Data och Informationsteknik / Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
Erland Holmström

Göteborg 12 feb 2015

Examination i

PROGRAMMERINGSTEKNIK F1/TM1 TIN212

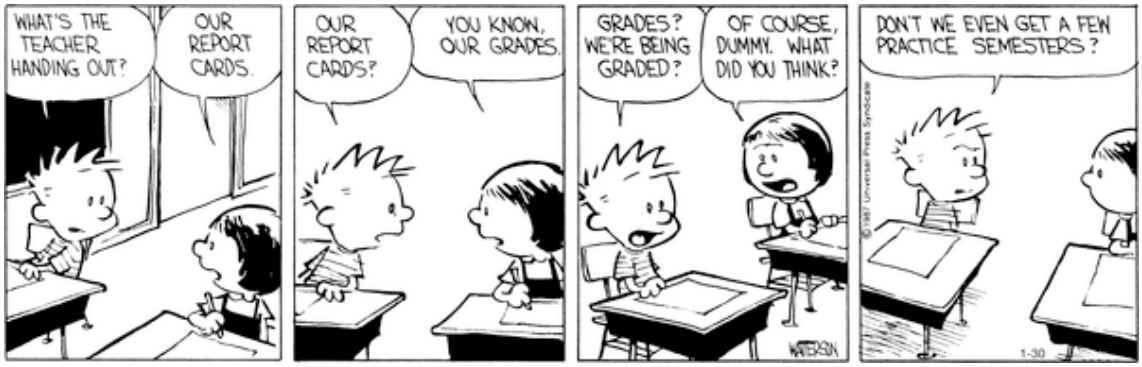
Dag: Måndag Datum: 2015-02-16 Tid: 8.30-13.30 (OBS 5 tim) Rum: V+M

Ansvarig lärare:	Erland Holmström tel. 1007, mobil 0708-710 600
Resultat:	Skickas med mail från Ladok.
Lösningar:	Läggs eventuellt på hemsidan.
Tentagranskning:	Tentan finns efter att resultatet publicerats på vår expedition. Tid för klagomål på rättningen annonseras på hemsidan efter att resultatet är publiserat (eller maila mig så försöker vi hitta en tid).
Betygsgränser:	CTH: 3=28p, 4=38p, 5= 48p, max 60p
Hjälpmedel:	**Bravaco, Simonson: Java Programming From the Grounf Up**

Observera:

- Börja med att läsa igenom alla problem så du kan ställa frågor när jag kommer.
  Jag brukar komma efter ca 1-2 timmar (men det är typ 6 salar så det kan bli senare).
- Alla svar måste motiveras där så är möjligt/lämpligt.
- Skriv läsligt! Rita figurer. Lösningar som är svårlästa bedöms inte.
- Skriv kortfattat och precist.
- Råd och anvisningar som getts under kursen skall följas.
- Programs skall skivas i Java, skall vara indenterade och lämpligt kommenterade.
- Börja nya problem på ny sida.
-
- ***Lämna inte in tesen med tentan utan behåll den.*** Lämna inte heller in kladdpapper/skisser.

Lycka till!



Problem 1. Sant eller falskt? (Motivera.) Eller svara på frågorna.

a) Vad är skillnaden mellan en typ och ett objekt? Ge exempel på deklaration av en typ och av ett objekt. Vid vilken deklaration reserveras minne? (3)

b) Vad skriver följande program ut och varför? (2)

```
String str = "Hejsan";
String str2 =str.substring(0,3);
if(str2=="Hej"){
  System.out.println("Lika");
}
```

c) Med deklarationen

```
JLabel l = new JLabel("Java",JLabel.CENTER);
```

skapas ett objekt av klassen `JLabel`. Vad är `CENTER`? Vad är `CENTERs funktion`? Var (i vilken klass) finns `CENTER` deklarerad? (2)

d) Förklara följande ords (Java-) betydelse: syntax och semantik, aktuella och formella parametrar. (2)

9p)

Problem 2. Testar: enkla loopar, aritmetik, typer

Skriv kodavsnitt som skriver ut talföljden *0.0, 0.1, 0.2, ... 1.0* med hjälp av en for loop:

(2p)

Problem 3. Testar: strängar, loopar, användandet av String och Character klasserna

I denna uppgift skall vi granska påståendet "Ett positivt heltal är delbart med siffran *k* om och endast om talets siffersumma är det". Det är lätt att matematiskt visa att det är sant för *k = 3* och för *k=9* men inte annars. T ex har talet *123111* siffersumman *1+2+3+1+1+1 = 9* och är därför delbart med *3* och *9*.

