DATAVETENSKAP
Göteborg (2007-01-08)

TENTAMEN

CTH

2006-12-22

TENTAMEN I
PROGRAMMERINGSTEKNIK F1

DAG: **TORSDAG** TID: **8.30-12.30** SAL:**V**

Ansvarig:	Erland Holmström tel. 1007, hem 270358.
Resultat:	Anslås den **2007-01-21**
Lösningar:	Anslås (eventuellt) på hemsidan 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 - 28p, 4 - 36p, 5 - 46p, max=60.** (Siffror inom parentes anger maximal poäng på varje uppgift. I vissa fall ges även ungefärlig poängfördelning inom uppgiften.)
Hjälpmedel:	Horstman Java Concepts eller Skansholm Java Direkt

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.

a) Vad innebär det att man deklarerar en medlemsvariabel som static final? Vilka fördelar har detta?

b) Vad är en typ i ett programmeringsspråk? Skillnad mellan typ och klass?

c) Antag följande programskal:

```
public class Param {
     static int var0;
     static String var1;
     int var2;
     String var3;
     static String enMetod(int var4, String var5) {
          int var6;
          String var7;
          ...(1)
          var4 = 8;
          var5 = "hobbe";
     }
     void enMetodTill(int var8,) {
          int var9;
          ...(2)
     }
     public static void main(String[] args) {
          int var10;
          String var11;

          ...(3)
          if ( nånting ) {
               int var10=10;
               ...(4)
               var11 = "kalle";
               String x = enMetod(var10, var11);
               ...(5)

          }

          for (int var12 = 5; var12 <= b; var12++) {//
               ...(6)
          }
          ...(7)
     }
}
```

1) Beskriv de olika variablernas (varX) “scope” dvs var dom är åtkomliga

Ange om kompilatorn kommer att protestera mot några av deklarationerna?.

2) För vart och ett av de numrerade ställena (1-7) skall du ange värdet av de variabler som är åtkomliga dvs i de flesta fallen vad dom initierats till.

Gör lämpligen en tabell där streck betyder att variabeln inte är känd, ett värde är dess värde och “K” innebär att kompilatorn skulle protestera

var0, var1, ...

(1)

(2)

...

(10p)

Uppgift 2. Skriv en *rekursiv* metod `String nat2base(int nat, int base)` som omvandlar ett naturligt tal `nat` till ett tal i basen `base` i form av en sträng.

Ex: `nat2base(17, 10)` ger strängen "17" och `nat2base(17, 2)` ger "10001"

`nat2base` behöver bara klara 2 <= `base` <= 10 och `nat` >= 0. Om detta inte är uppfyllt kastas en exception `IllegalArgumentException` med ett lämpligt felmeddelande.

(Detta är en enkel variant av `Integer.toString(nat, base)`. Det är ok att använda `Integer.toString` för att omvandla *enstaka tecken* dvs tex 5 till "5" men för övrigt får du inte använda några API metoder. Man kan också använda ascii-tabellen för att göra det.)

(Man omvandlar ett decimalt tal till ett binärt tal genom att upprepade gånger dividera det decimala talet med 2 dvs med basen på det tal man skall omvandla till. Det binära talet får man genom att ta resterna vid divisionerna, minst signifikanta siffran fås först. Fråga om du inte förstår detta för det är inte det som frågan skall testa.)

(8p)

Uppgift 3. Skriv metoden `primtal(int p)` som avgör om `p` är ett primtal. `primtal` skall ge ett rimligt svar för alla indata.

(4p)

Uppgift 4. **Primtalspalindromer.** Talet 151 är en primtalspalindrom eftersom det är både ett primtal och en palindrom (en palindrom är samma tal (ord) oavsett om man läser fram eller baklänges).

Skriv ett program som läser in två *positiva* heltal, a och b, samt skriver ut alla primtalspalindromer mellan a och b. Utskriften skall ske i fem kolumner enligt nedan.

Följande underprogram skall ingå: `reverse` som "vänder" på en sträng dvs `reverse`("kalle") = "ellak", och `nat2base` som omvandlar ett naturligt tal till en sträng samt `primtal` som avgör om ett tal är primtal (se föregående uppgifter).

