Turun yliopisto
University of Turku

Olio-ohjelmoinnin metodiikka
Viikkoharjoitustyö 2

Ryhmä:
Pasi Toivanen (517487)
Janina Kuosmanen (516580)
Santeri Loitomaa (516587)
Tommi Heikkinen (517749)

Viikkoharjoitustyö 2
29. syyskuuta 2018
?? sivua

Turun yliopisto
Tulevaisuuden teknologioiden laitos
Tietotekniikka
Olio-ohjelmoinnin metodiikka

Sisältö

1 Tehtävä 1

Tehtävässä tutustutaan oom-kit -kehykseen ja siitä löytyviin luokkiin

1.1 fi.utu.oomkit -luokat

fi.utu.oomkit.gui -paketista löytyy luokat:

```
1  public interface Console {}
2  class DefaultDialogFactory implements DialogFactory {}
3  public interface DialogFactory {}
4  class FXConsole extends OutputStream implements Console {}
5  public interface MainWindow extends WindowContent {}
6  public abstract class MainWindowBase implements MainWindow {}
7  class MergedStream extends OutputStream {}
8  public abstract class OOMApp extends Application {}
9  public class ReactiveCanvas<X> extends SimpleCanvas implements Observer<X> {}
10 public class ReactiveLabel<T> extends Label implements Observer<T> {
11         public interface LabelHandler<T> {}
12 }
13 public class SimpleCanvas extends Canvas {}
14 class StreamWrapper extends OutputStream {}
15 public interface WindowContent {}
```

Joista kaikki ovat julkisia paketin sisällä, mutta paketin ulkopuolella näkyvistä on jätetty luokat DefaultDialogFactory, FXConsole, MergedStream ja StreamWrapper. fi.utu.oomkit -paketista löytyy luokat:

```
1  public final class AppConfiguration {}
2  public interface AppLogic extends KeyHandler, Scheduled {}
```

Joista kaikki ovat julkisia luokkia niin paketin sisällä kuin ulkopuolella.

1.2 OOMApp -luokka sovelluksen ikkunan luomiseksi

OOMApp -luokka peritään ja sille kirjoitetaan generateMainWindow -metodi.

GenerateMainWindow -metodilla määritellään mikä on ohjelman (=app) nimi ja kuinka ison alueen ohjelma käyttää. Metodin tulee palauttaa MainWindow -alaluokka. OOMApp -luokan käyttöä on esitetty DemoApp1 -tutoriaalissa seuraavasti:

```
1  public class DemoApp1 extends OOMApp {
2      // alustaa pelilogiikan
3      final static LaatikkoGameLogic gameLogic = new LaatikkoGameLogic();
4
5      // kytkee piirtopinnan käyttöliittymään
6      @Override
7      protected MainWindow generateMainWindow(String appName, double width,
8      double height) { return new SimpleMainWindow(appName, width, height) {
9              @Override public SimpleCanvas mainContent() {
10                 return gameLogic.piirtoPinta;
11             }
12         };
13     }
14 }
```

EmptyApp -luokassa generateMainWindow -metodille on tehty mielekäs esimerkki-implementaatio, jolloin sitä voi käyttää sellaisenaan muodostaessa ohjelman ikkunaa. Siitä löytyy valmiiksi ajettava main-metodi.

1.3 Ohjelman ikkunan sisällön hallitseminen

Ikkuna luodaan käyttämällä luokkaa MainWindow. Huomionarvoista on, että MainWindow -olion sisältö koostuu ylä- ja alapalkista, sekä Canvas-piirtoalueesta. Oletuksena nämä ovat tyhjiä, jolloin oman sisällön voi luoda ylikirjoittamalla metodit topBarContent(), mainContent(), bottomBarContent().

```
1  public class EmptyApp extends OOMApp {
2          @Override
3          protected MainWindow generateMainWindow(){
```

```
4                 return new SimpleMainWindow(appName, width, height) {
5
6                         @Override
7                         public List<Node> topBarContent() {
8                                 //return yläpalkin sisältö
9                         }
10
11                        @Override
12                        public Canvas mainContent() {
13                                //return pääsisältö
14                        }
15
16                        @Override
17                        public List<Node> bottomBarContent() {
18                                //return alapalkin sisältö
19                        }
20                }
21        }
22}
```

