Game of Life

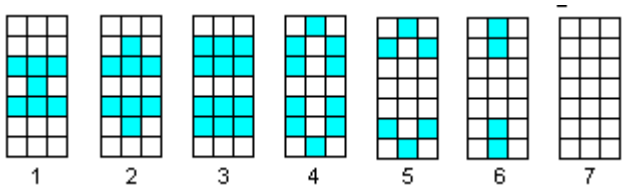
AvHG, Inf, My

Das „Lebens-Spiel“ wurde 1970 von dem englischen Mathematiker John Conway entwickelt. Es simuliert die Population von einfachen Lebewesen, die gegenseitig von einander abhängig sind. Der Lebensraum der Lebewesen wird durch ein Gitternetz aus quadratischen Zellen dargestellt. In jeder Zelle kann sich ein Lebewesen befinden (oder auch nicht).

Nach einem Lebenszyklus überlebt ein Lebewesen, wenn es zwei oder drei Nachbarn hat. Wenn es weniger als zwei Nachbarn hat, stirbt es aus Einsamkeit. Wenn es mehr als drei Nachbarn hat, stirbt es wegen Überbevölkerung.

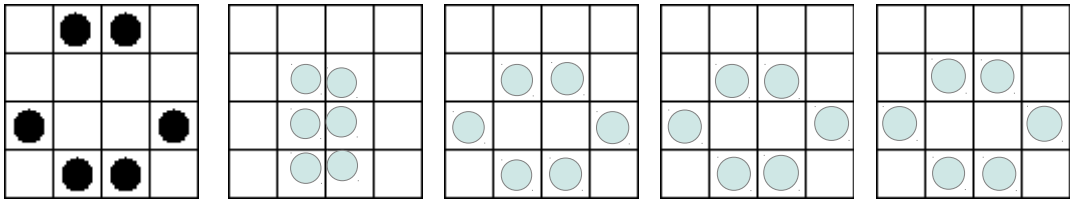
Wenn eine leere Zelle genau drei lebende Nachbarn hat, entsteht in der Zelle ein neues Leben.

In dem nachfolgenden Beispiel beginnt die Simulation mit einer Population von 7 Lebewesen. Die Population wächst bis auf 12 Lebewesen an und stirbt nach 6 Generationen aus:



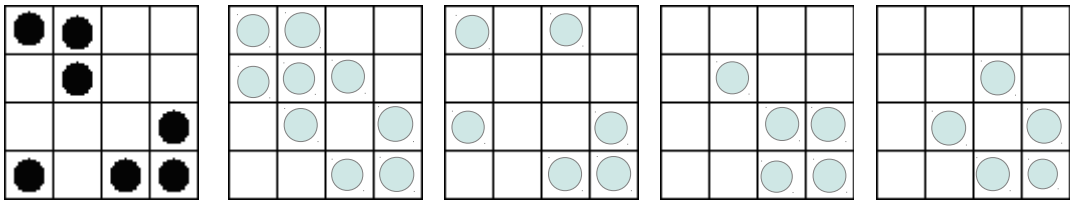
Führe zunächst einige einfache Simulationen per Hand durch:

(a)

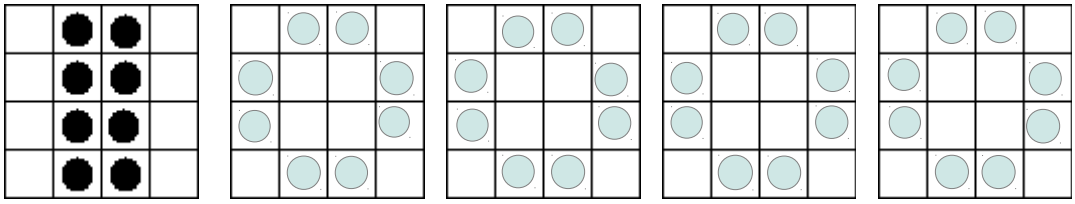


→ stabil

(b)

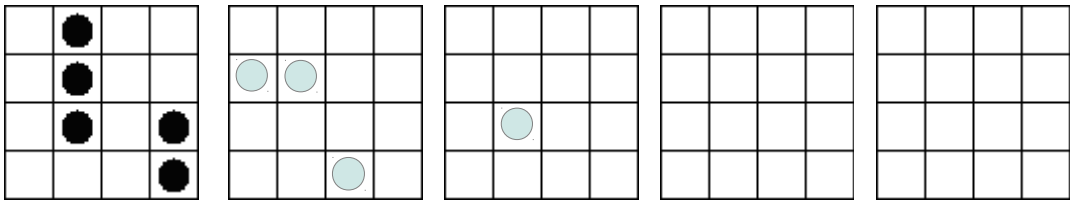


(c)

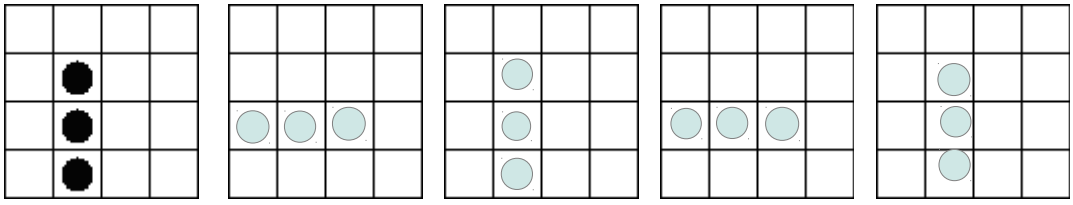


→ stabil

(d)



(e)



→ ***Flip-Flop***

Programmieraufgabe:

Schreibe eine Java-Anwendung, die das ***Game of Life*** simuliert. Gehe in folgenden Teilschritten vor:

(a) Zum Testen ist es hilfreich, wenn man anfangs mit wenigen Zellen arbeitet (z.B. 3*3 Zellen). Wenn alles funktioniert, kann man dann ein großes Feld erzeugen (z.B. 150*100 Zellen). Definiere deshalb Konstanten, die die Anzahl der Zellen in x- und in y-Richtung festlegen. Lege in einer weiteren Konstanten die Breite einer Zelle fest.

(b) Erzeuge ein zweidimensionales boolesches Array, das für jede Zelle abspeichert ob sie lebendig (`true`) oder tot (`false`) ist. Fülle das Array mit zufälligen Anfangswerten.

(c) Zeichne die Zellen. Eine lebendige Zelle wird als ausgefülltes Quadrat gezeichnet. Eine tote Zelle wird als hohles Quadrat gezeichnet.

(d) Programmiere eine Methode, die die Anzahl der lebenden Nachbarn einer Zelle zählt. Dabei wird zunächst davon ausgegangen, dass sich die Zelle in der Mitte des Feldes befindet (das heißt sie besitzt alle 8 Nachbarn). Die Methode erhält die Koordinaten der Zelle (x- und y-Position) als Parameter übergeben und gibt die Anzahl der Nachbarn zurück. Teste die korrekte Funktionsweise der Methode gründlich aus, ehe du weiter arbeitest. Rufe dazu die Methode für irgendeine Zelle auf (z.B. Zelle (2|1) und überprüfe den Rückgabewert in dem du die Anzahl der Nachbarn von Hand abzählst.

(e) Erweitere die Methode aus Aufgabe (d) so, dass sie auch für Zellen funktioniert, die sich am Rand des Gitternetzes befinden. Dazu musst du durch if-Abfragen sicherstellen, dass beim Zählen der lebenden Nachbarn niemals Koordinaten von Zellen aufgerufen werden, die nicht existieren. Teste die Methode für verschiedene Rand-Zellen aus (linker Rand, rechter Rand, oberer Rand und unterer Rand).

(f) Programmiere eine Methode, die entsprechend der Regeln von Conway berechnet, ob eine bestimmte Zelle im nächsten Zyklus lebt oder nicht. Die Methode erhält als Parameter die x- und y-Koordinate der Zelle übergeben sowie die Anzahl ihrer Nachbarzellen (die mit der Methode aus Aufgabe (e) ermittelt wird). Der Rückgabewert ist ein boolescher Wert, der angibt, ob die Zelle im nächsten Zyklus lebt (true) oder nicht (false).

(g) Rufe in der `myPaint()`-Methode des Anwendungsfensters die Methoden aus Aufgabe (e) und (f) für jede Zelle des Feldes einmal auf (durch zwei ineinander geschachtelte Schleifen), um den nächsten Lebenszyklus zu berechnen. Wichtig ist, dass die neuen „Lebenswerte“ nicht sofort in das boolesche Array hineingeschrieben werden, weil dann die Berechnung der Nachbarzellen nicht mehr korrekt erfolgen würde. Deshalb muss ein zweites Array von booleschen Werten angelegt werden, das die neuen „Lebenswerte“ vorübergehend abspeichert. Nachdem die neuen „Lebenswerte“ aller Zellen berechnet wurden, werden die booleschen Werte des Hilfs-Arrays in das echte Array hinüber kopiert. Dazu benötigt man wieder zwei ineinander geschachtelte Schleifen um jeden Wert einzeln zu kopieren. Anschließend wird das veränderte Feld mit dem in (c) programmierten Code erneut gezeichnet.

Füge in das Anwendungsfenster ein Objekt der Klasse `Timer` ein, damit die `myPaint()`-Methode in regelmäßigen Zeitabständen aufgerufen wird, um einen neuen Lebenszyklus zu berechnen.