```java
//@author Maxmilian Raspe
/**
 *
 */

import java.awt.Color;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.Timer;



/**
 * @author Sonja 10.10.2018
 */
public class SpaceInvaders {

    private static boolean fertig = false;
    private static boolean schuss;
    private static int spaceBarX; // linker Rand der SpaceBar
    private static final int spaceBarY = 22; // Zeile der SpaceBar
    private static int obereGrenze = 5; // hoechste Zeile, in der Aliens
    // vorkommen
    private static int aktuellspaceBarX = 30;
    // Eine Konstante die angibt, dass sich die SpaceBar nach links bewegt.
    private static final boolean SPACE_BAR_MOVE_TO_THE_LEFT = true;

    // Eine Konstante die angibt, dass sich die SpaceBar nach rechts bewegt.
    private static final boolean SPACE_BAR_MOVE_TO_THE_RIGHT = false;

    private static int numberOfAliens = 0; // speichert die Anzahl der Aliens
    private static Timer alienTimer; // Timer fuer das Updaten der Aliens
    private static Timer bulletTimer; // Timer fuer das Updaten der Bullets

    /**
     * In der Console.java wurde ein KeyListener eingebunden, dessen Methoden in
     * der Console ganz unten implementiert sind. In einer dieser Methoden
     * stehen 2 Zeilen Code, wobei eine der Zeilen auf eine Methode der
     * Hauptklasse zugreift.
     */

    /**
     * @param args
     *            Programmargumente
     */
    public static void main(String[] args) {
        // Spiel vorbereiten
        prepareGame();

        // Timer erstellen
        initializeTimers();
        //Spiel starten
        startGame();
    }

    /**
     * Setzt Voreinstellungen fuer das Spiel und die initiale Setzung der
     * Objekte im Feld.
     */
    private static void prepareGame() {
        // Farben setzen
        Console.setBackground(Color.black);
```

```
    Console.setForeground(Color.white);
    Console.gotoXY(0, 0);
    Console.write("links - a, rechts - d, schieÃŸen - w");
    createAliens(); // Aliens erstellen
    spaceBarX = 10;
    drawSpaceBar(); // Spacebar erstellen
}

/**
 * Timer erstellen, initialisieren und starten. Sobalt die Timer gestartet
 * worden sind startet direkt das Spiel.
 */
private static void initializeTimers() {
    // Timer fuer das Updaten der Bullets
    bulletTimer = new Timer(400, new ActionListener() {

        @Override
        public void actionPerformed(ActionEvent arg0) {
            // Display ausschalten, damit das Programm fluessig laeuft
            Console.displayOff();
            // Bullets updaten
            updateBullets();
            // Display wieder einschalten, aka Aenderungen wirksam machen
            Console.displayOn();
        }
    });

    // Timer fuer das Updaten der Aliens
    alienTimer = new Timer(5000, new ActionListener() {

        @Override
        public void actionPerformed(ActionEvent e) {
            Console.displayOff();
            updateAliens();
            Console.displayOn();
        }
    });

    // damit keine Aktionen verschwinden
    bulletTimer.setCoalesce(true);
    alienTimer.setCoalesce(true);

    // Starten der Timer
    bulletTimer.start();
    alienTimer.start();
}

/**
 * Aktion nach Tasteneingabe. Bewegung nach links oder rechts
 * (moveSpaceBar(boolean)) mit entsprechender Eingabe) oder schiessen
 * (shoot()) einer Bullet.
 *
 * @param c
 *            Ueber Tastatur eingegebener char dem eine entsprechende
 *            Methode zugeordnet wird.
 */
public static void move(char c) {
    // TODO
    char a = 'a';    //bestimmter tastendruck wird zugeordnet zu taste
    char d = 'd';
    char w = 'w';
```

```
        if(c == a) moveSpaceBar(true);
        if(c == d) moveSpaceBar(false);
        if(c == w) shoot();
    }

    /**
     * Erstellt Aliens zu Beginn des Spiels. Kann fuer mehrere Level erweitert
     * werden.
     *
     * Erstellt 11 * 5 Aliens im oberen, mittleren Bereich der Console.
     * Verwendet dazu die setAliensInHoricontalLine()-Methode.
     */
    private static void createAliens() {
        // TODO
        setAliensInHoricontalLine(28,3,11);         //setzt aliens
    }

    /**
     * Hilfsmethode zum Malen von Aliens in einer Reihe.
