UMEÅ UNIVERSITET
Inst. för Datavetenskap

Praktiskt prov Datavetenskap

Uppgift (max poäng)	Inlämnad	Poäng
1 (6)		
2 (6)		
3 (4)		
4 (4)		
5 (6)		
6 (8)		
Summa		

Högst 2 av dessa får inlämnas

Kurs : Programmering i Java 5p.
Sommaruniversitetet

Datum : 010630

Namn (texta) :_____

Personnr :_____

E-mail adress (@cs.umu.se) :_____

_____

Namnteckning

UMEÅ UNIVERSITET 901 87 UMEÅ Tel.: 090-786 77 08 Fax: 090-786 61 26 e-mail: marie@cs.umu.se

PRAKTISKT PROV

PROGRAMMERING I JAVA, 5P
SOMMARUNIVERSITETET

Datum	:	010630
Tid	:	9-12
Hjälpmedel	:	Allt. Kommunikation med andra personer (direkt eller indirekt) är dock förbjuden, som t ex mail, mobiltelefon, gemensama kataloger etc.
Antal uppgifter	:	4
Totalpoäng	:	26 (halva poängtalet krävs normalt för godkänt)

- Provet består av 4 uppgifter : 1 och 2 samt 2 fritt valda uppgifter bland 3,4,5 & 6.
- Kryssa för de uppgifter du lämnar in.
- Källkod skrivs ut i ett ickeproportionellt typsnitt (t.ex. `courier`).
- Namn, personnummer, användarnamn och sökvägen till filen / filarna skall finnas på all källkod.
- Lösningarna skall vara snyggt och prydligt nedskrivna. Tankegången skall vara lätt att följa. Alla antaganden som inte är uppenbara skall redovisas.

OBS! Kolla att din dator fungerar innan du börjar på allvar.

Lycka till!

Uppgift 1 (6 poäng)

I denna uppgift skall du implementera en metod **kollaKontrollsiffra** som testar om kontrollsiffran i ett personnummer är korrekt. Metoden skall ta ett argument (som kan vara av valfri typ) som representerar ett personnummer, och den skall returnera en boolean, true om allt är ok, false annars. De enklaste sätten att representera personnummer är som heltal eller String.

Följande main-metod skall fungera ihop med din metod **kollaKontrollsiffra**:

```
public static void main(String[] args)
{
  long pnr = 5511268996L; // eller annan typ (valfritt)
                          // `L´ står för `long´ annars skulle det bli en int

  if (kollaKontrollsiffra(pnr))
    System.out.println("Rätt kontrollsiffra");
  else
    System.out.println("Fel kontrollsiffra");
}
```

Metoden för att beräkna kontrollsiffran ser ut så här:

- Ta bort sista siffran (kontrollsiffran)
- Multiplicera alla enskilda siffror med omväxlande 2 och 1, så att första siffran (den till vänster) multipliceras med 2, den andra med 1, den tredje med 2 igen, osv.
- Om något resultat av en multiplikation blir tvåsiffrigt, beräkna siffersumman (dvs. lägg ihop entalssiffra och tiotalssiffra) Exempel: 8 * 2 blir 16, lägg ihop 1 + 6 = 7.
- Lägg ihop alla resultat av multiplikationerna
- Beräkna närmast högre tiotal, och dra bort summan från det. Exempel: summa = 37, närmast högre tiotal är 40, 40 - 37 = 3.
- Jämför resultatet med kontrollsiffran, om de är lika är personnumret korrekt.

Ett exempel:

Kontrollsiffra

pnr =	5	5	1	1	2	6	8	9	9	(6)
mult. med	2*	1*	2*	1*	2*	1*	2*	1*	2*	
resultat	10	5	2	1	4	6	16	9	18	
siffersumma	1	5	2	1	4	6	7	9	9	
totalsumma	44									

närmast högre tiotal =50, `50-44 = 6` som är lika med kontrollsiffran

Uppgift 2 (6 poäng)