Exempel: Om a=1 och b = 919 så skall utskriften bli i fem kolumner enligt:

```
Primtalspalindromer mellan 1 and 919:
  2       3       5       7      11
101     131     151     181     191
313     353     373     383     727
757     787     797     919
```

(8p)

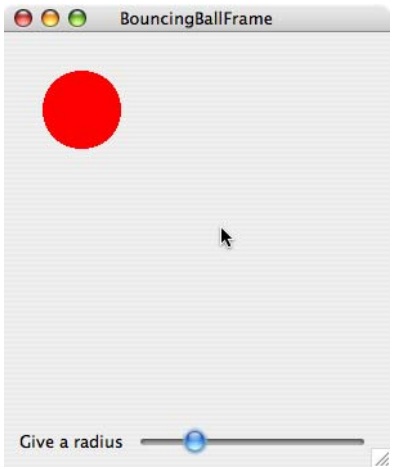
Uppgift 5. Du skall skriva en klass `BallShape` som implementerar interfacet `Shape` och som är en boll med röd färg. En boll har förutom egenskaperna i `Shape` en radie. Konstruktorn skall ta x,y och radie som parametrar. Det räcker om du implementerar interfacet, konstruktorn och getter/setter för radien.

```
public interface Shape {
     // drawing color
     public void setColor(Color c);
     // width and height of the component
     // i.e. size of smallest enclosing rectangle
     public int getWidth();
     public int getHeight();
     // position of upper left corner
     public int getX();
     public int getY();

     public void draw(Graphics g);
}
```

(8p)

Uppgift 6. Antag nu att du skall åstadkomma figuren till höger. Den röda bollen från förra uppgiften ligger i en JPanel (`BallPanel` se nästa uppgift, antag att `BallPanel` finns i denna uppgiften) och nedanför den ligger en JLabel och en JSlider. När man flyttar slidern så ändras cirkelns radie, minsta värde kan vara 10 och största 100 på slidern, det aktuella värdet i figuren är 30 på såväl cirkel som slider. Slidern skapas här men ändringar i slidern hanteras i `BallPanel` (som alltså skall lyssna på dessa).



En slider fungerar ungefär som en knapp men använder en `ChangeListener` istället för en ActionListener och metoden `stateChanged(ChangeEvent e)` istället för `actionPerformed(ActionEvent e)`

Den här klassen innehåller också en `main` metod som du också skriver. Döp klassen till `BallFrame`.

(10p)

Uppgift 7. Nu är det dags för panelen i vilken bollen ritas upp (`BallPanel`). Konstruktorn skall ta vidd och höjd som parametrar och den skapar en boll. Klassen lyssnar också på ändringar i slidern och ändrar cirkelns storlek när detta sker. Också ritar den upp sig. (Tänk på att en `ChangeEvent` kan bero på andra saker än en ändring i slidern)

(12p)

Uppg 1
a) static innebär att variablen är gemensam för alla objekt
+ om man behöver data som ändras för alla objekt när ett objekt ändrar den
tex vill kunna räkna antalet skapade objekt.
final innebär att den är en konstant som inte kan ändras
+ man slipper samma magiska tal på olika ställen i koden (viktigast)
+ men framförallt så är den enkel att ändra :-) dvs om man vill ändra
det konstanta värdet så behöver man bara göra det på ett ställe.
+ spar minne
att den inte kan ändras är kanske en fördel men det är ju det definitionen säger

b) En typ bestämmer en variabels/objekts egenskaper.
(tex minneskrav, operationer/metoder).
En klass är en beskrivning av typen för ett objekt.

c.1)

var0 - hela klassen
var1 - hela klassen
var2 - hela klassen men inte inuti statiska metoder dvs bara i enMetodTill
var3 - hela klassen men inte inuti statiska metoder dvs bara i enMetodTill
var4 - bara inuti enMetod
var5 - bara inuti enMetod
var6 - bara inuti enMetod
var7 - bara inuti enMetod
var8 - bara inuti enMetodTill
var9 - bara inuti enMetodTill
var10- första deklarationen: inuti main, andra deklarationen bara inuti if satsen
var11- inuti main
var12- inuti for loopen

Kompilatorn klagar bara på dubbeldeklarationen av "var10" (när det gäller variabler).
Antingen måste den inre deklarationen ändras till var10 = 10 dvs en tilldelning,
eller också måste man byta namn och då blir tabellvärdet NS på plats 6+7 i tabellen
för där är det den första deklarationen som gäller.

