Bei einem Palindrom handelt es sich um eine Zeichenkette, die vorwärts und rückwärts gelesen identisch ist. Schreibe ein Programm, das einen String von der Tastatur einliest und überprüft, ob es sich bei der Eingabe um ein Palindrom handelt. Ein beispielhafter Ablauf des Programms könnte z. B. wie folgt aussehen:

```
Bitte gib ein Wort ein: lagerregal
Bei der Eingabe "lagerregal" handelt es sich um ein Palindrom.
```

🌶️ Aufgabe 3

Schreibe ein Programm, das beliebig viele positive Zahlen vom Datentyp `double` von der Tastatur einliest (Hinweis: `nextDouble()`-Methode der `Scanner`-Klasse). Wird die Zahl Null oder eine negative Zahl eingegeben, soll die Eingabe abbrechen. Danach soll das Programm das arithmetische Mittel (also den Durchschnitt) aller eingegebenen Zahlen berechnen und am Bildschirm ausgeben.
Überlege, wie du das Problem der beliebig vielen Eingaben lösen kannst. Versuche also **nicht**, alle Eingaben in einem (sehr großen) Array zu speichern.
Überlege auch, wie das Programm damit umgehen kann, dass die erste eingegebene Zahl eventuell bereits zum Abbruch führt.

Objektorientierte Programmierung I
freiwillige Übungsaufgaben
17.05.2022

🌶🌶🌶 Aufgabe 4

Die sog. „Collatz-Folge“ ist wie folgt definiert (Quelle: https://de.wikipedia.org/wiki/Collatz-Problem):

- Beginne mit irgendeiner natürlichen Zahl $n > 0$.
- Ist n gerade, so nimm als nächstes $n/2$.
- Ist n ungerade, so nimm als nächstes $3n + 1$.
- Wiederhole die Vorgehensweise mit der erhaltenen Zahl.

Zitat: „Das Collatz-Problem […] ist ein ungelöstes mathematisches Problem, das 1937 von Lothar Collatz gestellt wurde. […] Das Problem gilt als notorisch schwierig, obwohl es einfach zu formulieren ist. […] Anscheinend mündet die Folge mit jedem $n > 0$ in den Zyklus $4, 2, 1$, egal, mit welcher natürlichen Zahl $n > 0$ man beginnt.“

Aufgabenstellung: Schreibe ein Programm, das für jedes $n \in [1; 100]$ zeigt, dass die zugehörige Collatz-Folge in den Zyklus $4, 2, 1$ mündet. Gib dabei auch jedes Folgenglied am Bildschirm aus. Wenn der Zyklus einmal erreicht wurde, soll die Folge für die aktuelle Zahl n abbrechen. Die Ausgabe des Programms sollte also wie folgt aussehen:

```
n = 1: 1, 4, 2, 1
n = 2: 2, 1, 4, 2, 1
n = 3: 3, 10, 5, 16, 8, 4, 2, 1
…
```

Hinweis: Du darfst annehmen, dass der Zyklus erreicht wurde, sobald das Programm bei der Zahl Vier ankommt. Dennoch soll die Ausgabe wie eben gezeigt aussehen.

🌶 Aufgabe 5

Werden in Java die Daten von Arrays auf dem Stack oder auf dem Heap gespeichert? Überlege dir, wie man das überprüfen könnte.

Objektorientierte Programmierung I
freiwillige Übungsaufgaben
17.05.2022

🌶🌶 Aufgabe 6

Beim sog. „Sieb des Eratosthenes“ handelt es sich um einen Algorithmus, mit dem effizient Primzahlen im Intervall $I = [1; n]$ berechnet werden können. Der „Preis“ dafür ist ein erhöhter Speicherverbrauch (verglichen mit der Implementierung in der Musterlösung zu Übungsblatt 4). Der Algorithmus läuft wie folgt ab:

- Es wird ein `boolean`-Array mit $n + 1$ Elementen angelegt.
- Alle Elemente des Arrays werden auf den Wert `true` gesetzt.
- Für alle Ganzzahlen $k \in [2; n]$ tue folgendes:
  - Wenn das Array-Element mit Index k den Wert `false` hat, tue nichts.
  - Ansonsten: Für alle Vielfachen j von k mit $2k \leq j \leq n$ tue folgendes:
    - Setze das Array-Element mit Index j auf `false`.
- Gib alle Array-Indizes aus, die größer als 1 sind und den Wert `true` haben.

🌶🌶🌶 Aufgabe 7

Gegeben sei das folgende Programm:

```
import java.util.Random;

class Sort {

    public static void main(String[] args) {
        int[] numbers = new int[20];
        Random random = new Random();

        for (int i = 0; i < numbers.length; ++i) {
            numbers[i] = random.nextInt(50) + 1;
        }

        // hier Code einfügen

        for (int i = 0; i < numbers.length; ++i) {
            System.out.print(numbers[i] + " ");
        }
        System.out.println();
    }

}
```

Füge an der markierten Stelle Programmcode ein, der das Array `numbers` aufsteigend sortiert. Versuche zunächst, dir den nötigen Algorithmus dafür selbst zu überlegen. Solltest du keine Lösung finden, recherchiere im Internet den „Bubblesort“-Algorithmus.