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

Göteborg 14 dec 2009

Examination in

PROGRAMMERINGSTEKNIK F1 TIN211

DAY: TUESDAY DATE: 2009-12-15 TIME: 14.00-18.00 ROOM: M

Responsible teacher:	Erland Holmström tel. 1007, home 270358
Results:	Are sent by mail from Ladok.
Solutions:	Are eventually posted on homepage.
Inspection of grading:	The exam can be found in our study expedition after posting of results.
	Time for complaints about grading are announced on homepage after the result are published or mail me and we find a time.
Grade limits:	CTH: 3=26p, 4=36p, 5= 46p, max 60p
Aids on the exam:	**Bravaco, Simonson Java Programming From the Grounf Up**
	or Horstmann Java Concepts and A print of Chapter 18. or Skansholm Java direkt.

Observe:

- Start by reading through all questions so you can ask questions when I come. I usually will come after appr. 2 hours.
- All answers must be motivated when applicable.
- Write legible! Draw figures. Solutions that are difficult to read are not evaluated!
- Answer concisely and to the point.
- The advice and directions given during course must be followed.
- Programs should be written in Java, indent properly, use comments and so on.
- Start every new problem on a new sheet of paper.

Good Luck!

Problem 1. a) Förklara vad som orsakat följande felmeddelande.

```
int[] f = {1, 2, 3, 4};
...
NSquareSortMethods.java:20: operator || cannot be applied to
<nulltype>,boolean
    if ( f = null || f.length <= 1) {
```

b) Antag att vi har följande klasser:

```
public interface Vehicle  { ... }
public class Car implements Vehicle { ... }
public class Suv extends Car  { ... }
```

Vilka av följande är då korrekta satser?

```
Vehicle v = new Car();
Vehicle v = new Suv();
Vehicle v = new Vehicle();
Car c = new Suv();
Car c = new Vehicle();
Suv s = new Suv();
Suv s = new Car();
```

c) Förklara(säkerhets) problemet med följande kodsnutt:

```
public class Array {
  private int[] f = new int[5];
  ...
  public int[] getArray() {
    int[] fcopy = f;
    return fcopy;
  }
}
```

d) Varför är följande interface felaktigt? (det är minst 3 fel ...)

```
import java.awt.*;
public Interface HasColor {
  private Color col = new Color(255, 255, 255);
  public Color getColor() {
    return col;
  }
}
```

(10p)

Problem 2. Vad gör följande program? Beskriv också hur programmet fungerar.

(Swap(f, i, j) byter plats på elementen f(i) och f(j).)

```
public static void b(int[] f, int n) {
  if (n > 1) {
    for ( int i=0; i<=n-1; i++) {
      if ( f[i] > f[i+1] ) {
        swap(f, i, i+1);
      }
    }
    b(f, n-1);
  }
} // end b
```

(5p)

Problem 3. Testar fält, loopar.

Vi säger att en följd heltal a_0, a_1,..., a_{n-1} är växande om $a_0 < a_1 < a_2 < \ldots < a_{n-1}$. (Om $n = 1$ anses följden växande oberoende av a_0 och om $n = 0$ så är följden tom och anses då också växande.) Definiera en klass Increasing med tre metoder

```
public static boolean increasing(int[] f);
public static boolean increasing2(int[] f, int n) {
public static void main(String[] args);
```

De 2 första av dessa avgör om talen i argumentfältet utgör en växande följd. Den första är iterativ och den andra är rekursiv. Den sista är en main-rutin som testar de första metoderna genom att kommandoradsargumenten lagras i ett fält av heltal, metoderna increasing/increasing2 anropas med detta fält som argument samt resultatet från anropet skrivs ut.

Fel vid parsningen (dvs översättningen från sträng till tal) av argumenten skall hanteras med en lämplig exception. Exempel på användning:

```
% java Increasing 0 1 2 3
true true
% java Increasing 0 1 2 2 3
false false
% java Increasing 0 1 2 5 3
false false
% java Increasing 0 1 2 fem 3
fel i indata
```

(10p)

Problem 4. Testar uttryck, val, toString, klasskonstuktion. Dom följande tre uppgifterna kan lösas oberoende av varandra dvs i varje uppgift kan du förutsätta att dom andra två finns tillgängliga.

Du vill visa hur ett föremål faller från en höjd h till marken. Tyngdkraften gör att fallande objekt accelererar med $9.81 m/s$ om vi bortser från friktion med luften. Hastigheten som objektet faller med blir då $v = 9.81 * t$ där t är antalet sekunder från det att man släppte objektet.

Avståndet som objektet faller är $d = 4.9 * t^2$.

Om objektet börjar falla från en höjd h så är dess höjd efter t sekunder

$$\text{aktuell höjd} = h - d = h - 4.9 * t^2.$$

I nästa uppgift skall du göra ett grafisk gränssnitt för detta (så du kan få lite mer känsla för uppgiften genom att titta på figurerna där) men här skall du bara skriva resultatet på terminalen. Om vi släpper objektet från $180m$ så vill du ha utskriften (180 kan vara en konstant i koden så länge):

```
% java GravityMain
Time: 0.0s  Height= 180.0m  Velocity= 0.0m/s
Time: 1.0s  Height= 175.1m  Velocity= 9.8m/s
Time: 2.0s  Height= 160.4m  Velocity= 19.6m/s
Time: 3.0s  Height= 135.9m  Velocity= 29.4m/s
Time: 4.0s  Height= 101.6m  Velocity= 39.2m/s
Time: 5.0s  Height= 57.5m  Velocity= 49.0m/s
```

```
Time: 6.0s  Height= 3.5m  Velocity= 58.8m/s
Time: 7.0s Height= 0.0m  Velocity= 0.0m/s
```

Du bestämmer dig för att göra en klass som håller reda på en ögonblicksbild av objektets resa mot jorden (en av tidsuppgifterna ovan alltså).

```
public class Gravity {
  public Gravity(double dropHeight,
                                double timeResolution){...
  public void tick() {...}
  public String toString() {...}
  public double getTime() {...}
  public double getHeight() {...}
  public double getVelocity() {...}
} // end Gravity
```

Konstruktorn tar dels höjden man släpper objektet från och dels tidsupplösningen dvs hur ofta man skall göra en utskrift. I exemplet har vi gjort en utskrift varje sekund (`timeResulution` = 1). Metoden `tick` gör att objektet avancerar till nästa ögonblicksbild, dvs "klockan" ökar med "`timeResolution`".

De övriga metoderna används för att hämta ett tillstånd tex för utskrift. Speciellt skall `toString` (automatiskt) kunna användas för utskrifterna ovan i traditionell java "stil".

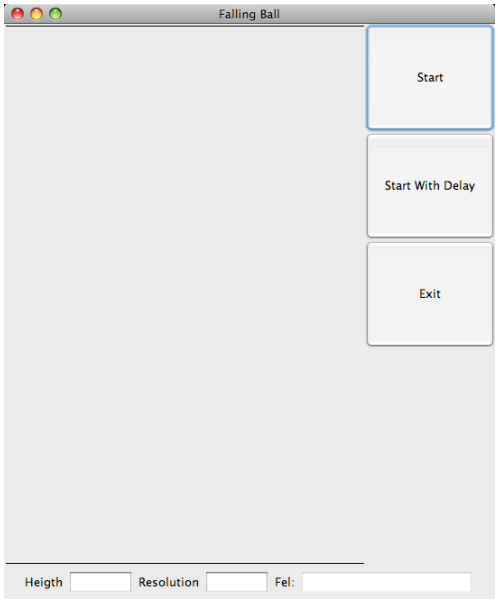
Observera också att när föremålet tar mark så blir höjd och fart = 0 och tiden "stannar" på raden efter, se de 2 sista radera i utskriften ovan. Mellan sek 5 och 6 faller objektet 54m, mellan sek 6 och 7 bara 3.5m, sen tar det mark. Du måste fixa till nollorna manuellt i koden.

Skriv klassen.

Skriv också ett huvudprogram som använder klassen och fixar utskriften ovan.

(10p)

Problem 5. Testar om du kan rita med en Jpanel och enkel klasskonstruktion samt hur man representerar relationer.



Det är ju lite tråkigt med utskrifterna ovan och du minns fysiklektionerna då man använde en stroboskoplampa för att åskådliggöra förloppet. Nu är det ju svårt att göra det när man släpper från 180m höjd men man kan ju simulera med ett program med lite snyggare gränssnitt.
Du bestämmer dig alltså för att skriva en klass som ärver en Jpanel i vilken du kan rita upp förloppet.
I bilderna till höger/nedan är det ritytan med cirklarna (som symboliserar objektet vi släpper) och texten (som är samma text som i uppgiften ovan) som är den här klassen. Klassen innehåller bara en konstruktor, med parametrarna `double dropHeight, double resolution, double delay`, och en metod som ritar. Kalla klassen "`BallWindow`" så blir rättarna glada. Delay-variabeln

används inte ännu, den var tänkt som en framtida utveckling av programmet. Du använder dig naturligtvis av klassen Gravity från förra uppgiften.

Bra storlek på den här panelen är 400x600 (det är enda storleken som behöver sättas).

Några användbara metoder i klassen Graphics är:

drawOval(int x, int y, int width, int height)

fillOval(int x, int y, int width, int height)

drawString(String str, int x, int y)
Draws the text given by the specified string, using this graphics context's current font and color. (Denna metod skriver alltså text i det grafiska fönstret med start i position x, y)

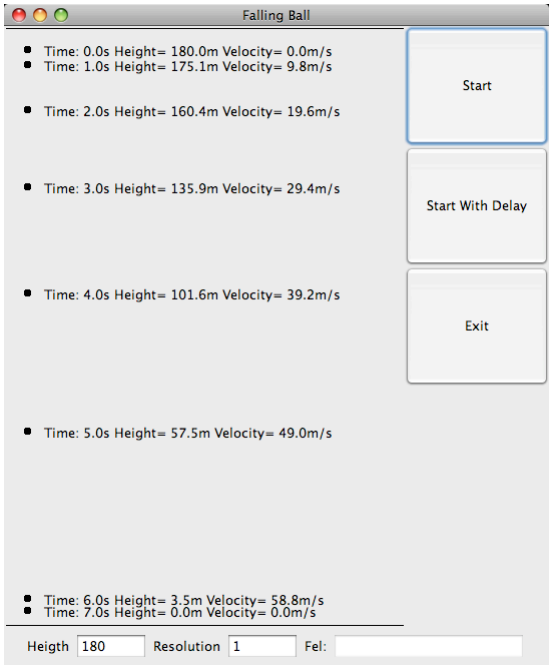
(10p)

Problem 6. Testar: grafiskt användargränssnitt, händelsehantering.

Dags att skapa ramen, knappar mm enligt bilden nedan. Tanken är att man skriver in "släpp"-höjden i fältet efter Height, dvs 180 i exemplet nedan, och eventuellt också tidsupplösningen. När man sedan klickar på "Start"- knappen så ritas förloppet ut.

Om man inte skriver något i Heigth-fältet så skall man få en felutskrift (i textfältet efter texten ”Fel:”) och sedan får man försöka igen och om man inte skriver något i Resolution-fältet så skall programmet använda defaultvärdet 1.

Klickar man på "Start with delay" så skall Panelen initieras med delay=10 (men panelen gör inget åt det i nuläget), annars initieras panelen med med delay=0 (dvs vid klick på "Start").



Klickar man på "Exit", ja det naturliga beteendet förväntas.

Kalla klassen "FrameAndMain".

Några tips:

Tomrummet nedanför knapparna kan man åstadkomma genom att ha en gridlayout som har mer än 3 grids och så fyller man den inte.

Du måste hantera att användaren inte skriver något alls i textfälten innan han trycker på "Start" och att det han skriver är fel tex ”1,0” istället för ”1.0”. Det är ok att kasta och fånga en exception i klassen för detta men det kan enkelt lösas utan också.

Negativa tal är korrekta tal men skall inte resultera i någon utritning så klart.

(15p)

(Den här uppgiften är inte så svår som poängen antyder utan den höga poängen motiveras av att det är mycket att skriva.)

Lösningsskisser tentamen F1 programmeringsteknik 2009-12-15

U1a)
if (f = null || f.length <= 1) {
f=null är ett uttryck och förutom att f blir lika med null så har uttrycket null som värde.
det innebär att när f=null beräknats så blir if satsen
if (null || f.length <= 1) {
Nu har vi alltså ett uttryck (med)om jag ersätter värdena med värdenas typer)
<nulltype> || boolean och "||" vill ha booleaner på bägge sidorna.

U1b)
```
Vehicle v = new Car(); // ok
Vehicle v = new Suv(); // ok
Vehicle v = new Vehicle();  // Vehicle is abstract; cannot be instantiated
Car c = new Suv(); // ok
Car c = new Vehicle(); // Vehicle is abstract; cannot be instantiated
Suv s = new Suv(); // ok
Suv s = new Car();   // incompatible types
```

U1c)
Man returnerar en pekare till fältet och då kan man ändra fältet utanför klassen.

U1d)
Interface skall vara interface (litet i)
instansvariabler är inte tillåtna (och inte private heller)
implementeringar av metoder är inte tillåtet

U2)
Om n är index för sista positionen: (dvs n=f.length-1)
Programmet sorterar fältet med en udda metod som påminner bubbelsort.

Först bubblas det största elementet till sista positionen i loopen.
Sedan sorteras resten av fältet rekursivt dvs det näst minsta elementet bubblas till näst sista positionen osv.

om n<f.length-1 så sorteras bara en del av fältet enligt ovan (upp till n)
om n>= f.length så fås en exception

Sen finns det även en bugg i koden som gör att programmet inte gör rätt om det minsta talet finns sist.

U3)

```
 public class Increasing {

    public static boolean increasing(int[] f) { //2
        if ( f != null ) {
            for( int i=0; i < f.length-1; i++ ) {
                if( f[i] >= f[i+1] ) {
                    return false;
                }
            }
        }
        return true;
    }

    public static boolean increasing2(int[] f, int n) { //3
        if ( f == null || n<=1 ) {
            return true;
        } else if( f[n-1] >= f[n] ) {
                 return false;
        } else {
            return increasing2(f, n-1);
        }
    }

    public static void main(String[] args) { // 5
        int[] f = new int[args.length];
        try {
            for(int i=0;i < f.length; i++) {
                f[i] = Integer.parseInt(args[i]);
            }
            System.out.println(increasing(f));
            System.out.println(increasing2(f, f.length-1));

        } catch (NumberFormatException e) {
            System.out.println("fel i indta");
        }
    }
 }
```

U4+5+6) se separata filer

```
public class Gravity   {

    private double time = 0;  // time in seconds
    private double dropHeight =0;
    private double velocity = 0;
    private double initialHeight =0;
    private double resolution = 1;

    public Gravity(double dropH, double timeResolution) {
        initialHeight = dropH;
        dropHeight = dropH;
        resolution = timeResolution;
        time = 0;
        velocity = 0;
    }

    public void tick() {
        time = time + resolution;
        if ( dropHeight > 0 ) {
            dropHeight = initialHeight - 4.9*time*time;
            velocity = 9.81*time;
            if ( dropHeight <= 0 ) {
                dropHeight = 0;
                velocity = 0;
            }
        }
    } // end tick

    public String toString() {
        return  "Time: " + time + "s" +
                " Height= " + (int)(dropHeight*10)/10.0 + "m" +
                " Velocity= " + (int)(velocity*10)/10.0 + "m/s";
    } // end toString

    public double getTime() {
        return time;
    } // end getTime
    public double getHeig() {
        return dropHeight;
    } // end getHeight
    public double getVelocity() {
        return velocity;
    } // end getVelocity

} // end Gravity
```

```
// I separat fil
public class GravityMain {

    public static void main (String [] args) {
        Gravity falling = new Gravity(180, 1);
        System.out.println( falling );
        while( falling.getHeig() > 0 ) {
            falling.tick();
            System.out.println( falling );
        }
    }
} // end GravityMain
```

```
import java.awt.*;
import javax.swing.*;
// ett fönster för att rita ut den fallande bollen
public class BallWindow extends JPanel {
    private static final int WIDTH = 400;
    private static final int HEIGHT = 600;
    private double height = 0;
    private double resolution=1; // denna behöver väl inte sparas här
egentligen
    private double delay = 0;
    private Gravity fallingObject = null;
    private double scaleFactor = 1;

    public BallWindow(double dropHeight, double resolution, double delay ) {
        setPreferredSize(new Dimension(WIDTH, HEIGHT));
        this.height = dropHeight;
        this.resolution = resolution;
        this.delay = delay;
        fallingObject = new Gravity(height, resolution);
        repaint(); // onödig här
    }

    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        int panelWidth  = getWidth(); // dessa 2 rader behövs inte
        int panelHeight = getHeight(); // om pack tar hänsy till
setPreferredSize
        g.drawLine(2, 2, WIDTH-2, 2); // detta är också överkurs
        g.drawLine(2, HEIGHT-2,  WIDTH-2, HEIGHT-2);
        scaleFactor = (panelHeight-40)/height;
        int px = 20;
        int py = (int)(panelHeight - fallingObject.getHeig()*scaleFactor)-20;
        //System.out.println(   fallingObject ); // to screen
        g.fillOval(px, py, 6, 6);
        g.drawString(fallingObject.toString(), px + 20, py+10);
        while( fallingObject.getHeig() > 0 ) {
            fallingObject.tick();
            //System.out.println(   fallingObject );  // to screen
            px = 20;
            py = (int)(panelHeight - fallingObject.getHeig()*scaleFactor)-20;
            g.fillOval(px, py, 6, 6);
            g.drawString(fallingObject.toString(), px + 20, py + 10);
        }
    }
} // end BallWindow
```