Ylä- ja alapalkkiin voidaan lisätä mitä tahansa graafisia solmuja, jotka voivat olla erilaisia graafisia käyttöliittymäkomponentteja. Ylikirjoittamalla topBarContent() -metodin voidaan sijoittaa käyttöliittymään lista mitä tahansa elementtejä kuten esimerkiksi luokkien Canvas, ImageView, Shape tai Button luokkien oliot. Yläpalkin lisäämisessä voitaisiin käyttää esimerkiksi seuraavaa metodia:

```
1 @Override
2 public List<Node> topBarContent() {
3         return Arrays.asList(
4                 new Button("File"),
5                 new Button("Edit"),
6                 new Button("Window"),
7                 new Button("Help")
8         );
9 }
```

Tämä lisää työkalurivin kaltaiset painikkeet yläpalkkiin

1.4 Ohjelmanaikaiset tapahtumat

OOMApp -luokan olion konstruktorilla pystytään luomaan useanlaisia logiikoita ohjelman suoritukseksi. Oma logiikka voidaan lisätä käyttämällä OOMApp konstruktoria, johon syötetään ohjelmalogiikka. Ohjelmalogiikkaluokka peritään AppLogic -luokalta. Seuraavassa esimerkissä on esitetty, kuinka MyLogic -luokalla voidaan hallita ohjelmalogiikkaa niin, että asiakkaan koodi suoritetaan joka 25ms.

```
1  public class MyApp extends OOMApp {
2          public MyApp() {
3                  super(new MyLogic());
4          }
5  }
6
7  public class MyLogic extends AppLogic {
8          @override
9          AppConfiguration configuration() {
10                 return new AppConfiguration(25, "Asiakkaan ohjelma", true);
11         }
12
13         @override
14         void tick() {
15                 //asiakkaan koodi
16         }
17 }
```

- Pasi Toivanen (517487)

2 Tehtävä 2

2.1 a-kohta

Luokka lopputeksti sisältää intit rivimäärällä ja max merkkimäärän, sen lisäksi String Arrayn jossa on kaikki kirjoitettava teksti. Luokassa on erikseen metodi yksittäisten rivien merkkimäärän tarkistukseen ja koko lopputeksti olion tarkastukseen. Näiden lisäksi on mukana metodit tekstin lukemiseen ja muokkaamiseen.

```
1  public class Lopputeksti {
2      int rivimäärä;
3      int merkkimäärä;
4      String[] sisältö;
5
6      /**
7       * Luo lopputeksti luokan ja testaa onko sen sisältämä teksti liian pitkää yms.
8         ilmoittaen siitä jos näin on.
9       * @.pre r = int, m = int, s String[]
10      * @.post new Lopputeksti (&& Throw IndexOutOfBoundsException & virheilmoitus)
11      */
12     public Lopputeksti(int r, int m, String[] s) {
13       this.rivimäärä=r;
14       this.merkkimäärä=m;
15       this.sisältö=s;
16       rajoitteet(r,m,s);
17     }
18     /**
19      * Tarkistaa ja ilmoittaa onko lopputeksti haluttujen rajojen mukaista.
20      * @.pre r = int, m = int, s String[]
21      * @.post (IndexOutOfBoundsException & Ilmoitus virheestä)
22      */
```

```
23    public void rajoitteet(int rm, int mm, String[] syöte) {
24      for (int x=0;x<syöte.length;x++) {
25        if (riviTarkistus(syöte[x],mm) == false) {
26          try {
27            throw new IndexOutOfBoundsException();
28          } catch (IndexOutOfBoundsException i) {
29              System.out.println("Rivillä " + x+1 + ". on liian monta merkkiä!");
30          }
31        }
32      }
33      if (rm<syöte.length) {
34        try {
35          throw new IndexOutOfBoundsException();
36        } catch (IndexOutOfBoundsException i) {
37            System.out.println("Rivejä on liikaa!");
38        }
39      }
40    }
41    /**
42     * Tarkistaa rivin merkkimäärän, verraten sitä sallittuun maksimiin
43     * @p.pre tarkistettava = String, maksimi = int
44     * @.post return true || false
45     */
46    public boolean riviTarkistus(String tarkistettava, int maksimi) {
47      if (tarkistettava.length()>maksimi) {
48        return false;
49      }
50      return true;
51    }
52    /**
53     * Muokkaa annetun rivin tekstiä
54     * @.pre rivi = int, uusi = String
55     * @.post sisältö[rivi] = uusi;
56     */
57    public void tekstinMuokkaus(int rivi, String uusi ) {
58      sisältö[rivi] = uusi;
59    }
60    /**
61     * Palauttaa annetun rivin
62     * @.pre rivi = int
63     * @.post return sisältö[rivi];
```

```
64       */
65     public String tekstinLuku(int rivi) {
66       return sisältö[rivi];
67     }
68   }
```

