```
/*
 *  Luokassa toteutetaan kaikki simulaation liittyvä
 *  piirtäminen ja hiiren tapahtumakäsittely.
 */

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class RuudukkoAlusta extends JPanel implements MouseMotionListener,
                                                      MouseListener {


  private final int SOLUNKOKO = 4;
  private boolean vasenNappi;
  private SoluRuudukko soluRuudukko;
  private boolean[][] ruudukko;
  private int viive;
  private int solujaApu;
  private int sukupolviApu;
  private Timer ajastin;
  private PainikeRivi painikeRivi;


  /*
   *  Luokan konstruktori.
   */

  public RuudukkoAlusta() {

    viive = 300;
    solujaApu = 0;
    sukupolviApu = 0;

    // Alustetaan uusi soluruudukko

    soluRuudukko = new SoluRuudukko(SOLUNKOKO);
    soluRuudukko.tyhjennä();
    ruudukko = soluRuudukko.annaRuudukko();

    addMouseListener(this);
    addMouseMotionListener(this);

    // Luodaan Timer-luokasta ilmentymä hoitamaan sukupolven
    // vaihtuminen aikavälein. Tässä käytetty sisäluokaa
    // ActionListener-rajapinnan toteuttamiseen, jotta
    // ajastimeen tarvittavat toiminnot olisivat kaikki samassa
    // paikassa.

    ajastin = new Timer(viive,
      new ActionListener() {
        public void actionPerformed(ActionEvent tapahtuma) {
              soluRuudukko.generoiSeuraavaSukupolvi();
              ruudukko = soluRuudukko.annaRuudukko();
              repaint();
        }
      }
    );

  }
```

```
/*
 *  Piirretään elävät solut ruudulle.
 */

private void piirräElävätSolut(Graphics piirtopinta, boolean[][] ruudukko) {
  Color solunVäri = new Color(136, 0, 0);
  piirtopinta.setColor(solunVäri);

  for (int y = 0; y < soluRuudukko.annaKorkeus(); y++) {
    for (int x = 0; x < soluRuudukko.annaLeveys(); x++) {
      if (ruudukko[y][x]) {
        int drawFromX = x * SOLUNKOKO;
        int drawFromY = y * SOLUNKOKO;
        piirtopinta.fillRect(drawFromX, drawFromY,
                    SOLUNKOKO, SOLUNKOKO);
      }
    }
  }

}


/*
 *  Päivitetään jokaisella piirtokerralla Sukupolvi-, ja
 *  Soluja-tekstikenttien arvot.
 */

private void päivitäSukupolviJaSolujaKentät() {
  painikeRivi.asetaSukupolvi(soluRuudukko.annaSukupolvi());
  painikeRivi.asetaSoluja(soluRuudukko.annaSolujenMäärä());
}


/*
 *  Katsotaan onko simulaatio vakiintunut toistuviin kuvioihin.
 *  Tarkistetaan solujen lukumäärä alussa, ja siitä lähtien sadan
 *  sukupolven välein. Jos peräkkäisissä sadalla jaollisissa
 *  sukupolvissa on sama määrä soluja, pysäytetään simulaatio.
 *  Muuten otetaan luvut talteen ja tarkistetaan taas sadan
 *  sukupolven päästä.
 */

private void testaaVakiintuminen() {

  // Solujen määrä alussa.

  if (soluRuudukko.annaSukupolvi() == 0) {
    sukupolviApu = soluRuudukko.annaSukupolvi(); // tietenkin nolla
    solujaApu = soluRuudukko.annaSolujenMäärä();
  }

  // Tarkistetaan solujen määrä sadan sukupolven kuluttua.
  // Jos lukujen erotus on nolla, pysäytetään simulaatio.
  // Jos erotus ei ole nolla, otetaan sukupolven numero ja
  // solujen määrä talteen ja jatketaan simulaatiota.