Du skall skriva en metod, *delbar(k, tal)* vars värde skall vara `true` om siffersumman för "tal" är delbar med *k* och annars `false`. Så `delbar(3, "123")` skall ge värdet `true` (eftersom *1+2+3=6* som är delbart med *3*) och `delbar(3, "1231")` värdet `false` (eftersom *1+2+3+1=7*).

Talen som skall testas kan vara (mycket) större än vad som ryms i en `int` eller en `long` så parametern `tal` måste vara en sträng enligt nedan. Siffersumman kommer dock alltid att rymmas i en `int`, ett tal med tex *1000* siffror har ju max siffersumma *9000*.

Fel i talet, dvs att tal inte är ett tal tex "123a45", skall hanteras med *lämplig* exceptions och felobjektet skall innehålla en kort förklarande text; " Fel i indata" och den felaktiga strängen.

```
public static boolean delbar(int k, String s)
                                throws  <lämplig exception> ...
```

Skriv också ett huvudprogram som anropar "delbar" och tar hand om eventuella exceptions

(max 8p)

Problem 4. *Testar: vektorer, matriser, loopar, metoder*

a) Skriv en boolesk metod, `ortoVec`, som avgör om två vektorer är ortogonala. Två vektorer

$a = a_{1,} a_{2,} \ldots, a_n$ och $b = b_{1,} b_{2,} \ldots, b_n$

säges[1] vara ortogonala om skalärprodukten (inre produkten) är lika med noll:

$$a \cdot b = \sum_{i=1}^{n} a_i b_i = a_1 b_1 + a_2 b_2 + \ldots + a_n b_n = 0 \ .$$

ortoVec skall hantera (alla) fel i vektorerna och returnera lämpligt värde (true eller false) för alla alternativ.

b) Skriv en boolesk metod, `ortoMat`, som avgör om en given, godtycklig matris är ortogonal. En matris sägs vara ortogonal om varje par av olika radvektorer är ortogonala. ortoMat behöver inte ta hand om fel i matrisen.

c) Skriv ett program som skapar en matris och avgör om den är ortogonal.

(8p)

Problem 5. *Testar: arv, metodheirarkier*

Antag att följande klasser finns

```
public class Moo {
  public void method1() {System.out.println("moo 1");}
  public void method2() {System.out.println("moo 2");}
  public String toString() {return "moo";}
} // end Moo  ==================================
public class Blue extends Moo {
  public void method1() {System.out.println("blue 1");}
} // end Blue  =================================
public class Flute extends Blue {
  public void method2() {System.out.println("flute 2");}
  public String toString() {return "flute";}
} // end Flute ================================
public class Shoe extends Flute {
  public void method1() {System.out.println("shoe 1");}
}
```

Vad skriver nu följande program ut?

```
public class Test {
  public static void main(String[] args) {
    Moo[] elements = {new Shoe(), new Flute(),
                      new Moo(), new Blue()};
    for (int i=0; i< elements.length; i++) {
      System.out.println(elements[i]);
      elements[i].method1();
      elements[i].method2();
      System.out.println();
    }
  }
} // end Test
```

(6p)

1Om vektorernas komponenter är kända i en ortonormerad bas men det är ointressant här.