För att kunna kompilera måste man fixa en del syntaxfel:
-lägga till ";//" vid varje ...(x)
-ta bort ",", vid var8,
-lägga till "int b = 12;" före loopen
-ändra if (nånting) till if (true)

-ändra int var10=10; till var10=10; ELLER int var10x=10;
-lägga till "return ("something");" sist i enMetod

För att få utskrifter (enligt nedan) måste man lägga till sist i main:
Param y = new Param(); // för att kunna anropa
y.enMetodTill(8); // för att kunna anropa och få utskrift

c.2)
NS = non-static variable varX cannot be referenced from a static context
  dvs odefinierad/okänd (det som är "-" i tesen)
CFS= cannot find symbol dvs odefinierad/okänd (det som är "-" i tesen)
NI = variable varX might not have been initialized (dvs känd men utan värde)

Reglerna säger att instans och klassvariabler initieras till ett defaultvärde 0 eller null.
Lokala varibler i en metod initieras inte och när man då försöker läsa den får man NI.
Observera att "null" ŸR ett värde.

	var0	var1	var2	var3	var4	var5	var6	var7	var8	var9	var10	var11	var12
(1)	0	null	NS	NS	10	kalle	NI	NI	CFS	CFS	CFS	CFS	CFS
(2)	-"-	-"-	0	null	CFS	CFS	CFS	CFS	8	NI	CFS	CFS	CFS
(3)	-"-	-"-	NS	NS	CFS	CFS	CFS	CFS	CFS	CFS	NI	NI	CFS
(4)	-"-	-"-	NS	NS	CFS	CFS	CFS	CFS	CFS	CFS	10	NI	CFS
(5)	-"-	-"-	NS	NS	CFS	CFS	CFS	CFS	CFS	CFS	10	kalle	CFS
(6)	-"-	-"-	NS	NS	CFS	CFS	CFS	CFS	CFS	CFS	10/NS	kalle	5
(7)	-"-	-"-	NS	NS	CFS	CFS	CFS	CFS	CFS	CFS	10/NS	kalle	CFS

(1)======= kommer ut som 3e utskrift
var0= 0
var1= null
var2= non-static variable var2 cannot be referenced from a static context
var3= non-static variable var3 cannot be referenced from a static context
var4= 10
var5= kalle
var4-12= cannot find symbol
=======
(2)======= kommer ut sist (7)
var0= 0
var1= null
var2= 0
var3= null

var4-7= cannot find symbol
var8= 8
var9= variable var9 might not have been initialized
var10-12= cannot find symbol
=======
(3)======= kommer ut som 1a utskrift
var0= 0
var1= null
var2= non-static variable var2 cannot be referenced from a static context
var3= non-static variable var3 cannot be referenced from a static context
var4-9= cannot find symbol
var10= variable var10 might not have been initialized
var11= variable var11 might not have been initialized
var12= cannot find symbol
=======
(4)======= kommer som 2a utskrift
var0= 0
var1= null
var2= non-static variable var2 cannot be referenced from a static context
var3= non-static variable var3 cannot be referenced from a static context
var4-9= cannot find symbol
var10= 10
var11= variable var11 might not have been initialized
var12= cannot find symbol
=======
(5)======= kommer som 4e utskrift
var0= 0
var1= null
var2= non-static variable var2 cannot be referenced from a static context
var3= non-static variable var3 cannot be referenced from a static context
var4-9= cannot find symbol
var10= 10
var11= kalle
var12= cannot find symbol
=======
(6)======= kommer som 5e utskrift
var0= 0
var1= null
var2= non-static variable var2 cannot be referenced from a static context
var3= non-static variable var3 cannot be referenced from a static context
var4-9= cannot find symbol
var10= 10
var11= kalle