     *
     * @param column
     *            Spalte, in der die Reihe anfaengt.
     * @param row
     *            Zeile, in der die Reihe steht.
     * @param length
     *            Laenge der Reihe in Bezug auf Anzahl der Aliens ("@")
     */
    private static void setAliensInHoricontalLine(int column, int row, int length) {
        // TODO
        Console.setForeground(Color.green);  //zeichnet aliens, 11 aliens pro zeile, 5
zeilen
        for(int i = 3; i < 8; i++) {
            Console.gotoXY(column, i);
            for(int j = 0; j < length; j++) {
                Console.write(" @");
            }
        }
        numberOfAliens = 55;


    }

    /**
     * Verschiebt alle Aliens eine Zeile nach unten, prÃ¼ft dabei mittels
     * isGameOver()-Methode ob die Aliens die Zeile der SpaceBar erreicht haben
     * und ruft dann die end(boolean)-Methode mit entsprechender Eingabe auf.
     */
    private static void updateAliens() {

        Console.displayOff();
        for (int row=21; row >= 3; --row) {
            for (int column=0; column <= 77; column++) {
                Console.gotoXY(column, row-1);
                char upperRowContent = Console.getChar();
                Console.gotoXY(column, row);
                char lowerRowContent = Console.getChar();
                if (upperRowContent != lowerRowContent) {
                    Console.setForeground(Color.green);
                    Console.setChar(upperRowContent);
                }
            }
        }
```

```
        isGameOver();
        Console.displayOn();
    }

    /**
     * Testet ob die Aliens auf HÃ¶he der Spacebar angekommen sind.
     *
     * @return true wenn das Spiel verloren wurde, false ansonsten
     */
    private static boolean isGameOver() {
      if(numberOfAliens == 0) {
            fertig = true;
            end(true);
      }
        for (int column=0; column < 77; column++) {
            Console.gotoXY(column, 21);
            if (Console.getChar() == '@') {
                fertig = true;
                end(false);
                return true;
            }
        }
        return false;
    }

    /**
     * Zeichnet die Spacebar an ihrer aktuellen Position (spaceBarX, spaceBarY).
     * Die SpaceBar sie wie folgt aus: "  ^  ".
     */
    private static void drawSpaceBar() {
        // TODO
        Console.setForeground(Color.yellow);
        Console.gotoXY(aktuellspaceBarX + spaceBarX , spaceBarY);
        Console.write(" ^ ");
        aktuellspaceBarX = spaceBarX + aktuellspaceBarX;
    }

    /**
     * Bewegt die Spacebar um eine Position (2 Stellen) nach links (true) oder
     * rechts (false). Es wird dabei zunÃ¤chst geprÃ¼ft, ob eine Bewegung in die
     * angegebene Richtung noch mÃ¶glich ist. Sollte sich die SpaceBar bereits am
     * Rand des Spielfeldes befinden und weiter auf den Rand zu bewegen, so wird
     * diese Bewegung verworfen. Ruft am ende die drawSpaceBar()-Methode auf,
     * die die geupdatete SpaceBar zeichnet.
     *
     * @param direction
     *            Richtung, in welche sich die Spacebar bewegen soll. True steht
     *            fÃ¼r links, False fÃ¼r rechts.
     */
    private static void moveSpaceBar(boolean direction) {
        // TODO

        if(direction == false && aktuellspaceBarX < 77) { // setzt spacebarx auf 1 um
nach rechts zu laufen
            spaceBarX = 1;
            drawSpaceBar();
        }


        if(direction == true && aktuellspaceBarX > 0) {   //setzt spacebarx auf -1 um
nach links zu laufen
            spaceBarX = -1;
```

```
            drawSpaceBar();
        }


    }

    /**
     * Schiesst eine neue Bullet "^". Die neue Bullet wird direkt ueber der
     * SpaceBar ins Feld geschrieben ausser es befindet sich in dieser direkt
     * ein Alien, dann wird dieses direkt getoetet.
     */
    private static void shoot() {
        // TODO
        schuss = true;
        int x = aktuellspaceBarX;      // schießt eine bullet über dem aktuellen
spacebar feld
        Console.setForeground(Color.red);
        Console.gotoXY(x + 1, 21);
        Console.write("^");
        Console.gotoXY(x + 1, 20);
        if(Console.getChar() == '@') {
            Console.write("");
            schuss = false;
        }
        updateBullets();
    }


    /**
     * Setzt akke Bullets eine Zeile weiter nach oben, falls eine Bullet auf ein
     * Alien trifft, wird dieses getoetet.
