Rinnakkaisohjelmointi kurssi

Opintopiiri työskentelyn raportti

Opintopiiri: Heikki Karimo, Jesse Paakkari ja Keijo Karhu
Päiväys: 15.12.2006

Ohjelmointitehtävä C i

C i : Säikeet ja kriittisen vaiheen kontrollointi niiden käytön yhteydessä

a. Menetelmän käyttötarkoitus ja sovellusalue
Java ohjelmointikieli tukee samanaikaisuutta säikeiden avulla. Javassa on määritelty luokka **Tread** ja rajapinta **Runnable**, joiden avulla samanaikaisuuden voi toteuttaa. Säikeiden käyttötarkoitus on suorittaa samanaikaisesti useita tehtäviä ohjelmassa. Sovellusalueena on esimerkiksi verkkoyhteyden toteutus omassa säikeessään, jotta käyttöliittymä olisi käytettävissä samaan aikaan. Kriittisen vaiheen kontollointiin Java tarjoaa **synchronized** tarkenteen. Sitä voidaan soveltaa monipuolisesti mihin tahansa olioon, käskysarjaan tai metodiin.

b. Menetelmän perusidea
Luodaan luokka, joka joko perii luokan Thread tai toteuttaa rajapinnan Runnable. Luokassa on run()-metodi, jossa on koodi joka halutaan ajaa. Säie käynnistetään luokan start()-metodilla. Kriittinen vaihe kontrolloidaan synchronized tarkenteen avulla.

c. Menetelmän yhteneväisyydet/eroavaisuudet kurssilla esitettyyn perustapaukseen verrattuna
On yhteneväinen kurssilla esitettyyn. Ratkaisu alkoi muistuttaa monitori-käsitettä ja sitä, miten se toteutetaan Javassa. Jos ohjelmointikieli itsessään tarjoaa ratkaisun kriittisen alueen kontrollointiin, kannattaa sitä käyttää selkeyden vuoksi.

d. Käyttöesimerkit
Tuottaja-kuluttaja esimerkki ProconTest.java
Laskuri esimerkki yhteisen muuttujan päivittämisestä Laskuri.java
Käyttöesimerkit on dokumentoitu lähdekoodissa.

e. Käyttöohje
Kopioi ProconTest.java tiedosto oman koneesi levylle.
Tarkista/aseta polkumuuttuja kuntoon, jotta voit kääntää ja ajaa komentoriviltä ohjelman.
Käännä ohjelma käskyllä >javac ProconTest.java
Aja käännetty ohjelma käskyllä >java ProconTest
Ohjelma neuvoo käytön eteenpäin.

Sama ohje käy myös Laskuri.java:lle.

Ohjelmointitehtävän listaus

Käyttöesimerkki: Tuottaja-kuluttaja

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public class ProconTest
{


   private static Producer p;
   private static Consumer c;

   private int value;
   private static final int MAX = 10;
   private int[] buff = new int[MAX];
   private boolean hasValue = true;
   private int kpl = 0;

   private static boolean sync = true;

   /**
    * @param args
    */
   public static void main(String[] args) throws IOException
   {
      BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
      String input;

      System.out.println("Haluatko ajaa synkronoituna/ei-synkronoituna/lopettaa [s/e/l]");
      do
      {
         input = br.readLine();

         if (input.equalsIgnoreCase("s"))
         {
            System.out.println("Ajamme synkronoituna ...");
            sync = true;
            ProconTest pc = new ProconTest();
            p = pc.new Producer();
            c = pc.new Consumer();
            p.start();
            c.start();
         }
         else if (input.equalsIgnoreCase("e"))
         {
            System.out.println("Ajamme ei-synkronoituna ......");
            sync = false;
            ProconTest pc = new ProconTest();
            p = pc.new Producer();
            c = pc.new Consumer();
            p.start();
            c.start();
         }
         else if (input.equalsIgnoreCase("l"))
         {
            System.out.println("Lopetit ajamisen");
            continue;
         }
         else
         {
            System.out.println("Virheellinen syöte!");
            System.out.println("Haluatko ajaa synkronoituna/ei-synkronoituna/lopettaa [s/e/l]");
         }
      } while (!input.equalsIgnoreCase("l"));

   }


   public void setUnsyncValue(int value)
   {
      if (kpl < MAX)
      {
         buff[kpl] = value;
         kpl++;
```

```
   }
}

public int getUnsyncValue()
{
   int temp = buff[0];
   if (kpl > 0)
   {
      for (int i = 1; i < MAX; i++)
      {
         buff[i - 1] = buff[i];
      }
      kpl--;
   }
   return temp;
}

// Kriittisen alueen kontrollointi
// Varmistetaan, ettei yhteistä muistia pääse käsittelemään samaan aikaan
// eri säikeet.
public synchronized void setValue(int value)
{
   // Jos puskuri on täynnä, odottaa
   while (kpl >= MAX)
   {
      try
      {
         wait();
      }
      catch (InterruptedException e)
      {
         System.err.println(e.toString());
      }
   }
   buff[kpl] = value;
   kpl++;
   notifyAll();
}