2.2 b-kohta

Luokkaan Lopputeksti lisättiin seuraavat kaksi metodia

```
1  /**
2       * Siirtää piirrettävän tekstin y koordinaattia ja kutsuu komentoa piirrärivi
3       * @.pre true
4       * @.post Point c = new Point (p.x,(p.y + i*size)); &&
5       *         piirräRivi(piirtopinta, canvas,c,size, i);
6       */
7     public void piirräTeksti(TekstiPiirtopinta piirtopinta, GraphicsContext canvas, Point p, int size
8       for (int i = 0; i<sisältö.length;i++) {
9         Point c = new Point (p.x,(p.y + i*size));
10        piirräRivi(piirtopinta, canvas,c,size, i);
11      }
12
13    }
14    /**
15     * Käy annetun tekstin kirjain kirjaimelta ja muuttaa x koordinaattia sopivasti,
16       kutsuen sitten drawText metodia.
17     * @.pre true
18     * @.post Point b = new Point ((p.x + i*size), p.y);
19          piirtopinta.drawText(canvas, b, l, size);
20     */
21    private void piirräRivi(TekstiPiirtopinta piirtopinta, GraphicsContext canvas, Point p, int size,
22      for (int i=0; i<sisältö[row].length();i++) {
23        char l = sisältö[row].charAt(i);
24        Point b = new Point ((p.x + i*size), p.y);
25        piirtopinta.drawText(canvas, b, l, size);
26      }
27    }
```

Joiden avulla kutsutaan TextDraw metodia tekstin kirjoittamiseksi. Tämä testattiin De-

moApp2 sisällä, luomalla testiolio Tekstipiirtopinnassa ja kutsumalla sitä drawForeGrounddContent sisältä. Lopputuloksena oli valmis teksti ruudulla.

```
Point a = new Point(50,50);
testi.piirräTeksti(this, canvas, a, koko);
```

2.3 c-kohta

Metodi piirräKirjainKerralla piirtää lopputeksti olion kirjaimia yksi kerrallaan. Ja lopettaa piirtämästä kun koko lopputeksti on päästy loppuun.

```
/**
     * Metodi ottaa annetuilla luvuilla kirjaimia lopputeksteistä ja kirjoittaa ne niille kuuluville
     * @.pre true
     * @.post piirtopinta.drawText(canvas, point, letter, size);
     */
    public void piirräKirjainKerralla(TekstiPiirtopinta piirtopinta, GraphicsContext canvas, int t, i
      int x = 0;
      int a = t;
      boolean kill = false;
      while (a>=sisältö[x].length()) {
        a= a-sisältö[x].length();
        x++;
        if (x>=sisältö.length) {
          kill = true;
          break;
        }
      }
      if (kill == false) {
        char letter = sisältö[x].charAt(a);
        Point point = new Point ((a*size+50),(x*size+50));
        piirtopinta.drawText(canvas, point, letter, size);
      }

    }
```

int t määritellään tickissä ja se kasvaa jokaisella tickin kutsulla, täten se käy lopputekstit läpi järjestyksessä. Olio itsessään luotiin tekstiGameLogicissa ja kutsuttiin tickissä.

