DATAVETENSKAP
Göteborg (2006-12-14)

TENTAMEN

CTH

2006-12-19

TENTAMEN I

Exempeltenta PROGRAMMERINGSTEKNIK

DAG: **TISDAG** TID: **14.00-18.00** SAL:**V**

Ansvarig:	Erland Holmström tel. 1007, hem 270358.
Resultat:	Anslås den **2007-01-19**
Lösningar:	Anslås (eventuellt) på MC samtidigt som resultatet.
Granskning: kan	Tentan finns (efter anslagning av resultatet) på expeditionen där rättningen granskas. Tid för klagomål mot rättningen meddelas samtidigt som resultatet.
Betygsgränser:	**Godkänd -18p, 4 - 24p, 5 - 30p. (maxpoäng är 40p)** (Siffror inom parentes anger maximal poäng på varje uppgift. I vissa fall ges även ungefärlig poängfördelning inom uppgiften.)
Hjälpmedel:	Kan diskuteras :-)

Var vänlig och läs detta:

- Börja med att läsa igenom HELA tesen så du kan ställa frågor när jag kommer.
- Observera att svar skall motiveras där så är lämpligt.
- **Skriv läsligt!** Rita gärna figurer. Svårlästa lösningar bedöms ej!!
- Börja varje uppgift på nytt blad. (Dock ej deluppgifter). Skriv endast på en sida av pappret.
- Skriv ditt personnummer på **alla** blad. Skriv också sektion och inskrivningsår åtminstone på omslaget.
- Programmen skall vara skrivna i Java och vara indenterade och kommenterade.
  I de uppgifter där det spelar någon roll, antas att man kör under UNIX-operativsystem.
- De råd och anvisningar som givits under kursen **skall** följas vid programkonstruktionerna. Det innebär bla att onödigt komplicerade, långa och/eller ostrukturerade lösningar i värsta fall ej bedöms. Programmen skall skrivas som generella enheter som är lätta att förstå (för andra än skrivaren) och lätta att ändra i när förutsättningarna ändras. De skall dessutom uppfylla normala krav på objektorienterade komponenter.

LYCKA TILL!!!

Uppgift 1. Javas enklaste fält (dvs array) är ganska statiska, dom kan inte enkelt ändra sin storlek. Antag nu att man ändå vill kunna fördubbla ett fälts storlek när det börjar bli fullt. Man skriver följande program:

```
public class Arr2twiceArr {
     static void arr2twiceArr(int[] arr) {
          int[] tmp;
          if ( arr.length > 0 ) {
               tmp = new int[2*arr.length];
               // kopiera över befintliga element till tmp
               for(int i=0; i<arr.length; i++) {
                    tmp[i] = arr[i];
               }
               // fyll resten med 0
               for(int i=arr.length; i<tmp.length; i++) {
                    tmp[i] = 0;
               }
          } else { // arr var ett tomt fält
               tmp = new int[1]; // fält med 1 element
               tmp[0] = 0;
          }
          arr = tmp; // byt namn
     } // end arr2twiceArr
     static void printTabell(int[] tab) {
          for(int i=0; i<tab.length; i++) {
               System.out.print(tab[i] + " ");
          }
     } // end fakTabell
     // ---------- ---------- ---------- ----------
     public static void main(String[] args) {
          int[] tab = {1,2,3,4,5,6,7,8,9,10};
          printTabell(tab); // anrop 1
          System.out.println();
          arr2twiceArr(tab);
          printTabell(tab);  // anrop 2
          System.out.println();
          System.exit(0);
     } // end main
} // end Arr2twiceArr
```

a) Programmet fungerar dock inte riktigt som det var tänkt. Vad skrivs ut av de bägge anropen av `printTabell` i main metoden? Varför blir det så? (Det bästa sättet att beskriva vad som händer är att rita figurer och kommentera dem

b) Skriv om `arr2twiceArr` och `main` så att det fungerar som tänkt. Du behöver bara beskriva ändringarna i programmet ovan om dom är få.

