```
/*
 *  Luokassa toteutaan ohjelman tietorakenne ja sen toiminta.
 */

public class SoluRuudukko {

  private final int LEVEYS;
  private final int KORKEUS;
  private final boolean ELOSSA = true;
  private final boolean KUOLLUT = false;
  private boolean[][] ruudukko;
  private int sukupolvi = 0;
  private int solujenMäärä = 0;


  /*
   *  Luokan konstruktori.
   */

  public SoluRuudukko(int solunKoko) {

    LEVEYS = 600 / solunKoko;
    KORKEUS = 400 / solunKoko;
    ruudukko = new boolean[KORKEUS][LEVEYS];

  }


  /*
   *  Lasketaan naapurit käsittelyssä olevalle solulle.
   */

  private int laskeNaapurit(boolean[][] ruudukko, int y, int x) {

    int naapurit = 0;
    int vasen, oikea, ylä, ala;

    // Otetaan talteen alkion ylä-, ala-, vasen- ja oikeanpuoleiset
    // paikat.

    ylä = y-1;
    if (ylä == -1)
      ylä = KORKEUS-1;

    ala = y+1;
    if (ala == KORKEUS)
      ala = 0;

    vasen = x-1;
    if (vasen == -1)
      vasen = LEVEYS-1;

    oikea = x+1;
    if (oikea == LEVEYS)
      oikea = 0;

    // indeksoidaan taulukkoa yllä olevilla muuttujilla.

    if (ruudukko[ylä][vasen] == ELOSSA) naapurit++;
    if (ruudukko[y][vasen] == ELOSSA) naapurit++;
    if (ruudukko[ala][vasen] == ELOSSA) naapurit++;
```

```
  if (ruudukko[ylä][x] == ELOSSA) naapurit++;
  if (ruudukko[ala][x] == ELOSSA) naapurit++;
  if (ruudukko[ylä][oikea] == ELOSSA) naapurit++;
  if (ruudukko[y][oikea] == ELOSSA) naapurit++;
  if (ruudukko[ala][oikea] == ELOSSA) naapurit++;

  return naapurit;
}


/*
 *  Palautetaan sukupolven numero.
 */

public int annaSukupolvi() {
  return sukupolvi;
}


/*
 *  Palautetaan solujen lukumäärä.
 */

public int annaSolujenMäärä() {
  return solujenMäärä;
}


/*
 *  Palautetaan ruudukko.
 */

public boolean[][] annaRuudukko() {
  return this.ruudukko;
}


/*
 *  Palautetaan solujen lukumäärä vaakasuunnassa.
 */

public int annaLeveys() {
  return LEVEYS;
}


/*
 *  Palautetaan solujen lukumärää pystysuunnassa.
 */

public int annaKorkeus() {
  return KORKEUS;
}


/*
 *  Täytetään ruudukko satunnaisesti elävillä ja
 *  kuolleilla soluilla.
 */

public void arvoRuudukko() {
```

```
    solujenMäärä = 0;

    for (int y = 0; y < KORKEUS; y++)
      for (int x = 0; x < LEVEYS; x++) {
        int arpa = (int)(Math.random()*2);
        if (arpa == 1) {
          this.ruudukko[y][x] = ELOSSA;
          solujenMäärä++;
        }
        else
          this.ruudukko[y][x] = KUOLLUT;
      }

    sukupolvi = 0;
  }


  /*
   *  Tyhjennetään ruudukko.
   */

  public void tyhjennä() {

    for (int y = 0; y < KORKEUS; y++)
      for (int x = 0; x < LEVEYS; x++)
        this.ruudukko[y][x] = KUOLLUT;

    sukupolvi = 0;
    solujenMäärä = 0;
  }


  /*
   *  Asetetaan x:n ja y:n osoittaman solun tila riippuen siitä,
   *  kumpaa hiiren nappia on painettu.
   */

  public void asetaSolunTila(int x, int y, int solunKoko, boolean
onkoVasenNappi) {
    x = x / solunKoko;
    y = y / solunKoko;

    // Jos hiiren kursori on soluruudukon alueella, tarkistetaan kumpaa
    // nappia on painettu ja vaihdetaan solun tila tarvittaessa.

    if( x>=0 && x<LEVEYS && y>=0 && y<KORKEUS ) {
      if (onkoVasenNappi == true && ruudukko[y][x] == KUOLLUT) {
        ruudukko[y][x] = ELOSSA;
        solujenMäärä++;
      }
      else if (onkoVasenNappi == false && ruudukko[y][x] == ELOSSA) {
        ruudukko[y][x] = KUOLLUT;
        solujenMäärä--;
      }
    }
  }


  /*
   *  Muodostetaan seuraava sukupolvi.
   */
```

```
  public void generoiSeuraavaSukupolvi() {

    boolean[][] seuraavaSukupolvi = new boolean[KORKEUS][LEVEYS];
    int naapureita;  // solun naapurien määrä

    // Käydään läpi kaikki solut, lasketaan naapurien määrä
    // joka solulle.

    for (int y = 0; y < KORKEUS; y++) {
      for (int x = 0; x < LEVEYS; x++) {

        naapureita = laskeNaapurit(ruudukko, y, x);

        // Jos solu on elossa, katsotaan kuoleeko se.
        // Jos solu on kuollut, katsotaan syntyykö se uudestaan.

        if (this.ruudukko[y][x] == ELOSSA) { // Jos solu elossa...
          if (naapureita == 2 || naapureita == 3) // ...solu pysyy elossa.
            seuraavaSukupolvi[y][x] = ELOSSA;
          else {// ...solu kuolee.
            seuraavaSukupolvi[y][x] = KUOLLUT;
            solujenMäärä--;
          }
        }
        else { // Jos solu kuollut...
          if (naapureita == 3) { // ...solu syntyy uudestaan.
            seuraavaSukupolvi[y][x] = ELOSSA;
            solujenMäärä++;
          }
          else // ...solu pysyy kuolleena.
            seuraavaSukupolvi[y][x] = KUOLLUT;
        }

      }
    }

    sukupolvi++;

    // Vaihdetaan ohjelman nykyinen ruudukko seuraavaan sukupolveen.

    ruudukko = seuraavaSukupolvi;
  }


}
```