Jag visade ett litet program på föreläsning som visualiserade laddningar och deras påverkan på varandra. Jag tänkte att ni skulle bygga vidare på det programmet här. Jag kunde ju visa laddningarna dels i färg (nedre bilderna nedan) och dels i svart-vitt (övre bilderna) och dels med ”kontinuerliga” färger (vänstra bilderna) och dels med diskreta färger (högra bilderna) men man var tvungen att hacka runt i koden för att byta mellan dom. Nu har jag skrivit om modellen så den klarar att byta om man anropar den med lämplig parameter.
Dessutom skulle det vara trevligt att kunna ”glida” mellan den kontinuerliga visningen och den diskreta så jag har också infört en skalfaktor som med värdet 255 ger den vänstra bilden och med 50 ger den högra och värden däremellan ger just bilder däremellan. Detta skall styras med en Jslider och på den sista bilden har jag dragit ner den till 22. Så fort man drar i slidern och anropar modellen så går modellen över till ”sliderläge”. De andra 4 bilderna får man genom att klicka på respektive knapp.

Själva modellen finns redan (se kod sist i tentan men du behöver inte förstå den i detalj) men jag behöver hjälp med att skriva en vy-klass och en kombinerad control och main-klass. Modellen skall också läsa in laddningarna från en fil till en matris så det är en metod i modellen som du också skall skriva. Uppgifterna är ganska fristående från varandra. Jag börjar med vyn men du kan börja var som helst.

Problem 6. *Testar: rita med Swing, loopar*

Du skall alltså skriva en vy-klass som kan användas i tex en frame för att presentera modellen dvs laddningarna enligt nedan. Knappar och slider kommer i följande uppgifter, det är bara själva bilden som skall göras här. Modellen skapar en matris med allt du behöver och man kan anropa metoden getPotential så får man en matris med färger som man kan skriva ut. Konstruktorn måste ha en modell som parameter och storleken kan du hämta i modellen.

(6p)

Problem 7. *Testar: Scanner, exceptions, loopar, fält,*

Modellen läser in laddningarnas placeringar och deras laddning från en fil och lagrar dom i ett fält, se PotentialModel, och den metoden vill jag att du skriver. Vi har inte pratat så mycket om att läsa från filer men scanner-klassen funkar utmärkt till det. Ge scannern en fil som parameter bara så fungerar det som vanligt sedan. En fil kan man skapa av en sträng, str, enligt *new File(str)*. En sak tillkommer gentemot när vi läser från tangentbordet och det är att scannern kan kasta en exception `FileNotFoundException` som man måste fånga. Gör det och kasta istället en `IllegalArgumentException` (som man sedan inte behöver fånga).

Det är alltså den här metoden du skall skriva, se mer beskrivning i koden sist.

```
public void readPotential(String file) throws
                          IllegalArgumentException {<fyll i här>}
```

Konstuktorn för en Charge ser ut så här

```
public Charge(double x0, double y0, double q0).
```

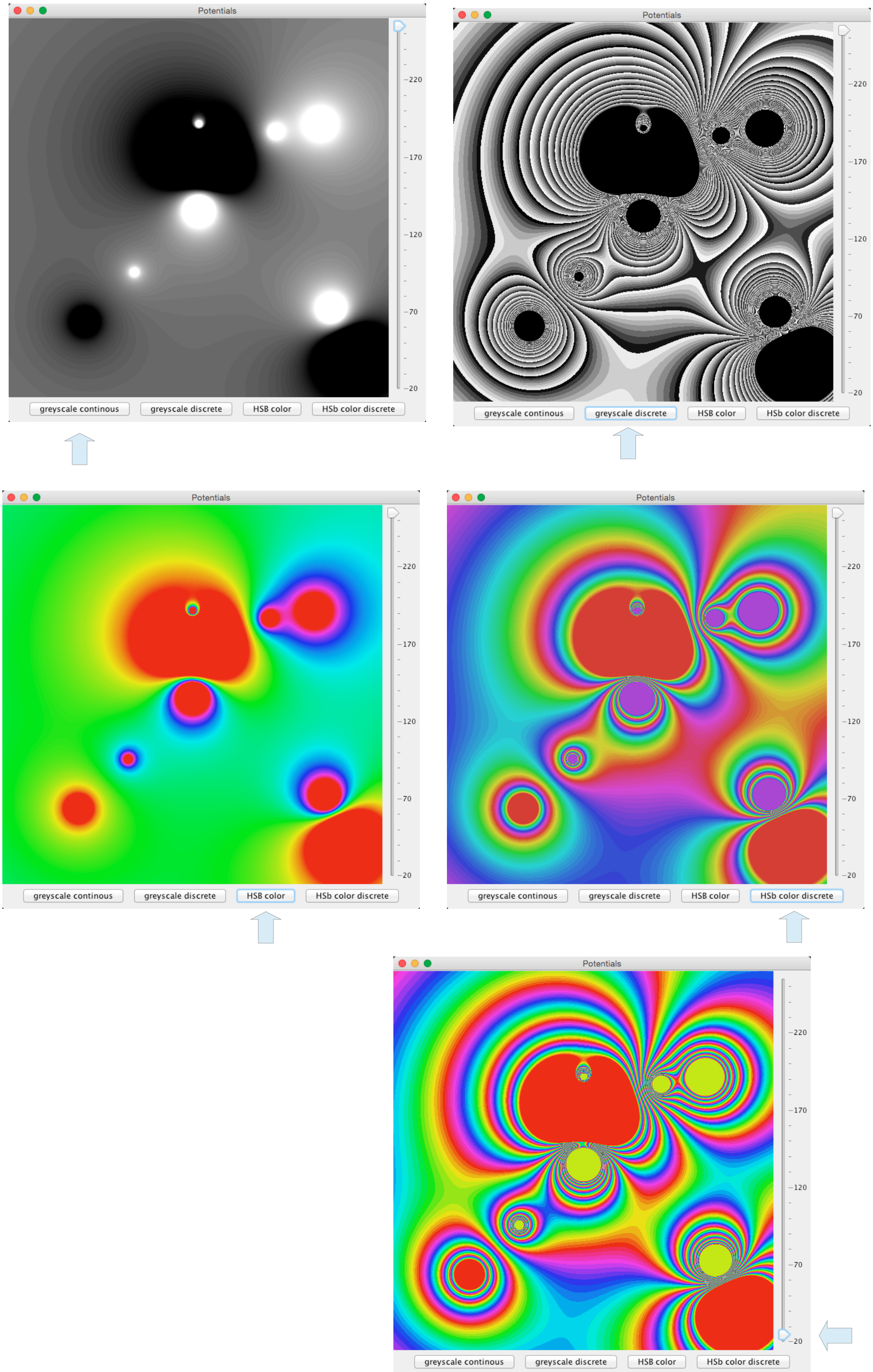
(6p)

Problem 8. *Testar: Swing, lyssnare mm*

Du skall också skriva en controler (som lämpligen är en Jframe) som skapar en modell och en vy samt en frame enligt figurena sist och också tar hand om knapptryck och sliderförändringar (Jslider API finns att låna). Det är troligen enklast att skriva en inre klass för varje knapp och en för slidern, du behöver bara skriva kod för en knapp och en knapplyssnare. En main metod skall finnas här också.
Man byter bild i modellen genom att anropa update med lämplig parameter.

Slidern skall ha major tick varje 50 och minor tick varje 10 och den skall starta på 255, se figurerna. Textfilen med laddningar finns i ”`charges2.txt`”.

(15p)