// Kriittisen alueen kontrollointi
// Varmistetaan, ettei yhteistä muistia pääse käsittelemään samaan aikaan
// eri säikeet.
public synchronized int getValue()
{
   // Jos puskuri on tyhjä, odottaa
   while (kpl == 0)
   {
      try
      {
         wait();
      }
      catch (InterruptedException e)
      {
         System.err.println(e.toString());
      }

   }
   int temp = buff[0];
   // Järjestetään puskuri
   for (int i = 1; i < MAX; i++)
   {
      buff[i - 1] = buff[i];
   }
   kpl--;
   notifyAll();
   return temp;
}

public boolean hasValue()
{
   return hasValue;
}

public void setHasValue(boolean hasValue)
{
   this.hasValue = hasValue;
}
```

```
/* -------------- Inner class Producer -----------------------
*
*          Tuottaa 10 arvoa yhteiseen puskuriin epämääräiseen tahtiin.
*
*
*
*/

public class Producer extends Thread
{

   public void run()
   {
      for (int i = 0; i < MAX; i++)
      {
         try
         {
            Thread.sleep((int) (Math.random() * 3000));
         }
         catch (Exception e)
         {
            System.err.println(e.toString());
         }
         if (sync)
         {
            setValue(i);
            System.out.println("Tuottaja asetti : " + i);
         }
         else
         {
            setUnsyncValue(i);
            System.out.println("Tuottaja asetti : " + i);
         }
      }
      setHasValue(false);
   }

}

/* -------------- Inner class Consumer -----------------------
*
*          Kuluttaa arvoja yhteisestä puskurista epämääräiseen tahtiin ja
*          säilöö ne omaan sisäiseen puskuriinsa, josta tulostaa ne lopuksi,
*          jotta voidaan varmistua, että kuluttaja on saanut kaikki arvot.
*
*/


public class Consumer extends Thread
{
   int[] cBuff = new int[MAX];

   int index = 0;

   public void run()
   {
      int value;

      while (hasValue() || kpl > 0)
      {
         try
         {
            Thread.sleep((int) (Math.random() * 3000));
         }
         catch (Exception e)
         {
            System.err.println(e.toString());
         }
         if (sync)
         {
            value = getValue();
            cBuff[index] = value;
            index++;
            System.out.println("Kuluttaja sai : " + value);
         }
         else
```

```
            {
               value = getUnsyncValue();
               if (index < MAX)
               {
                  cBuff[index] = value;
                  index++;
               }
               System.out.println("Kuluttaja sai : " + value);
            }
         }

      System.out.println("Kuluttaja sai seuraavat arvot:");
         for (int i = 0; i < MAX; i++)
         {
            System.out.print(" " + cBuff[i]);
         }
         System.out.println("\nHaluatko ajaa synkronoituna/ei-synkronoituna/lopettaa [s/e/l]");
       }

   }

}
```

Käyttöesimerkki: Yhteinen muuttuja, sitä käsittelevä laskuri.

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public class Laskuri
{
   private static PPlus p;
   private static QPlus q;


  private static final long MAX = 1000000;

  private static long n = 0;

   private static boolean sync = true;


  public static void main(String[] args) throws IOException
   {
      BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
      String input;

      System.out.println("Haluatko ajaa synkronoituna/ei-synkronoituna/lopettaa [s/e/l]");
      do
      {
         input = br.readLine();

         if (input.equalsIgnoreCase("s"))
         {
            System.out.println("Ajamme synkronoituna ...");
            sync = true;
            n = 0;
            Laskuri l = new Laskuri();
            p = l.new PPlus();
            q = l.new QPlus();
            p.start();
            q.start();
         }
         else if (input.equalsIgnoreCase("e"))
         {
            System.out.println("Ajamme ei-synkronoituna ......");
            sync = false;
            n = 0;
            Laskuri l = new Laskuri();
            p = l.new PPlus();
            q = l.new QPlus();
            p.start();
            q.start();
         }
         else if (input.equalsIgnoreCase("l"))
         {
            System.out.println("Lopetit ajamisen");
            continue;
         }
         else
         {
            System.out.println("Virheellinen syöte!");
            System.out.println("Haluatko ajaa synkronoituna/ei-synkronoituna/lopettaa [s/e/l]");
         }
      } while (!input.equalsIgnoreCase("l"));

   }

   // Kriittisen alueen kontrollointi
   public synchronized void calc()
   {
      n = n + 1;
      System.out.println("Calc = " + n);
   }

   // Kriittistä aluetta ei kontrolloida, antaa yleensä vääriä tuloksia
   public void unsyncCalc()
   {
```

```
      n = n + 1;
      System.out.println("UnsyncCalc = " + n);
   }


  /* -------------- Inner class PPlus -----------------------
   *
   *          Suorittaa säikeessä laskentapyynnön.
   */

   public class PPlus extends Thread
   {

      public void run()
      {
         for (int i = 0; i < MAX; i++)
         {
            if (sync)
            {
               calc();
            }
            else
            {
               unsyncCalc();
            }
         }
      }

   }

   /* -------------- Inner class QPlus -----------------------
   *
   *          Suorittaa säikeessä laskentapyynnön.
   */


   public class QPlus extends Thread
   {

      public void run()
      {
         for (int i = 0; i < MAX; i++)
         {
            if (sync)
            {
               calc();
            }
            else
            {
               unsyncCalc();
            }
         }
      }

   }

}
```