Luonti textGameLogicissa

```
1  private int i = 0;
2    String[] joku = new String[] {"moi","moi taas","abcdefghijklmnopqrs"};
3    Lopputeksti testi = new Lopputeksti(25,10,joku);
```

kutsu Tickissä

```
1 testi.piirräKirjainKerralla(piirtoPinta, piirtoPinta.getGraphicsContext2D(), i, 24);
```

Kussakin "sarakkessa"ts. Stringissä on vain riville kirjoitettavat merkit, joten niissä ei ole ylimääräistä, toki jos rivin suorituksen haluaa kestävän kauemmin voi rivin loppuun lisätä välilyöntejä.

2.4 d-kohta

Aiemman tehtävänannon metodi ei muutu, tickissä vain lineaarisesti kasvavan i:n sijasta arvotaan se Randomia hyväksikäyttäen. Antane lopputekstin kirjainten tulla satunnaisessa järjestyksessä.

```
1  Random r = new Random();
2      i = r.nextInt(30);
```

Tekijänä Tommi Heikkinen (517749)

3 Tehtävä 3

3.1 a-kohta

Luokka lopputeksti sisältää intit rivimäärällä ja max merkkimäärän, sen lisäksi String Arrayn jossa on kaikki kirjoitettava teksti. Luokassa on erikseen metodi yksittäisten rivien merkkimäärän tarkistukseen ja koko lopputeksti olion tarkastukseen. Näiden lisäksi on mukana metodit tekstin lukemiseen ja muokkaamiseen.

```
1  public class Lopputeksti {
2      int rivimäärä;
3      int merkkimäärä;
4      String[] sisältö;
5
6      /**
7       * Luo lopputeksti luokan ja testaa onko sen sisältämä teksti liian pitkää yms.
8         ilmoittaen siitä jos näin on.
9       * @.pre r = int, m = int, s String[]
10      * @.post new Lopputeksti (&& Throw IndexOutOfBoundsException & virheilmoitus)
11      */
12     public Lopputeksti(int r, int m, String[] s) {
13       this.rivimäärä=r;
14       this.merkkimäärä=m;
15       this.sisältö=s;
16       rajoitteet(r,m,s);
17     }
18     /**
19      * Tarkistaa ja ilmoittaa onko lopputeksti haluttujen rajojen mukaista.
20      * @.pre r = int, m = int, s String[]
21      * @.post (IndexOutOfBoundsException & Ilmoitus virheestä)
22      */
```

```
23    public void rajoitteet(int rm, int mm, String[] syöte) {
24      for (int x=0;x<syöte.length;x++) {
25        if (riviTarkistus(syöte[x],mm) == false) {
26          try {
27            throw new IndexOutOfBoundsException();
28          } catch (IndexOutOfBoundsException i) {
29              System.out.println("Rivillä " + x+1 + ". on liian monta merkkiä!");
30          }
31        }
32      }
33      if (rm<syöte.length) {
34        try {
35          throw new IndexOutOfBoundsException();
36        } catch (IndexOutOfBoundsException i) {
37            System.out.println("Rivejä on liikaa!");
38        }
39      }
40    }
41    /**
42     * Tarkistaa rivin merkkimäärän, verraten sitä sallittuun maksimiin
43     * @p.pre tarkistettava = String, maksimi = int
44     * @.post return true || false
45     */
46    public boolean riviTarkistus(String tarkistettava, int maksimi) {
47      if (tarkistettava.length()>maksimi) {
48        return false;
49      }
50      return true;
51    }
52    /**
53     * Muokkaa annetun rivin tekstiä
54     * @.pre rivi = int, uusi = String
55     * @.post sisältö[rivi] = uusi;
56     */
57    public void tekstinMuokkaus(int rivi, String uusi ) {
58      sisältö[rivi] = uusi;
59    }
60    /**
61     * Palauttaa annetun rivin
62     * @.pre rivi = int
63     * @.post return sisältö[rivi];
```

```
64       */
65     public String tekstinLuku(int rivi) {
66       return sisältö[rivi];
67     }
68   }
```