Potentials
220
170
120
70
20
greyscale continous
greyscale discrete
HSB color
HSb color discrete

```
/*************************************************
 *  Potential value visualization for a set of charges.
 *************************************************/
import java.awt.Color;
import java.util.Scanner;
import java.io.*;
import java.util.Locale;
public class PotentialModel {
  // array with potentials
  private Charge[] arrayOfCharges = null;
  // array with colors of potentials
  private Color[][] pic  = null;
  // internal array of charges in each point of the ”picture”
  private int[][] v = null;
  private int size = 600; // nbr of prefered pixels
  private float spline = 255.0f; // for scaling

  public PotentialModel(String file){
    readPotential(file);
  }

  /** Reads a set of charges from a file and stores them in a
  * 3-D matrix.
  * @param file The input file that contains charges.
  * The file contains a number n (int) which indicate how many
  * lines that follow, and then n lines with 3 numbers on each
  * line. The first two are the position of the charge (double)
  * and the last one is the “charge” at that position.
  * Together they represent a Charge.
  * This method stores the Charge objects in
  * the array "arrayOfCharges".
  * Here is the beginning of
    * beginning of a possible file
    * 9
    * .51 .63 -100
    * .50 .50   40
  ...
  */
  public void readPotential(String file)
                              throws IllegalArgumentException {
    // read in N point charges, assuming correct points
    Problem XXXXXX to go here
    getPotential(); // update tables, ignore result
  } // end readData
```

```
// pre-compute the potential at each point.
// only done once
private void updateV() {
  v = new int[size][size];
  pic = new Color[size][size];
  for (int i = 0; i < size; i++) {
    for (int j = 0; j < size; j++) {
      double V = 0.0;
      // ask each point how much it contributes to the charge
      for (int k = 0; k < arrayOfCharges.length; k++) {
        double x = 1.0 * i / size; // scale to 0<=x<=1
        double y = 1.0 * j / size;
        V = V + arrayOfCharges[k].potentialAt(x, y);
      }
      V = 128 + V / 2.0e10; // scale
      int t = 0;
      // int colors are 0<=t<0255
      if      (V <   0) t = 0;
      else if (V > 255) t = 255;
      else              t = (int) V;
      v[i][j] = t;
    }
  }
} // end updateV

/**
  * Returns a matrix with color values.
  * @return The matrix with color values.
*/
public Color[][] getPotential() {
  if (pic == null) {
    updateV();
    update(0);
  }
  return pic; // should be a copy
}
```

```
  /**
  * Updates the color matrix.
  */
  public void update(int colorSet) {
    if (v==null) {updateV();}
    // create an array of colors i.e.
    // compute the potential at each point and "plot"
    // i,j are picture indexes
    for (int i = 0; i < size; i++) {
      for (int j = 0; j < size; j++) {
        int t = v[i][j];
        Color c = null;
        switch (colorSet) {
        case 0: //"greyscaleCont": // kontinuerlig gråskala
          c = new Color(t, t, t);
          break;
          … på samma sätt för 1,2,3
        case 4: //"HSBslider": // kontinuerliga färger
          c = Color.getHSBColor(t / (float)spline, .9f, .9f);
          break;
        default:
          System.out.println("wrong color set " + colorSet);
        }
        pic[i][size-1-j] = c;
      }
    }
    System.out.println("update ended " + colorSet); // debug
  } // end update

  /**
  * spline is a scalefactor used by the JSlider.
  * @param The value of the spline.
  */
  public void setSpline(int spline) {
    this.spline = spline;
  }
  /**
  * Returns the prefered size of this ”picture”.
  * @return the prefered size
  */
  public int getSize() {
    return size;
  }

} // end class
```

Programmeringsteknik F 2015.02.16 Lösningsskisser

Uppg 1.
==========

a) Vad är skillnaden mellan en typ och ett objekt? Ge exempel på deklaration av en typ
och av ett objekt. Vid vilken deklaration reserveras minne?

Se boken/OH för mer om detta.
Man deklarerar en typ när man skriver en klass.
Man deklarerar ett objekt när man använder "new".
class XXXX ... är en typdeklaration
new XXXX(); är en objektdeklaration

b) Vad skriver följande program ut och varför?

```
String str = “Hejsan”;
String str2 =str.substring(0,3);
if(str2==”Hej”){
    System.out.println(“Lika”);
}
```

Ingenting. "==" jämför pekarna och dom är olika även om innehållet är samma.

c) Med deklarationen
JLabel l = new JLabel(”Java”,JLabel.CENTER);
skapas ett objekt av klassen JLabel. Vad är CENTER? Vad är CENTERs funktion?
Var (i vilken klass) finns CENTER deklarerad?

CENTER är en konstant i klassen JLabel (ärvs från SwingConstants). Den centrerar texten i
labeln.

d) Förklara följande ords (Java-) betydelse: syntax och semantik, aktuella och formella
parametrar.

se OH bilder och bok
syntax- hur man skriver, semantik- vad det betyder,
formell parameter är den som används som namn på parameter i en metod.
aktuell parameter är den man anropar metoden med.

Uppg 2.
==========

```
for (int i=0; i<=10; i=i+1) {
    System.out.println(i/10.0);
}
```

Observera att alla lösningar som loopar med double antagligen inte funkar utan dom skriver
tex

```
...
0.6
0.7
0.7999999999999999
...
0.8999999999999999
```

Uppg 3.
==========

```
public class DelbarMed3 {
    public static boolean delbar(int k, String s) throws  IllegalArgumentException {
        int sifsum = 0;
        for (int i=0; i < s.length(); i++) {
            char ch = s.charAt(i);
            if (!Character.isDigit(ch)) {
                throw new IllegalArgumentException("Ej ett tal: " + s);
            }
            sifsum = sifsum + (Character.digit(ch, 10));
            //sifsum = sifsum + (ch-'0'); // alternativ, det finns nog fler
        }
        return (sifsum % k == 0);
    }
// page breake

    public static void main(String[] s) {
        try {
            System.out.println(delbar(3, "11111112111"));
            System.out.println(delbar(3, "11111a1122"));
            System.out.println(delbar(3, "12345678901234567890101234523453453999999111"));
        }
        catch (IllegalArgumentException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

Uppg 4.
==========

```
public class OrtogonalMatris {

    private static boolean ortoVec(double[] a, double[] b) {
        if (a==null || b==null || (a.length != b.length)) {
            return false;
        } else {
            double sum = 0;
            for (int i=0; i<a.length; i++) {
                sum = sum + a[i]*b[i];
            }
            //return sum==0; // om man har heltalsfält
            return sum < 0.0000001;
        }
    }
    private static boolean ortoMat(double[][] m) {
        for (int i= 0; i<m.length-1; i++) {
            for (int j= i+1; j<m.length; j++) {
                if (!ortoVec(m[i], m[j])) {
                    return false;
                }
            }
        }
        return true;
    }
    public static void main(String[] args) {
        double[][] m = {{3,5,8,3,5},
                        {5,2,4,2,3},
                        {1,2,3,4,5},
                        {5,4,3,2,1}
                        };
        System.out.println(ortoMat(m));
    }
}
```

// page breake

Uppg 5.
==========
Motiveringarna är som vanligt viktiga på den här typen av frågor.
Motiveringar kan bygga på att man förklarar hur den dynamiska bindningen fungerar
(eller anger anropskedjan), eller att man förklarar arv/överskuggning (se bok/OH bilder).

Arvsordning: Moe <- Blue <- Flute <- Shoe
Anropsordning: Shoe, Flute, Moo, Blue
Vid ett anrop av typ o.metod() söker man efter en metod först hos "sig själv" dvs i klassen för o,
sedan hos förälderklassen till o, sedan hos "grandparent" osv.

resultat anrop Shoe:
flute Shoe anropar Flutes toString (ts)
shoe 1 egen
flute 2 Shoe anropar Flutes m2

resultat anrop Flute:
flute egen ts
blue 1 Blues m1
flute 2 egen m2

resultat anrop Moo:
moo alla egna
moo 1
moo 2

resultat anrop Blue:
moo Moo's ts
blue 1 egen
moo 2 moo's m2

En student hade gjort en tabell som jag tycker var kompakt och bra. ** => egen metod.
Man söker från höger till vänster tills man når en ** metod.

	Moo	Blue	Flute	Shoe
ts	moo**	moo	flute**	flute
m1	moo1**	blue1**	blue1	shoe1**
m2	moo2**	moo2	flute2**	flute1

Uppg 6.

