A3-2

A Spiel des Lebens – Variante 1

Schreiben Sie eine Klasse Life, die das Spiel des Lebens simuliert. Verwenden Sie dazu eine boolesche Matrix, die einem System von Zellen entspricht, die entweder tot oder lebendig sind. Das Spiel besteht darin, den Wert jeder Zelle zu prüfen und in Abhängigkeit von den Werten der Nachbarzellen den Wert der Zelle zu aktualisieren. (Nachbarzellen sind alle Zellen, die die Zelle umgeben, auch die auf der Diagonalen). Grundsätzlich bleiben lebendige Zellen lebendig und tote Zellen tot. Es gelten die folgenden Ausnahmen:

1. Eine tote Zelle mit genau drei lebenden Nachbarzellen wird als lebendig markiert.
2. Eine lebende Zelle mit genau einer lebenden Nachbarzelle stirbt.
3. Eine lebende Zelle mit mehr als drei lebendigen Nachbarzellen stirbt.

Initialisieren Sie das Programm mit einer zufälligen Belegung oder verwenden Sie das nachfolgende Muster.

```
Generation t
------
--*---
---*--
-***--
------
------
```

Die Ausgabe für das Muster für t=5 Simulationsschritte sieht wie folgt aus:

```
Generation t+1
------
------
-*-*--
--**--
--*---
------
Generation t+2
------
------
---*--
-*-*--
--**--
------
Generation t+3
------
------
--*---
---**-
--**--
------


Generation t+4
------
------
---*--
----*-
--***-
------
Generation t+5
------
------
------
--*-*-
---**-
---*--
```

Hinweise:

1. Schreiben Sie eine Methode, die die Anzahl der lebendigen Nachbarzellen für eine Zelle berechnet. Achten Sie dabei auf die Randfälle, Anfang/Ende einer Zeile, Anfang/Ende einer Spalte und vergessen Sie nicht, dass die Zelle, deren Nachbarn zu bestimmen sind, auf keinen Fall mitgezählt werden darf.
2. Merken Sie sich in zwei Arrays für jeden Simulationsschritt die Zellen die sich ändern, ein Array für die Zellen die lebendig werden und ein Array für die Zellen, die sterben.
3. Berechnen Sie die neue boolsche Matrix, wenn Sie für alle Zellen die Nachbarzellen ausgewertet haben.
4. Zerlegen Sie das Problem, indem Sie die folgenden Methoden schreiben:
    a. ***naechste_generation***
    b. ***lebende_nachbar_zellen(i,j)***: berechnet die Anzahl der lebenden Nachbarzellen zu einer Zelle an Position i,j der booleschen Matrix
    c. ***spielen(n):*** berechnet die nachfolgenden Generationen n-mal und gibt sie nach jedem Schritt aus
    d. ***spielfeld_zeichnen***
5. Geben Sie bei der Initialisierung die Größe des Spielfeldes vor.

B Spiel des Lebens – Variante 2

Implementieren Sie das Spiel des Lebens in der Toolbox.

Implementieren Sie die folgenden Methoden für die Klasse ***Life***:

1. die Initialisierung mit zwei Parametern,
   a. einem Parameter für die Anzahl der Zellen
   b. einem Parameter für das Muster der Anfangsgeneration
2. ***naechste_generation()***
   a. Berechnet nach dem Verfahren in A Variante 1 die nachfolgende Generation
3. ***simuliere(n)***
   a. Berechnet n-mal die nachfolgende Generation und gibt diese auf der Leinwand aus.
   b. Verwalten Sie die Generationen in einem Array und berechnen Sie die nachfolgenden Generation auf Basis der letzten im Array eingetragenen Generation.
4. eine Methode ***zuruecksetzen(),*** die das Spiel auf den Anfangszustand zurücksetzt.

Schreiben Sie eine Klasse ***Zelle***, die ihre Position und den Zustand (lebendig / tot) kennt und für die Darstellung ein Quadrat verwendet. Tote Zellen sollen als nicht gefüllte Quadrate dargestellt werden. Lebendige Zellen sollen als gefüllte Quadrate dargestellt werden. Die Klasse Zelle hat eine Methode für das ***zeichnen*** sowie Methoden, um den Zustand zwischen lebendig und tot zu wechseln und um diesen Zustand abzufragen.

Um nicht gefüllte Rechtecke anzuzeigen muss die Methode ***farbe_aendern*** der Klasse Rechteck mit einem zweiten Parameter (boolescher Parameter) aufgerufen werden. Ist dieser ***false*** wird das Rechteck ohne Füllung dargestellt.

Die Ausgabe für das obige Muster für t=5 sieht wie folgt aus: