Übung 2a [12]: Game Of Life

Schreiben Sie ein Programm, das das Spiel "Game of Life" implementiert.

Schauplatz dieses Spiels ist eine zweidimensionale Matrix aus Zellen. Jede Zelle hat einen Zustand, der entweder "tot" oder "lebendig" sein kann. Das Spiel läuft in diskreten Schritten ab. Bei jedem Schritt kann sich der Zustand einer Zelle verändern. Ob sich der Zustand einer Zelle verändert, hängt von ihren acht Nachbarzellen ab:

Eine Zelle, die zum Zeitpunkt t tot war, wird dann und nur dann zur Zeit t+1 lebendig, wenn genau drei ihrer acht Nachbarzellen zum Zeitpunkt t gelebt haben.

Eine Zelle, die zum Zeitpunkt t gelebt hat, stirbt zur Zeit t+1, wenn zur Zeit t weniger als zwei oder mehr als drei Nachbarn am Leben waren.

Zellen am Rand der Matrix haben entsprechend weniger Nachbarn, es gelten aber trotzdem die gleichen Regeln.

Implementierungshinweise:

Sie können die Größe der Matrix alx fix mit 10x10 annehmen.

Die Matrix soll zu Beginn des Programms mit einem willkürlichen Zustand (Hinweis: Math.random() verwenden) initialisiert werden. Anschließend sollen der Ursprungszustand und die ersten 15 Folgezustände ausgegeben werden. Tote Zellen sollen durch “-“ und lebendige Zellen durch “X” dargestellt werden.

Beispiel:

```
Anfangszustand:          Zustand der Generation 1     Zustand der Generation 2
----------               -----X----                   ----------
----XXX---               -----X----                   ----XXX---
----------               -----X----                   ----------
----------               ----------                   ----XX----
----XX----               ----XX----                   ----X-X---
-----X----               ----XXX---                   ----X-X---
------X---               ----------                   -----X----
----------               ----------                   ----------
----------               ----------                   ----------
----------               ----------                   ----------
```

Übung 2b [6+6]: Primzahlensieb

Schreiben Sie zwei Programme, die alle Primzahlen kleiner als N berechnen, speichern und anschließend ausgeben. Der Grenzwert ist wie im ersten Beispiel eine Konstante, die als einzige für einen anderen Bereich verändert werden darf.

Speichern Sie die Primzahlen beim ersten Programm in einem Array (überlegen Sie sich die minimal notwendige Größe!) und beim Zweiten in einem Vektor (Verwenden Sie zum Durchlauf eine Enumeration und keinen Zugriff per elementAt()!).

- Berechnen Sie jeweils den Füllgrad des Arrays/Vektors.
- Überlegen Sie sich, mit welchen Werten sie Ihre Programme zumindest testen müssen!

Beispiel:

```
Ausgabe der Primzahlen bis 100

Array:  2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73
79 83 89 97
Vektor: 2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73
79 83 89 97

Anzahl der Primzahlen bis 100: 25
Fuellgrad Array:  50.0 %
Fuellgrad Vektor: 62.5 %
Fuellgrad Vektor nach trimToSize: 100.0 %
```