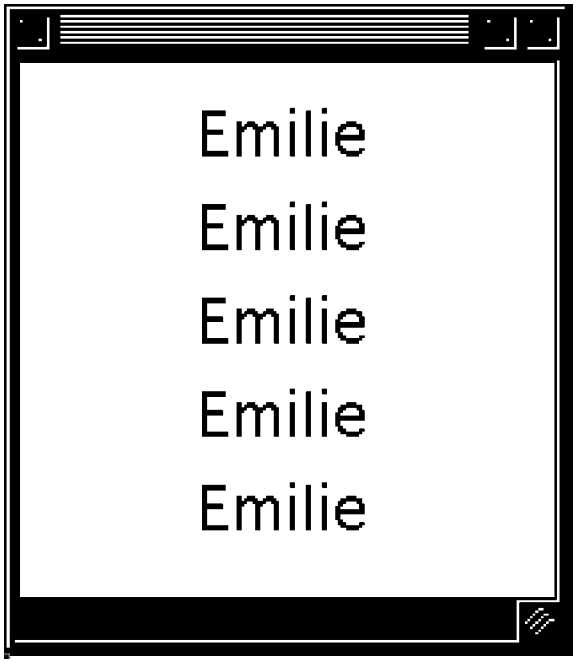
I denna uppgift skall du designa (ej implementera) en Java applikation som realiserar ett stoppur. Stoppuret skall visa timmar, minuter och sekunder i stilen `12 : 34 : 56`. Det skall finnas fyra knappar för att starta, stoppa, återuppta och nollställa tidsräkningen.

Beskriv alla klasser, metoder och attribut som ingår i designen. Du får göra det med hjälp av UML diagram eller som halvfärdiga Java klasser. Alla klasser, metoder och attribut skall kommenteras.

I den följande delen får du lämna in högst två av uppgifterna

Uppgift 3 (4 poäng)

Implementera en applet som skriver ut ditt namn flera gånger i olika färg.
Här ser du ett exempel :



Uppgift 4 (4 poäng)

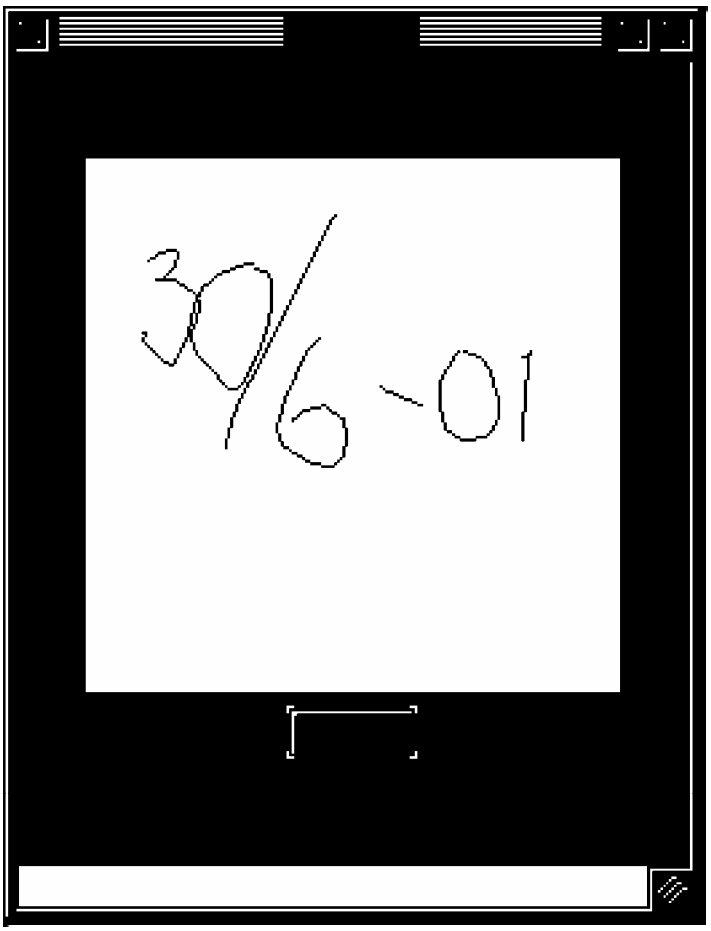
Implementera en metod **search** som söker igenom ett fält med heltal av valfri längd. Metoden skall returnera indexen på det sökta elementet om elementet finns med i fältet och −1 om det sökta elementet i nte finns i fältet.