3.2 b-kohta

Luokkaan Lopputeksti lisättiin seuraavat kaksi metodia

```
1  /**
2       * Siirtää piirrettävän tekstin y koordinaattia ja kutsuu komentoa piirrärivi
3       * @.pre true
4       * @.post Point c = new Point (p.x,(p.y + i*size)); &&
5       *         piirräRivi(piirtopinta, canvas,c,size, i);
6       */
7     public void piirräTeksti(TekstiPiirtopinta piirtopinta, GraphicsContext canvas, Point p, int size
8       for (int i = 0; i<sisältö.length;i++) {
9         Point c = new Point (p.x,(p.y + i*size));
10        piirräRivi(piirtopinta, canvas,c,size, i);
11      }
12
13    }
14    /**
15     * Käy annetun tekstin kirjain kirjaimelta ja muuttaa x koordinaattia sopivasti,
16       kutsuen sitten drawText metodia.
17     * @.pre true
18     * @.post Point b = new Point ((p.x + i*size), p.y);
19          piirtopinta.drawText(canvas, b, l, size);
20     */
21    private void piirräRivi(TekstiPiirtopinta piirtopinta, GraphicsContext canvas, Point p, int size,
22      for (int i=0; i<sisältö[row].length();i++) {
23        char l = sisältö[row].charAt(i);
24        Point b = new Point ((p.x + i*size), p.y);
25        piirtopinta.drawText(canvas, b, l, size);
26      }
27    }
```

Joiden avulla kutsutaan TextDraw metodia tekstin kirjoittamiseksi. Tämä testattiin De-

moApp2 sisällä, luomalla testiolio Tekstipiirtopinnassa ja kutsumalla sitä drawForeGrounddContent sisältä. Lopputuloksena oli valmis teksti ruudulla.

```
Point a = new Point(50,50);
testi.piirräTeksti(this, canvas, a, koko);
```

3.3 c-kohta

Metodi piirräKirjainKerralla piirtää lopputeksti olion kirjaimia yksi kerrallaan. Ja lopettaa piirtämästä kun koko lopputeksti on päästy loppuun.

```
/**
     * Metodi ottaa annetuilla luvuilla kirjaimia lopputeksteistä ja kirjoittaa ne niille kuuluville
     * @.pre true
     * @.post piirtopinta.drawText(canvas, point, letter, size);
     */
    public void piirräKirjainKerralla(TekstiPiirtopinta piirtopinta, GraphicsContext canvas, int t, i
      int x = 0;
      int a = t;
      boolean kill = false;
      while (a>=sisältö[x].length()) {
        a= a-sisältö[x].length();
        x++;
        if (x>=sisältö.length) {
          kill = true;
          break;
        }
      }
      if (kill == false) {
        char letter = sisältö[x].charAt(a);
        Point point = new Point ((a*size+50),(x*size+50));
        piirtopinta.drawText(canvas, point, letter, size);
      }

    }
```

int t määritellään tickissä ja se kasvaa jokaisella tickin kutsulla, täten se käy lopputekstit läpi järjestyksessä. Olio itsessään luotiin tekstiGameLogicissa ja kutsuttiin tickissä.

Luonti textGameLogicissa

```
1  private int i = 0;
2    String[] joku = new String[] {"moi","moi taas","abcdefghijklmnopqrs"};
3    Lopputeksti testi = new Lopputeksti(25,10,joku);
```

kutsu Tickissä

```
1 testi.piirräKirjainKerralla(piirtoPinta, piirtoPinta.getGraphicsContext2D(), i, 24);
```

Kussakin "sarakkessa"ts. Stringissä on vain riville kirjoitettavat merkit, joten niissä ei ole ylimääräistä, toki jos rivin suorituksen haluaa kestävän kauemmin voi rivin loppuun lisätä välilyöntejä.

3.4 d-kohta

Aiemman tehtävänannon metodi ei muutu, tickissä vain lineaarisesti kasvavan i:n sijasta arvotaan se Randomia hyväksikäyttäen. Antane lopputekstin kirjainten tulla satunnaisessa järjestyksessä.

```
1  Random r = new Random();
2      i = r.nextInt(30);
```

Tekijänä Tommi Heikkinen (517749)

4 Tehtävä 4

Tehtävän tarkoituksena on luoda lukitusmenetelmä OOMKit-ohjelmiin, joka estää kaiken näkyvyyden ohjelmaan kunnes ohjelman avaava koodi syötetään täysin oikein. Aikarajaa salasanan syöttöön ei ole.