(5p)

Uppgift 2. Man kan beräkna summan av ett antal kvadrater mellan m och n, $m \leq n$,

$m^2 + (m+1)^2 + ... + (n-1)^2 + n^2$, på många sätt. Här är en iterativ metod för det

```
static int sumOfSquares(int m, int n) {
     int sum = 0;
     for (int i=m; i<=n; i++) {
          sum = sum + i*i;
     }
     return sum;
}
```

Skriv en *rekursiv* funktion som beräknar sumOfSquares genom att dela intervallet m..n på mitten. (5p)

Uppgift 3. **Goldbachs Hypotes**:

"Varje jämnt naturligt tal större än två går att skriva som en summa av två primtal"

Till exempel gäller att 4=2+2, 6=3+3, 8=5+3, 10=5+5, 12=5+7. Denna löst grundade förmodan framställdes år 1742 av den Preussiske matematikern Christian Goldbach i ett brev till Leonhard Euler, (E.T. Bell, *Matematikens drottning*, i Sigma, band 5 sid 1388, Forum 1977) och anses allmänt vara sann. Satsen lär dock ännu inte vara bevisad.

Din uppgift är nu att försöka undersöka om hypotesen verka stämma. Ett sätt att göra det är ju att pröva så många tal som möjligt, ju fler man prövar ju större är sannolikheten att hypotesen stämmer.

a) Uppgiften är att skriva ett program som upprepade gånger frågar efter ett heltal och om det är jämnt skriver ut alla möjliga kombinationer av två primtal som bildar dess summa. Varje kombination av primtal skall endast skrivas ut en gång. Till exempel skall programmet för indata 14 endast ge ett av primtalsparen 3,11 och 11,3 som utdata. Är talet udda skall programmet ge meddelande om detta. Körningen avslutas när ett tal ≤ 2 ges. Använd en tabell, `primTab`, enligt uppg b.

```
% java Goldbach                          exekvering
ange ett heltal (sluta: <=2): 14         programmet frågar, indata 14
3 + 11                                   utdata
7 + 7
2 kombinationer          skriv ut antalet kombinationer som hittats
Ge ett tal: 1            programmet stannar
```

b) Man kommer ju att undersöka samma primtal många gånger så det borde löna sig att först beräkna någon sorts tabell som anger vilka tal som är primtal, och sedan använda denna. Tabellen kan t.ex. konstrueras m.h.a. den nedan beskrivna metoden med Erathostenes såll, som hittar alla primtal mindre än eller lika med ett givet naturligt tal `n`.

Erathostenes såll

> Skriv upp de naturliga talen från och med 2 till och med `n`. Utgå från talet 2 och stryk alla multipler av 2 i följden utom 2 självt. Utgå från talet 3 och stryk alla multipler av 3 i följden utom 3 självt. Nästa tal som inte är struket är 5. Utgå från det och stryk alla dess multipler i följden utom 5 självt. Fortsätt att på detta sätt stryka alla multipler av nästa ej strukna tal *p* (som är ett primtal) för varje sådant $p \le \sqrt{n}$. De tal som nu ej är strukna utgör alla primtalen som är mindre än eller lika med `n`.

Du behöver alltså en metod som fyller tabellen

```
static public void fyllTabell(boolean[] primTab) {...
```

När metoden implementeras i Java är det praktiskt att som tabell använda ett fält (kalla det "`primTab`") som indexeras med talen *2..n* och vars komponenter är av typen `BOOLEAN`. Alla komponenter initieras till `TRUE`. Ett tal *n* stryks genom att sätta position *n* i tabellen till `FALSE`. Att använda (den ifyllda) tabellen för att avgöra om ett tal är ett primtal är lätt. Om tabellingång *n* är `TRUE` så är *n* ett primtal.

(12p)

Uppgift 4. Beskriv kort följande Java begrepp och beskriv skillnaden mellan dem.

- JFrame och en Jpanel?
- Klass, abstrakt klass och interface?
- Objekt och klass?
- Olassvariabel och instansvariabel

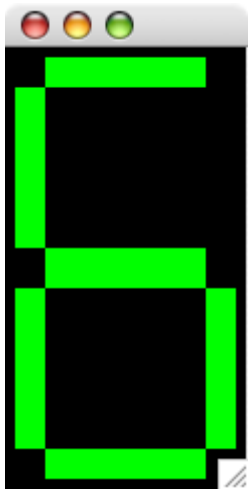
(4p)