Följande main-metod skall fungera ihop med din metod **search**:

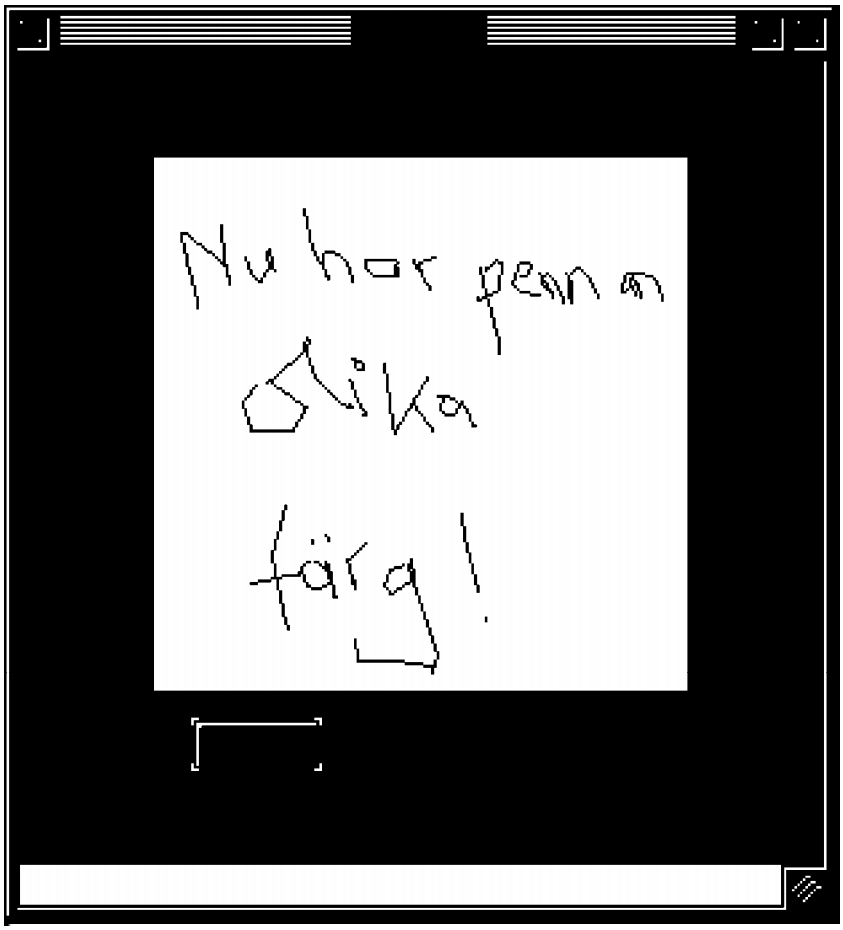
```
public static void main (String[] args)
 {
    int[] array1 = {…};

    int i = 0;
    int index = -1;

    int index = search (i, array1);

    if (index >= 0)
       System.out.println ("array1[" + index + "] == " + i);
    else
       System.out.println ("The number " + i + " does not occur in array1");
}
```

Uppgift 5 (6 poäng)

I kursboken (Lewis-Loftus) finns en applet som tillhandahåller en Doodlegraphics (s 367), där man med hjälp av musen kan rita lite på en rityta. Det finns en knapp i appleten som "suddar" hela ritytan.



Din uppgift är att komplettera denna applet med ytterligare några knappar, med vilka man kan byta färg på "ritstiftet".