4.1 Lukitus-luokka (kohdat a ja b)

Kohdassa a tuli määrittää luokka Lukitus, jolla voisi lukita OOMKit-ohjelman. Kohdassa b se taas tuli toteuttaa. Kirjaan molemmat kohdat tähän samaan osioon.

Lukitus-luokka

```
1  /**
2   * @author Santeri Loitomaa
3   */
4  public class Lukitus {
5    private ArrayList<Key> codeLukituskoodi;
6    private ArrayList<Key> yritysLukituskoodi;
7    private boolean lukossa;
8    /**
9     * Konstruktori lukolle.
10    * @param codeLukituskoodi
11    */
12   public Lukitus(ArrayList<Key> codeLukituskoodi) {
13     this.codeLukituskoodi = codeLukituskoodi;
14     this.yritysLukituskoodi = new ArrayList<Key>();
15     this.lukossa = false;
16   }
```

```
17  /**
18   * Asettaa boolean lukossa-arvon oikeaksi.
19   * @.pre Tätä metodia tulee kutsua vain jos ohjelma on lukossa.
20   * @.post   if(merkki == oikein) lukossa = true;
21   *       if(merkki != oikein) lukossa = true;
22   *       (kun kaikki merkit oikein) lukossa = false;
23   * @param Key merkki
24   */
25  public void tarkista(Key merkki){
26    yritysLukituskoodi.add(merkki);
27    if(codeLukituskoodi.equals(new ArrayList<Key>())) {
28      yritysLukituskoodi = new ArrayList<Key>();
29      lukossa = false;
30    }
31    else if(codeLukituskoodi.get(yritysLukituskoodi.size()-1) !=
32        yritysLukituskoodi.get(yritysLukituskoodi.size()-1)) {
33      System.out.println("Nyt meni väärin. Aloita alusta.");
34      yritysLukituskoodi = new ArrayList<Key>();
35      lukossa = true;
36    }
37    else if(codeLukituskoodi.size() == yritysLukituskoodi.size()) {
38      System.out.println("Koodi on syötetty oikein.");
39      yritysLukituskoodi = new ArrayList<Key>();
40      lukossa = false;
41    }
42  }
43  /**
44   * Kertoo onko lukko lukossa.
45   * @return true jos lukossa.
46   */
47  public boolean onLukossa() {
48    return lukossa;
49  }
50  /**
51   * Merkitsee lukon lukituksi.
52   * @.post lukossa = true;
53   */
54  public void lukitse() {
55    lukossa = true;
56  }
57  /**
```

```
58      * Vaihtaa salasanan.
59      * @.post codeLukituskoodi = codeLukituskoodi
60      * @param codeLukituskoodi
61      */
62    public void setCodeLukituskoodi(ArrayList<Key> codeLukituskoodi) {
63      this.codeLukituskoodi = codeLukituskoodi;
64    }
65    /**
66      * Kertoo nykyisen lukituskoodin.
67      * @return codeLukituskoodi
68      */
69    public ArrayList<Key> getCodeLukituskoodi(){
70      return codeLukituskoodi;
71    }
72  }
```

Tulin tällaisen lopputulokseen. Tämä toimii melko hyvin lisättynä OOMKitin DemoApp3:een pienten muutosten kera.