```
==========================================
public class PotentialView extends JPanel {
    PotentialModel m;
    public PotentialView(PotentialModel m) {
        this.m = m;
        setPreferredSize(new Dimension(m.getSize(), m.getSize()));
    }
    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        Color[][] pic = m.getPotential();
        for(int i=0; i<pic.length; i++) {
            for(int j=0; j<pic[i].length; j++) {
                g.setColor(pic[i][j]);
                g.fillOval(i, j, 2, 2);
            }
        }
    }
} // end class

==========================================
Uppg 7)
public void readPotential(String file) throws IllegalArgumentException {
    Scanner in; // read in N point charges, assuming correct points
    try {
        in = new Scanner(new File(file));
    }
    catch(FileNotFoundException e) {
        throw new IllegalArgumentException("fil existerar ej");
    }
    int n = in.nextInt();
    arrayOfCharges = new Charge[n];
    for (int k = 0; k < n; k++) {
        double x0 = in.nextDouble();
        double y0 = in.nextDouble();
        double q0 = in.nextDouble(); // automatiska typkonvertering, man kan läsa en int
ocksåå
        arrayOfCharges[k] = new Charge(x0, y0, q0);
    }
    getPotential(); // behövde inte vara med på tentan
} // end readData

==========================================
Uppg 8)
public class PotentialControl extends JFrame {

    private PotentialModel m; // lagrar parametern till konstruktorn
    private PotentialView v;

    public PotentialControl() {
        setTitle("Potentials");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new BorderLayout()); // Create a Layout manager.
        setLocation(600, 50);
        m = new PotentialModel("charges2.txt");
        v = new PotentialView(m);
        JPanel buttonP = new JPanel();  // Buttonpanel
        JSlider slider;
// page breake
```

```
        // Create the buttons.
        JButton b1 = new JButton("greyscale continous");
        JButton b2 = new JButton("greyscale discrete");
        JButton b3 = new JButton("HSB color");
        JButton b4 = new JButton("HSb color discrete");
        b1.setActionCommand("b1");
        b2.setActionCommand("b2");
        b3.setActionCommand("b3");
        b4.setActionCommand("b4");
        b1.addActionListener(new setGrayscaleCont());
        b2.addActionListener(new setGrayscaleDisc());
        b3.addActionListener(new setHSBColor());
        b4.addActionListener(new setHSBcolorDisc());
        buttonP.add(b1);
        buttonP.add(b2);
        buttonP.add(b3);
        buttonP.add(b4);
        add(buttonP, BorderLayout.SOUTH);
        // Create the slider.
        slider = new JSlider(JSlider.VERTICAL, 20, 255, 255);
        slider.setMajorTickSpacing(50);  // Major tick every 50
        slider.setMinorTickSpacing(10);  // Minor tick every 10
        slider.setPaintTicks(true);      // Display tick marks
        slider.setPaintLabels(true);     // Display numbers
        slider.addChangeListener(new Listener());
        add(slider, BorderLayout.EAST);
        add(v, BorderLayout.CENTER);
        pack();
        setVisible(true);
    } // end constructor

    // Handle ActionEvent from buttons
    private class setGrayscaleCont  implements ActionListener {
        public void actionPerformed(ActionEvent e) {
            m.update(0); //greyscaleCont
            v.repaint();
        }
    }

    ... 4 till likadana

    /* alternativ med en lyssnarklass
    public void actionPerformed(ActionEvent e) {
        String ac = e.getActionCommand();
        try {
            if (e.getSource() instanceof JButton) {
                if ("b1".equals(ac)) {
        osv
    */
    // take care of slider changes
    public class Listener implements ChangeListener {
        public void stateChanged(ChangeEvent e) {
            if ( e.getSource() instanceof JSlider ) {
                JSlider s = (JSlider)e.getSource();
                double newGrade = s.getValue();
                m.setSpline((int)newGrade);
                m.update(4); // HSBslider
                v.repaint();
            }
        }
    }
    public static void main(String[] args) {
        new PotentialControl();
    }
}
```