

Übungsblatt #4a

Aufgabe 1:

Schreiben Sie ein Programm, das ein 2-dimensionales boolesches Array anlegt, dieses Array zufällig mit Werten initialisiert. Verwenden Sie dazu die Funktion `java.lang.Math.random()`, die zufällige Werte zwischen 0 und 1 liefert. Dieses Array soll auf dem Bildschirm in einer rechteckigen Form wieder ausgegeben werden.

Aufgabe 2:

Modifizieren Sie Ihr Programm aus der letzten Aufgabe derart, dass Sie 2 Methoden schreiben. Die 1. Methode bekommt ein 2-dimensionales boolesches Array übergeben und initialisiert dieses zufällig. Die 2. Methode bekommt ein 2-dimensionales boolesches Array übergeben und druckt dieses in einer rechteckigen Form auf dem Bildschirm aus. Halten Sie diese beiden Methoden allgemein, d.h. die Größe des Arrays soll nicht als konstante Zahl in den Methoden vorkommen.

Aufgabe 3:

Programmieren Sie eine Klasse `Life`, die ein 2-dimensionale boolesche Array als Objektvariable enthält, eine Objektmethode `init` enthält, die das Array zufällig initialisiert und eine Objektmethode `print` enthält, die das Array ausdrückt.

Aufgabe 4:

Implementieren Sie für Ihre `Life` Klasse eine Objektmethode, die zu jeder Position berechnet, wie viele unmittelbare Nachbarfelder auf `true` gesetzt sind.

Aufgabe 5:

Erweitern Sie Ihre `Life` Klasse um eine Methode, die zu einer gegebenen Spielsituation eine folgende berechnet:

Ausgehend von einer vorgegebenen Anfangskonfiguration werden alle Felder gleichzeitig auf folgende 3 Regeln getestet:

1. Weiterleben: Wenn ein Stein in seiner unmittelbaren Nachbarschaft (1-Feld-Umgebung) 2-3 weitere Steine hat, überlebt er die Runde.

2. Sterben: Wenn ein Stein in seiner unmittelbaren Nachbarschaft (1-Feld-Umgebung) weniger als 2 oder mehr als 3 weitere Steine hat, stirbt er (an Einsamkeit oder Übervölkerung) nach dieser Runde.
3. Neu Geboren: Wenn ein unbelegtes Feld an genau 3 Steine grenzt, wird dort ein neuer Stein geboren.

Nachdem eine neue Spielfeldsituation berechnet worden ist, soll diese ausgegeben werden und dann erneut berechnet werden, usw. Nehmen Sie für die Nachbarschaft an, dass das Spielfeld eine Kugel ist, d.h. wenn Sie zum linken Rand rausgehen, kommen Sie rechts wieder rein, oben und unten analog.

Aufgabe 6:

Implementieren Sie 3 Konstruktoren für Ihre Lifeklasse, die die Initialisierung übernehmen; einer bekommt die Anzahl der Zeilen und Spalten, einer nur die Anzahl der Zeilen und setzt die Anzahl der Spalten konstant auf 30, und einer bekommt keine Parameter, sondern setzt die Anzahl der Spalten auf 30 und die Anzahl der Zeilen auf 40.

Aufgabe 7:

Deklariieren Sie die Objektvariablen als private.

Aufgabe 8:

Überprüfen Sie, welche der Methoden public sein müssen: deklarieren Sie so viele wie möglich als private.