4.2 DemoApp3-luokka (kohta c)

DemoApp3:een tein seuraavat muutokset jotta Lukitus-luokkaa voitaisiin käyttää hyvin.

```
1  class LaatikkoGameLogic2 implements AppLogic {
2    ...
3    private Lukitus lukko = new Lukitus(new ArrayList<Key>());
4    private boolean vaihdetaankoSalasana = false;
5    private boolean vaihdetaanSalasanaa = false;
6    private ArrayList<Key> uusiSalasana = new ArrayList<Key>();
7    private boolean lukitaanko = false;
8    ...
9    @Override
10   public void tick() {
11     ...
12     // piilottaa näkymän jos lukossa
13     piirtoPinta.asetaPiilotus(lukko.onLukossa());
14     ...
15   }
16   // käsittele näppäimen painallus
17   @Override
18   public void handleKey(Key k) {
19     System.out.println(k);
20     if(lukko.onLukossa()) {
21       lukko.tarkista(k);
22     }
23     else if(vaihdetaanSalasanaa) {
24       uusiSalasana.add(k);
25       if(k == Key.Backspace) {
26         uusiSalasana = new ArrayList<Key>();
27         System.out.println("Salasanan␣vaihto␣peruutettu.");
28         vaihdetaanSalasanaa = false;
29         vaihdetaankoSalasana = false;
30       }
31       if(k == Key.Enter) {
32         lukko.setCodeLukituskoodi(uusiSalasana);
33         uusiSalasana = new ArrayList<Key>();
34         System.out.println("Salasana␣vaihdettu.");
35         vaihdetaanSalasanaa = false;
36         vaihdetaankoSalasana = false;
```

```
37          }
38      }
39      else if(lukitaanko && k == Key.N) {
40        lukitaanko = false;
41        System.out.println("Lukitus peruutettiin.");
42      }
43      else if(lukitaanko && k == Key.Y) {
44        lukitaanko = false;
45        lukko.lukitse();
46        System.out.println("Lukittu.");
47      }
48      else if(vaihdetaankoSalasana && k == Key.N) {
49        vaihdetaankoSalasana = false;
50      }
51      else if(vaihdetaankoSalasana && k == Key.Y) {
52        vaihdetaanSalasanaa = true;
53        System.out.println("Anna uusi salasana merkki kerrallaan.");
54        System.out.println("Tallenna painamalla Enter.");
55        System.out.println("Peruuta painamalla Backspace.");
56      }
57      else if(k == Key.Space) {
58        System.out.println("Haluatko lukita ohjelman? Y/N");
59        lukitaanko = true;
60      }
61      else if(k == Key.Backspace) {
62        System.out.println("Nykyinen salasana:");
63        System.out.println(lukko.getCodeLukituskoodi());
64      }
65      else if(k == Key.Enter) {
66        System.out.println("Haluatko asettaa uuden salasanan? Y/N");
67        vaihdetaankoSalasana = true;
68      }
69    }
70  }
```

Lukitusta voidaan hallita handleKey(Key k)-metodilla.

4.3 Testi-luokka (kohta d)

En juuri tiedä, miten testiluokka tulisi tehdä Jqwikilla, joten teen sen JUnitilla. Ensin testataan oikealla salasanalla ja sitten uudelleen lukittuna väärällä.

```
1  class Testi {
2    static Lukitus lukko;
3    static ArrayList<Key> salasana = new ArrayList<Key>();
4    @BeforeAll
5    static void setUpBeforeClass() throws Exception {
6      salasana.add(Key.T);
7      salasana.add(Key.E);
8      salasana.add(Key.S);
9      salasana.add(Key.T);
10     salasana.add(Key.I);
11     lukko = new Lukitus(salasana);
12     lukko.lukitse();
13   }
14   @Test
15   void test() {
16     lukko.tarkista(Key.T);
17     lukko.tarkista(Key.E);
18     lukko.tarkista(Key.S);
19     lukko.tarkista(Key.T);
20     lukko.tarkista(Key.I);
21     if(lukko.onLukossa()) {
22       fail("Salasana␣ei␣toiminut");
23     }
24     lukko.lukitse();
25     lukko.tarkista(Key.T);
26     lukko.tarkista(Key.E);
27     lukko.tarkista(Key.S);
28     lukko.tarkista(Key.Y);
29     lukko.tarkista(Key.I);
30     if(!lukko.onLukossa()) {
31       fail("Salasana␣toimi␣vaikkei␣pitänyt");
32     }
33   }
34 }
```

Testi ei tuottanut yhtään virhettä ja konsolin tulosteet näyttävät oikealta.

```
1 Koodi on syötetty oikein.
2 Nyt meni väärin. Aloita alusta.
3 Nyt meni väärin. Aloita alusta.
```

Tekijänä Santeri Loitomaa (516587)