  if (soluRuudukko.annaSukupolvi() == sukupolviApu+100) {
    if (solujaApu - soluRuudukko.annaSolujenMäärä() == 0) {
      ajastin.stop();
    }
```

```
    else {
      solujaApu = soluRuudukko.annaSolujenMäärä();
      sukupolviApu = soluRuudukko.annaSukupolvi();
    }
  }
}


/*
 *  Piirretään ruudukon viivat.
 */

private void piirräRuudukonViivat(Graphics piirtopinta) {
  for (int i = 0; i <= 600; i += SOLUNKOKO) {
    Color viivojenVäri = new Color(190, 190, 190);
    piirtopinta.setColor(viivojenVäri);
    if (i > 400) { // piirretään juuri ikkunan kokoinen viivaruudukko
      piirtopinta.drawLine(i, 0, i, 400); // pystyviivat
    }
    else {
      piirtopinta.drawLine(0, i, 600, i); // vaakaviivat
      piirtopinta.drawLine(i, 0, i, 400); // pystyviivat
    }
  }
}


/*
 *  Käynnistetään ajastin.
 */

public void käynnistäAjastin() {
  ajastin.start();
}


/*
 *  Pysäytetään ajastin.
 */

public void pysäytäAjastin() {
  ajastin.stop();
}


/*
 *  Tyhjennetään ruudukko.
 */

public void tyhjennä() {
  soluRuudukko.tyhjennä();
  repaint();
}


/*
 *  Arvotaan uusi tilanne ruudukkoon.
 */

public void arvo() {
  soluRuudukko.arvoRuudukko();
```

```
  repaint();
}


/*
 *  Edettään seuraavaan sukupolveen.
 */

public void seuraavaSukupolvi() {
  soluRuudukko.generoiSeuraavaSukupolvi();
  ruudukko = soluRuudukko.annaRuudukko();
  repaint();
}


/*
 *  Asetetaan ohjelman käyttämä PainikeRivi-luokan
 *  ilmentymä, jotta voidaan päivittää Soluja ja
 *  Sukupolvi-kentät painikerivissä.
 */

public void asetaPainikeRivi(PainikeRivi painikeRivi) {
  this.painikeRivi = painikeRivi;
}


/*
 *  Asetetaan ajastimen viive millisekunneissa.
 */

public void asetaViive(int viive) {
  ajastin.setDelay(viive);
  repaint();
}


/*
 *  Piirtämisrutiinien määrittely
 */

public void paintComponent(Graphics piirtopinta) {
  super.paintComponent(piirtopinta);
  Color taustaVäri = new Color(180, 180, 180);
  setBackground(taustaVäri);

  piirräElävätSolut(piirtopinta, ruudukko);

  päivitäSukupolviJaSolujaKentät();

  testaaVakiintuminen();

  piirräRuudukonViivat(piirtopinta);
}


/*
 *  MouseMotionListener ja MouseListener-rajapintaluokkien
 *  vaatimien metodien toteutus.
 */

public void mousePressed(MouseEvent tapahtuma) {
```

```
      int x = tapahtuma.getX();
      int y = tapahtuma.getY();

      vasenNappi = SwingUtilities.isLeftMouseButton(tapahtuma);
      soluRuudukko.asetaSolunTila(x, y, SOLUNKOKO, vasenNappi);
      repaint();
    }


    public void mouseClicked(MouseEvent tapahtuma) {
      int x = tapahtuma.getX();
      int y = tapahtuma.getY();
      soluRuudukko.asetaSolunTila(x, y, SOLUNKOKO, vasenNappi);
      repaint();
    }


    public void mouseDragged(MouseEvent tapahtuma) {
      int x = tapahtuma.getX();
      int y = tapahtuma.getY();
      soluRuudukko.asetaSolunTila(x, y, SOLUNKOKO, vasenNappi);
      repaint();
    }


    /*
     *  Tyhjät metodit, jotka on pakko korvata.
     */

    public void mouseMoved(MouseEvent tapahtuma) {}

    public void mouseReleased(MouseEvent tapahtuma) {}

    public void mouseEntered(MouseEvent tapahtuma) {}

    public void mouseExited(MouseEvent tapahtuma) {}

}
```