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class FrameAndMain extends JFrame implements ActionListener {
    private JTextField jtf1, jtf2, jlbl;
    private BallWindow fallingW;
    private int delay = 0;

    public FrameAndMain() {
        // Panel p1 to hold text fields and labels
        JPanel p1 = new JPanel();
        p1.setLayout(new FlowLayout()); // default
        p1.add(new JLabel("Heigth"));
        p1.add(jtf1 = new JTextField(5));
        p1.add(new JLabel("Resolution"));
        p1.add(jtf2 = new JTextField(5));
        p1.add(new JLabel("Fel:"));
        p1.add(jlbl = new JTextField(15));
        jlbl.setEditable(false);

        // Panel p2 to hold buttons
        JPanel p2 = new JPanel();
        p2.setLayout(new GridLayout(5,1));
        JButton jbtStart, jbtStartDelay, jbtExit;
        p2.add(jbtStart = new JButton("Start"));
        p2.add(jbtStartDelay = new JButton("Start With Delay"));
        p2.add(jbtExit = new JButton("Exit"));

        // Register listeners
        jbtStart.addActionListener(this);
        jbtStartDelay.addActionListener(this);
        jbtExit.addActionListener(this);

        fallingW = new BallWindow(0, 1, 0);

        // the frame
        setTitle("Falling Ball");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        // Add panels to the frame
        setLayout(new BorderLayout());
        add(p1, BorderLayout.SOUTH);
        add(p2, BorderLayout.EAST);
        add(fallingW, BorderLayout.CENTER);
        setVisible(true);
        pack();
```

```
    }

    public void actionPerformed(ActionEvent e) {
        try {
            double height;
            double resolution;
            // Handle button events
                String actionCommand = e.getActionCommand();
            if ("Exit".equals(actionCommand)) {
                System.exit(0);
            } else if ("Start".equals(actionCommand.substring(0,5))) {
                String nbr1 = jtf1.getText();
                String nbr2 = jtf2.getText();
                if (nbr1 == null || nbr1.length()==0 ) {
                    throw new IllegalArgumentException("Heigth must be
given");
                }
                if ( nbr2 == null || nbr2.length()==0) {
                    nbr2 = "1";
                }
                height     = Double.parseDouble(nbr1.trim());
                resolution = Double.parseDouble(nbr2.trim());
                if (height < 0 ) height = 0;
                if ("Start With Delay".equals(actionCommand)) {
                    delay = 10;
                }
                jlbl.setText("");
                fallingW = new BallWindow(height, resolution, delay);
                //funkar inte om man inte adderar den nya komponeneten!!
                this.add(fallingW, BorderLayout.CENTER);
                validateTree(); // denna rad behövs ej på tentan
            }
        }
        catch ( IllegalArgumentException ex ) {
            // NumberFormatException som kastas av parseDouble ärver
            // IllegalArgumentException så den hamnar här också
            jlbl.setText(ex.getMessage());
            // efter denna kan man mata in nya tal och köra igen
        }
    }

    public static void main (String [] args) {
        FrameAndMain fam = new FrameAndMain();
    }
} // end GraphicMain
```