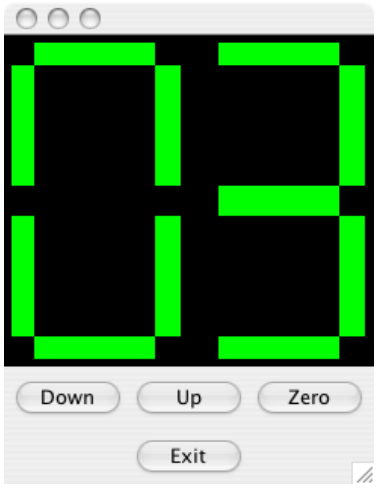
Uppgift 5. En klass med följande gränssnitt finns tillgänglig, klassen är en digital siffra enligt vänstra figuren nedan där `setNumber(6)` anropats.

```
public class Digit extends JPanel {
     // prefered, but not mandatory, colors
     // color is set in Digit by calling getForeground()
     // so users of the class can set color by calling setForeground
     public static final Color greenDigit      = Color.green;
     public static final Color redDigit        = Color.red;
     public static final Color backgroundColor = Color.black;

     public Digit();      // sets the digit to 0
     public Digit(int n); // sets the digit to n
     public void setNumber(int n); // sets the digit to n
     public int getNumber();
     public void paintComponent(Graphics g);

}
```





Du skall nu skriva en klass, `DigitalCounter,` som ser ut som figuren till höger ovan och som använder sig av klassen Digit. Den skall, förutom konstruktor och actionPerformed, innehålla följande metod:

```
// outputs a number in the range 0..99.
// when nbr<0 outputs 99, when nbr>99 outputs 0
public void outputNumber(int nbr) {
```

När man klickar på “down” skall talet som visas minskas med ett och när man klickar på “up” ökas med ett, bägge med “wraparound” dvs `up(99)` blir 0 osv. Knappen “zero” nollställer räknaren och “exit” avslutar programmet.

(14p)

```
/*
 Kort:
 Parameter–verf–ringen sker med v”rdeanrop och dÂ kan inte tab ”ndras av
funktionen.
 D”rf–r f–r”ndras inte tab och man fÂr 1,2,3,4,5,6,7,8,9,10 utskrivet 2
gÂnger.

 Lite l”ngre:
 N”r man anropar arr2twiceArr(tab); sÂ kopieras v”rdet av tab
 (som ”r en adress) till arr (som ”r en *ny* variabel) men arr kopieras
inte tillbaka.

 Satsen arr = tmp; funkar alldeles utm”rkt och g–r det man t”nkt sÂ l”
nge
 man "t”nker lokalt" dvs inuti funktionen. Med en utskrift alldeles
efter
 tilldelningen enligt
 arr = tmp;
 printTabell(arr);
 sÂ skulle man fÂ ut en vektor med f–rsta h”lften lika med tab och
 andra h”lften fylld med nollor, precis som man ville.
 Problemet ”r att arr ”r en lokal variabel i arr2twiceArr som alltsÂ
 upph–r att existera n”r man l”mnar funktionen och argumentet tab i
 arr2twiceArr(tab); f–r”ndras inte eftersom parameter–verf–ringen sker
med v”rdeanrop.
 */

public class Arr2twiceArr {

      static void arr2twiceArr(int[] arr) {
            int[] tmp;
            if ( arr.length > 0 ) {
                  tmp = new int[2*arr.length];
                  // kopiera –ver befintliga element till tmp
                  for(int i=0; i<arr.length; i++) {
                        tmp[i] = arr[i];
                  }
                  // fyll resten med 0
                  for(int i=arr.length; i<tmp.length; i++) {
                        tmp[i] = 0;
                  }
            } else { // arr var ett tomt f”lt
                  tmp = new int[1]; // f”lt med 1 element
                  tmp[0] = 0;
            }
            arr = tmp; // byt namn
      } // end arr2twiceArr
      // 2 f–r”ndringar:
      // - arr2twiceArr2 mÂste vara en funktion som returnerar ett f”lt
      // - return tmp; ist”llet f–r ar
      static int[] arr2twiceArr2(int [] arr) {
            int[] tmp;
            if ( arr.length   > 0 ) {
                  tmp = new int[2*arr.length];
                  // kopiera –ver befintliga element till tmp
                  for(int i=0; i<arr.length; i++) {
                        tmp[i] = arr[i];
                  }
                  // fyll resten med 0
                  for(int i=arr.length; i<tmp.length; i++) {
                        tmp[i] = 0;
```

```
                }
          } else { // arr var ett tomt f”lt
                tmp = new int[1]; // f”lt med 1 element
                tmp[0] = 0;
          }
          return tmp;
    } // end arr2twiceArr2


    static void printTabell(int[] tab) {
          System.out.println("Length = " + tab.length);
          System.out.print("[");
          for(int i=0; i<tab.length; i++) {
                System.out.print(tab[i] + " ");
          }
          System.out.println("]");
    } // end fakTabell
      // ----------   ---------- ---------- ----------
    public static void main(String[] args) {
          int[] tab = {1,2,3,4,5,6,7,8,9,10};
          printTabell(tab); // anrop 1
          System.out.println();
          arr2twiceArr(tab);
          printTabell(tab);  // anrop 2, ingen ”ndring
          System.out.println();
          tab = arr2twiceArr2(tab);
          printTabell(tab); // anrop 3 med "ny" tabell
          System.out.println();
          System.exit(0);
    } // end main
} // end Arr2twiceArr
```

```
public class SummaKvadrater {
// ---------- ---------- ---------- ----------
static int SumOfSqs5(int m, int n) {
      int mid = (m+n)/2;
      if (m > n) {
            return 0;
      } else if (m == n) {
            return m*m;
      } else {
            return SumOfSqs5(m,mid) +
                              SumOfSqs5s(mid+1,n);
      }
} // end SumOfSqs5;
// ---------- ---------- ---------- ----------
      public static void main(String[] args) {
            System.out.println("SumOfSqs5= " + SumOfSqs5(1, 3));
            System.out.println("SumOfSqs5= " + SumOfSqs5(2, 4));
            System.out.println("SumOfSqs5= " + SumOfSqs5(2, 2));
            System.exit(0);
    } // end main
} // end SummaKvadrater
```