     */
    private static void updateBullets() {
        // TODO
        Console.displayOff();
        int x = aktuellspaceBarX;
        // update previous shots
        for(int y = 20; y >= 1; --y) {
            Console.gotoXY(x, y-1);
            // skip until we get to the update
            if (Console.getChar() != '^') {
                continue;
            }

            if (Console.getChar() == '@') {
                // ziel löschen
                Console.write(" ");
                numberOfAliens--;      //wenn alien gestorben, wird 1 abgezogen
                break;

            } else {
                Console.setForeground(Color.red);
                // vorherige kugel löschen
                Console.write(" ");
                // neue kugel setzen
                Console.gotoXY(x, y);
                Console.write("^");
            }
        }

        // neuer schuss
        if(schuss == true) {
```

```
            schuss = false;
            x = aktuellspaceBarX;
            Console.gotoXY(x + 1, 20);
            Console.setForeground(Color.red);
        }

        Console.displayOn();
    }

    /**
     * Gibt das Ende des Spiels mit der Information ob es gewonnen oder verloren
     * wurde aus.
     *
     * Stoppt die beiden Timer.
     *
     * @param won
     *            Falls true ist das Spiel gewonnen, bei false ist das Spiel
     *            verloren.
     */
    private static void end(boolean won) {   //gibt den endstand aus
        alienTimer.stop();
        bulletTimer.stop();
        // TODO
        if(won == true) {
            Console.setForeground(Color.WHITE);
            Console.gotoXY(20, 5);
            Console.write("Gewonnen");
        }
        if(won == false) {
            Console.setForeground(Color.WHITE);
            Console.gotoXY(20, 5);
            Console.write("Verloren");
        }
    }
    private static void startGame() {

        while(fertig == false) {
            char c = Console.readKey();
            move(c);
            updateAliens();
        }
    }
}

//@author Maximilian Raspe
import java.util.Arrays;
import java.util.Scanner;
public class SelectionSort {

      public static void main(String[] args) {

            Scanner laenge = new Scanner(System.in);            //arraylänge einlesen
            System.out.println("Array länge eingeben");
            int length = laenge.nextInt();
            laenge.close();

            mixArray(newArray(length));

      }

      private static int[] newArray(int n) { //füllt den array mit zahlen
            int [] array = new int [n];
```

```
		for(int i = 0; i < n ; i++) {
			array[i] = i + 1;
		}
		return array;		//@return das erzeugt array
	}
	private static void mixArray(int[] array) {  //mischt den array
		int i = array.length / 2;
		int j = array.length - 1;
		while(i > 0) {		//misch den array, indem er in 2 geteilt wird und die
zahlen vorne und hiten vertauscht werden(wie beim karten mischen)
			int temp = array[i];
			array[i--] = array[j];
			array[j--] = temp;
		}
		int temp = array[0];		//damit auch die 1. zahl vertauscht wird, ist
das nochmal extra
		array[0] = array[array.length - 1];
		array[array.length - 1] = temp;

		showArray(array);
		selectionSort(array);
	}

	private static void showArray(int[] array) { //gibt den gemischten array aus

		System.out.println(Arrays.toString(array));
	}

	private static void selectionSort(int[] array) {
		int counter = 1;
		for(int i = 0; i <= array.length - 1; i++) { // array an i-ter stelle wird
hochgezählt
			for(int j = 1; j < array.length; j++) {		//array an stelle j
wird hochgezählt
				if(array[j] == counter) {		//wenn j == i + 1 ist, dann
werden die wert an j-ter und i-ter stelle vertauscht
					int temp = array[i];
					array[i] = counter;
					array[j] = temp;
					counter++;
					j = array.length;  //hilft bei der abbruchbedingung für
die innere schleife
				}
			}
		}
		System.out.println("Sortierter Array : " + Arrays.toString(array));
	}
}

//@author maximlian Raspe
import java.awt.BorderLayout;
import java.awt.Dimension;
import java.awt.Toolkit;

import javax.swing.JFrame;

/**
 * Das Spiel Game of Life
 *
 * @author Sven Loeffler
 * @date 16.11.2017
 * @version 1.0
```

```
 */
public class GameOfLife {

	private static final int FREQUENCY = 15;
	private final static double SPEED = 0.25;

	/**
	 * Die auszuführende Methode, wenn das Programm gestartet wird.