var12= 5
=======
(7)======= kommer som 6e utskrift
var0= 0
var1= null
var2= non-static variable var2 cannot be referenced from a static context
var3= non-static variable var3 cannot be referenced from a static context
var4-9= cannot find symbol
var10= 10
var11= kalle
var12= cannot find symbol
=======

```
import java.util.*;
public class Primpalin {
// uppg 4 plus U2 (nat2base) och U3 (primtal)
	/** ********** ********** ********** ********** **********
	* Omvandlar ett naturligt tal 'nat' till ett tal i basen base
	* Det nya talet representeras i form av en sträng
	* endast om 2 <= base <= 10 och nat >= 0
	* Detta är en enkel variant av Integer.toString(nat, base);
	********** ********** ********** ********** ********** */
	// Uppg 2
	static String nat2base(int nat, int base) {
		if (base < 2 || base > 10 || nat < 0) {//2
			throw new IllegalArgumentException("2 <= base <= 10 and nat >= 0: "
				+ " base= " + base + " " + "nat= " + nat);
		}
		if (nat < base) {//
			return Integer.toString(nat);//
		} else {
			return nat2base(nat/base, base) + Integer.toString(nat%base);//4
		}
	} // nat2base
	// ========== ========== ========== ========== ==========
	static String reverse(String str) {     //2
		if (str.length() <= 1) {
			return str;
		} else {
			return  reverse( str.substring(1) ) + str.charAt(0);
		}
	}
	// ========== ========== ========== ========== ==========
	static boolean palindrom(String str) {
		return ( str.equals(reverse(str)) );
	}
	// ========== ========== ========== ========== ==========
	//Uppg 3
	static public boolean primtal(int p) {
		// alla tal skall klaras av
		int rotenUrP;
		if (p == 2 || p == 3) return true; //
		if (p % 2 == 0 || p % 3 == 0 || p <= 1)  return false;//
		rotenUrP = (int)Math.floor(Math.sqrt((double)p)); // floor behövs nog inte
		for (int delare = 5; delare <= rotenUrP; delare = delare+2) {//
```

```
                if (p % delare == 0) { //
                        return false;
                }
        }
        return true;
    } // end primtal
        // ========== ========== ========== ========== ==========

        public static void main(String[] args) {
                int a, b;
                //int antal = 0;

                Scanner in = new Scanner(System.in);//
        System.out.println("ange a");
        a = in.nextInt();//
                System.out.println("ange b");
        b = in.nextInt();
        System.out.println();
                if ( a<0 || b<0 ) { //
                        System.out.println("Primpalin: a och b måste vara >= 0 ");
                        System.exit(0);
                }

                System.out.println("Primtalspalindromer mellan " + a + " och " + b + ":");
                int column = 0;
                for (int i = a; i <= b; i++) {//
                        if ( primtal(i) && palindrom(nat2base(i, 10)) ) {//3
                                if (column == 5) { // new line? //2
                                        column = 1;
                                        System.out.println();
                                } else {
                                        column = column + 1;
                                }
                                System.out.print(i + "\t");//
                        }
                } // end loop
                System.out.println();
        } // end main
}
```

```
import java.awt.*;
public class BallShape implements Shape {
	private int x, y; // position of the ball, upper left corner //
	private int width, height; // size of smallest enclosing rectangle
	private int radius; // default
	private Color ballColor = Color.red;

	public BallShape(int x, int y, int radius) { //
		setRadius(radius);
		this.x = x;
		this.y = y;
	}

	public int getWidth() { return width; }
	public int getHeight() { return height; }
	public int getX() { return x; }
	public int getY() { return y; }

	public void setRadius(int r) { //2
		radius = r;
		width =  2 * radius;
		height = 2 * radius;
	}

	public int getRadius() { return radius; }//

	public void setColor(Color c) { ballColor = c; }//

	public void draw( Graphics g) {
		g.setColor(ballColor);//
		g.fillOval( x, y, width, height );//
	}
}
```

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

public class BallFrame extends JFrame {
      private static final int WIDTH = 300;
   private static final int HEIGHT = 300;

      public BallFrame() {
            // ta hand om framen
            setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);//
      setTitle("BouncingBallFrame");
      setLayout(new BorderLayout());//

            //skapa en panel
            BallPanel p = new BallPanel(WIDTH, HEIGHT);//
            add(p, BorderLayout.CENTER);//

            JLabel jl = new JLabel("Give a radius");//

            JSlider slider = new JSlider(10, 100, 30);//
            slider.addChangeListener(p);//

            JPanel jp = new JPanel();
            jp.add(jl);//
            jp.add(slider);//
            add(jp, BorderLayout.SOUTH);//
   }

      public static void main(String[] args) {
            BallFrame w = new BallFrame();//
            w.pack();
      w.setVisible(true);//
      }
}
```

```
import java.awt.*;
import java.awt.Dimension;
import java.awt.Color;
import javax.swing.*;
import javax.swing.event.*;
import java.awt.event.*;

public class BallPanel extends JPanel implements ChangeListener {//
	BallShape b1;

	public BallPanel(int width, int height) {//
		//super();
		setPreferredSize(new Dimension(width, height));//
		// startposition f–r bollen
		int radius = 30;
		int x = 30;
		int y = 30;
		b1 = new BallShape(x, y, radius);//
		b1.setColor(Color.red);//
	}

	public void stateChanged(ChangeEvent e) {
		int value = b1.getRadius();//
		Object obj = e.getSource();//
		if (obj instanceof JSlider) {//
			JSlider js = (JSlider) obj;//
			value = js.getValue();//
		}
		b1.setRadius(value);//
		repaint();//
	}
	public void paintComponent( Graphics g ) {//
		super.paintComponent(g);//
		b1.draw(g); //
	}
}
```