JFrame och en Jpanel
B”gge ”r komponenter i Swing, anv”nds f–r att strukturera grafiska anv”ndargr”nssnitt
En frame har en menyrad och en "kant" vilket inte en panel har.

---
klass, abstrakt klass och interface
En klass ”r en beskrivning av hur ett objekt kommer att se ut n”r det skapas.
Klassen ”r "f”rdig" att anv”nda f–r att deklarera objekt.

En abstrakt klass ”r som en klass men en eller flera metod-kroppar har utel”mnats. Dessa mÂste skrivas i den klass som ”rver den abstrakta klassen.

I ett inteface har man utel”mnat alla metodkroppar och det kan inte finnas nÂgra instansvariabler. Interface ”rvs inte utan man implementerar dem.

---
objekt och klass
Klassen ”r objektets typ.
Klasser definierar objekt, ”r en mall/ritning f–r hur objektet skall se ut
och vad man kan g–ra med ett objekt.
En klass innehÂller en beskrivning av tillstÂndet dvs data och beteendet (handlingar) dvs metoderna.
Jmf ritning pÂ ett hus och huset sj”lvt.

---
klassvariabel och instansvariabel
Dessa innehÂller objektets tillstÂnd.
Varje objekt har en egen upplaga av instansvariablerna.
En klassvariabel ”r gemensam f–r alla objekt.

```
import java.awt.*;
import javax.swing.*;
import java.awt.Color;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;

public class DigitalCounter extends JFrame implements ActionListener {

      private int number = 0;
      private static final Color green = Digit.greenDigit; // ** beh–vs
ej f–r uppgiften
      private static final Color red   = Digit.redDigit;   // **

    private Digit a = new Digit();
    private Digit b = new Digit();

      private void colorSetter(Color col) { // **
            a.setForeground(col);
            b.setForeground(col);
      }

    // output a number to the respective digits
      public void outputNumber(int nbr) {
        number = nbr;
            if (number < 0) {
            number = 99;
        } else if (number > 99) {
            number = 0;
        }
        a.setNumber(number/10);
        b.setNumber(number%10);
      }

      public DigitalCounter() {
        // a panel for a 2 digit number
            JPanel digitPanel = new JPanel();
            digitPanel.setLayout(new GridLayout(1,2));
            digitPanel.setBackground(Color.black);
        // add the 2 digits to the panel
            digitPanel.add(a);
            digitPanel.add(b);

            // two panels for the buttons
            JPanel buttonPaneltop = new JPanel();
            buttonPaneltop.setLayout(new FlowLayout());
        JPanel buttonPanelbottom = new JPanel();
            buttonPanelbottom.setLayout(new FlowLayout());
            // buttons
            JButton down   = new JButton("Down");
            JButton up     = new JButton("Up");
        JButton zero   = new JButton("Zero");
            JButton exit   = new JButton("Exit");
            // set action command
        down.setActionCommand("down");
        up.setActionCommand("up");
        zero.setActionCommand("zero");
        exit.setActionCommand("exit");
            // register listeners
        down.addActionListener(this);
        up.addActionListener(this);
        zero.addActionListener(this);
```

```
        exit.addActionListener(this);
            // add buttons to the panels
        buttonPaneltop.add(down);
        buttonPaneltop.add(up);
        buttonPaneltop.add(zero);
        buttonPanelbottom.add(exit);
            // panel to include the 2 button panels
        JPanel buttonPanel = new JPanel();
            buttonPanel.setLayout(new BorderLayout());
            buttonPanel.setBackground(Color.black);
        buttonPanel.add(buttonPaneltop, BorderLayout.NORTH);
        buttonPanel.add(buttonPanelbottom, BorderLayout.SOUTH);
            // finally add everything to the frame
        setBackground(Color.black);
            setLayout(new BorderLayout());
        add(digitPanel, BorderLayout.CENTER);
            add(buttonPanel, BorderLayout.SOUTH);

            pack();
            setVisible(true);
            setDefaultCloseOperation(EXIT_ON_CLOSE); // **
      }
      // Controller