	 *
	 * @param args
	 *            Die Programmargumente.
	 */
	public static void main(String[] args) {

		JFrame frame = setFrame();
		boolean[][] cells = generateCells();

		// Schleife die das Spielfeld permanent neu zeichnet.
		while (true) {
			cells = conwaysRules(cells);
			frame.add(BorderLayout.CENTER, new MapView(cells));
			frame.setVisible(true);

			try {
				Thread.sleep((int) (SPEED * 1000));
			} catch (InterruptedException ex) {
				Thread.currentThread().interrupt();
			}
		}

	}

	/**
	 * Erzeugt ein neues Frame (Fenster) und stellt es richtig ein.
	 *
	 * @return Das erzeugte Frame.
	 */
	public static JFrame setFrame() {
		JFrame frame = new JFrame();

		Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
		Dimension frameSize = new Dimension((int) (screenSize.width), (int)
(screenSize.height));
		int frameWidth = MapView.getCELLSIZE() * 102 + 16;
		int frameHeight = MapView.getCELLSIZE() * 102 + 38;
		frame.setBounds((int) (0.5 * (frameSize.width - frameWidth)), (int) (0.5 *
(frameSize.height - frameHeight)),
				frameWidth, frameHeight);

		frame.setTitle("Game of Life");
		frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		frame.setLayout(new BorderLayout());

		return frame;
	}

	/**
	 * Gibt ein zweidimensionales (x- und y-Koordinate) boolean-Array mit
	 * zufäligem Inhalt aus, dabei entspricht true einer lebenden und false eine
	 * tote Zelle.
	 *
	 * @return Zweidimensionales Array mit lebenden und toten Zellen.
```

```
	 */
	public static boolean[][] generateCells() {
		boolean[][] cells = new boolean[102][102];

		for (int y = 1; y <= 100; y++) {
			for (int x = 1; x <= 100; x++) {
				int random = (int) (Math.random() * 10);
				if (random < (FREQUENCY / 10)) {
					cells[x][y] = true;
				} else {
					cells[x][y] = false;
				}
			}
		}

		for (int i = 0; i <= 101; i++) {
			cells[i][0] = false;
			cells[i][101] = false;
			cells[0][i] = false;
			cells[101][i] = false;
		}

		return cells;

	}

	/**
	 * Liest ein zweidimensionales Array von Zellen ein, auf dem die von Conway
	 * formulierten Regeln angewandt werden. Die aus den Regeln entstandenen
	 * toten und lebenden Zellen werden in einem neuen Array gespeichert und
	 * ausgegeben.
	 *
	 * @param cells
	 *            Das Array der Zellen, das ausgelesen wird.
	 *
	 * @return Neues zweidimensionales Array mit lebenden und toten Zellen.
	 */

	public static boolean[][] conwaysRules(boolean[][] cells) {
		boolean newCells[][] = new boolean[102][102];	// array für neue
zellen

		for(int y = 1; y <= 100; y++) {		//durchläuft den array f	r x und e
y wert
			for(int x = 1; x <= 100; x++) {

				int counterAlive = 0;	//zählt benachbarte zellen

				if(cells[x + 1][y ] == true) counterAlive++;
				if(cells[x + 1][y + 1 ] == true) counterAlive++;
				if(cells[x + 1][y - 1] == true) counterAlive++;
				if(cells[x][y + 1] == true) counterAlive++;
				if(cells[x][y - 1 ] == true) counterAlive++;
				if(cells[x - 1][y ] == true) counterAlive++;
				if(cells[x - 1][y + 1] == true) counterAlive++;
				if(cells[x - 1][y - 1] == true) counterAlive++;

				//conway regeln
				if(counterAlive == 3 && cells[x][y] == false) newCells[x][y]
= true;	// tote zelle mit 3 benachbarten zellen = lebende zelle

				if(counterAlive < 2 && cells[x][y] == true) newCells[x][y] =
```

```
false; //lebende zelle mit weniger als 2 nachbarn = tote zelle

                        if((counterAlive == 2 || counterAlive == 3) && cells[x][y] ==
true) newCells[x][y] = true;    //lebende zelle mit weniger als 2 oder 3 nachbarn =
lebende zelle

                        if(counterAlive > 3 && cells[x][y] == true) newCells[x][y] =
false; //lebende zelle mit weniger als 2 oder 3 nachbarn = lebende zelle
                        }

                    }

            return newCells;    //@return newCells
      }

}
```