      public void actionPerformed(ActionEvent e) {
        //System.out.println(e.getActionCommand());
        if (e.getActionCommand().equals("down")) {
            outputNumber(--number);
        }
        if (e.getActionCommand().equals("up")) {
            outputNumber(++number);
        }
        if (e.getActionCommand().equals("zero")) {
            outputNumber(0);
        }
        if (e.getActionCommand().equals("exit")) {
                  System.exit(0);
            }
    }
    public static void main(String[] arg) {  // **
            new DigitalCounter();

      }

}
```

```
import java.awt.*;
import java.awt.Color;
import javax.swing.*;
import javax.swing.border.*;

public class Digit extends JPanel {
    // prefered, but not mandatory, colors
    public static final Color greenDigit      = Color.green;
    public static final Color redDigit        = Color.red;
    public static final Color backgroundColor = Color.black;

    private int n; // remember the dispayed number, 0<=n<=9

      // first index is the number, second which line to lit
    private static final boolean[][] light = {
            {true,  true,  true,  false, true,  true,  true},     // 0
      1111
            {false, false, true,  false, false, true,  false},    // 1
      2  3
            {true,  false, true,  true,  true,  false, true},     // 2
      2  3
            {true,  false, true,  true,  false, true,  true},     // 3
      4444
            {false, true,  true,  true,  false, true,  false},    // 4
      5  6
            {true,  true,  false, true,  false, true,  true},     // 5
      5  6
            {true,  true,  false, true,  true,  true,  true},     // 6
      7777
            {true,  false, true,  false, false, true,  false},    // 7
            {true,  true,  true,  true,  true,  true,  true},     // 8
            {true,  true,  true,  true,  false, true,  true}};    // 9

    public Digit() {
            this(0);
    }
    public Digit(int n) {
            setPreferredSize(new Dimension(120, 220)); // default size
of one letter
      setBackground(backgroundColor);
        setBorder (new LineBorder(backgroundColor, 5)); //space between
numbers
        setForeground(greenDigit);
        setNumber(n);
    }

    public void setNumber(int n) {
            if (n >= 0 && n <= 9) {
            this.n = n;
            } else {
            throw new IllegalArgumentException("Single digit numbers
only");
        }
      repaint();
    }

    public int getNumber() {
            return n;
    }

      public void paintComponent(Graphics g) {
```

```
            super.paintComponent(g);

            // Get the appropriate size for the panel
            int panelWidth = getSize().width;
            int panelHeight = getSize().height;
            // calculate the size/coordinates of the lines
            // the proportions of the digital numbers is preferably 6*11
squares (width*heigth)
            // one square is
            int sqWidth = (panelWidth)/6;
            int sqHeight = (panelHeight)/11;
            // the lines are
            int horLL =  4*sqWidth;       // LineLength
            int verLL =  4*sqHeight;      // =lineThickness equals
sqWidth and sqHeight
            // finally calculate the coordinates for the 7 lines for
fillRect
            // linenbr      1        2         3          4           5
6           7
            int[] x =     { sqWidth, 0,        5*sqWidth, sqWidth,    0,
5*sqWidth,  sqWidth    };
            int[] y =     {  0,      sqHeight, sqHeight,  5*sqHeight, 6
*sqHeight, 6*sqHeight, 10*sqHeight};
            int[] width = { horLL,   sqWidth,  sqWidth,   horLL,
sqWidth,    sqWidth,    horLL      };
            int[] heigh = { sqHeight,verLL,    verLL,     sqHeight,
verLL,      verLL,      sqHeight   };

            g.setColor(getForeground());
            for (int i = 0; i < 7; i++) {
                  if (light[n][i]) {
                        g.fillRect((x[i]), (y[i]), (width[i]), (heigh
[i]));
                  }
            }
      }
}
```