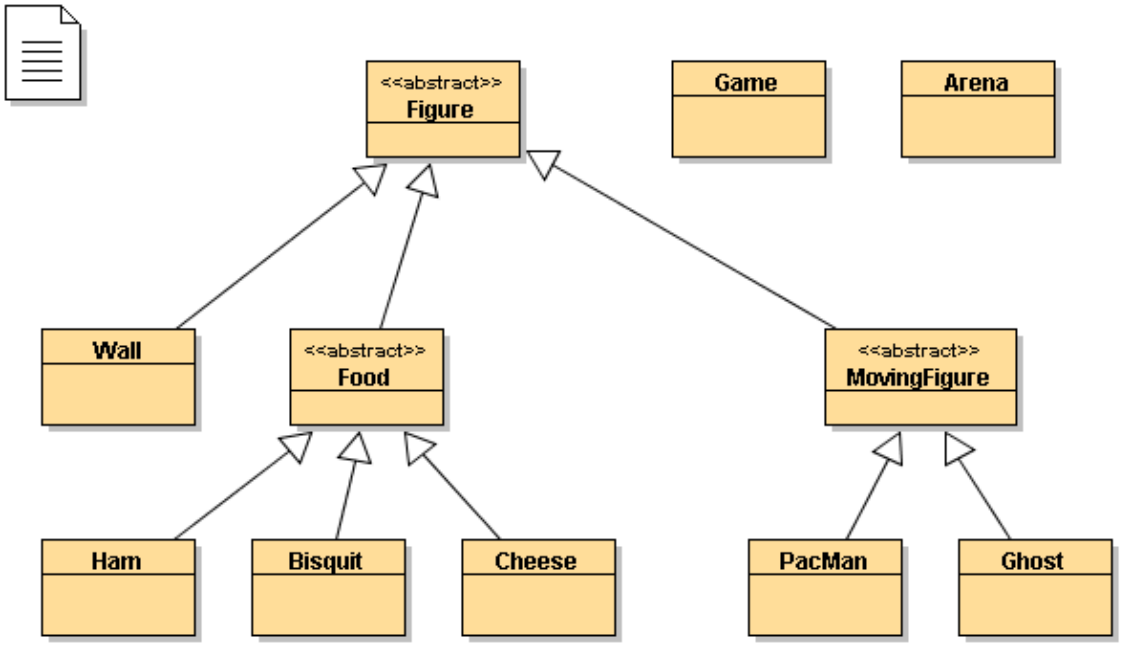


Koden till den ursprungliga appleten hittar du på kurshemsidan http://www.cs.umu.se/kurser/TDBA54/SU01/kod0630. Modifiera *denna* kod (observera att den är lite omgjord jämfört med boken).

Samma kod finns också som bilaga (s 8-10).

Uppgift 6 (8 poäng)

I denna uppgift skall du utöka spelet PacMan med två nya klasser. Koden till PacMan finns på webadressen http://www.cs.umu.se/kurser/TDBA54/SU01/kod0630. Klassdiagrammet ser ut så här:



Game är en klass med ett huvudprogram som låter PacMan gå runt 100 steg, och äta det den kommer över. En Arena innehåller ett antal Figure, varav minst en PacMan.

Du skall lägga till två klasser:

En ny sorts mat som heter **FortuneCookie**, och precis som en riktig lyckokaka så skall den innehålla en överraskning för PacMan när han äter den: värdet av en FortuneCookie skall vara slumpmässigt mellan -20 och +20, och skall vara olika för varje kaka. Hur en FortuneCookie skall se ut får du bestämma själv. Om du vill visa poängantalet eller låta det vara en riktig hemlighet är också valfritt.

En ny sorts PacMan, en **'SuperMan'** som blir hungrig: för varje steg den går skall dess poängantal minska med ett bestämt värde, förutom att det ju kan både öka och minska när han äter något. Om värdet sjunker under noll skall SuperMan försvinna ☺. Hur SuperMan skall se ut får du också bestämma själv.

Källkod uppgift 4

```
//********************************************************************
//  Doodle.java  Author Lewis & Loftus
//  ( slightly modified )
//
//  Demonstrates the use of GUI components
//********************************************************************

import java.awt.*;
import java.awt.event.*;
import java.applet.Applet;

public class Doodle extends Applet
{
   private int APPLETWIDTH = 250;
   private int APPLETHEIGTH = 300;

   private Label titleLabel;
   private DoodleCanvas canvas;

   private Button clearButton;
   private ClearListener clearListener;

   private Color appletColor = Color.white;

   //*****************************************************************
   //  Creates the GUI components and adds them to the applet
   //  A listener is added to the button
   //*****************************************************************
   public void init()
   {
      titleLabel = new Label("Rita med musen");
      titleLabel.setBackground(appletColor); // samma som appleten
      add(titleLabel);

      canvas = new DoodleCanvas();
      add(canvas);

      clearButton = new Button("Rensa");
      clearButton.setBackground(canvas.getBackground());
      clearListener = new ClearListener(canvas);
      clearButton.addActionListener(clearListener);
      add(clearButton);

      setBackground(appletColor);

      setSize(APPLETWIDTH,APPLETHEIGTH);
   }// init

}// Doodle
```

```
//********************************************************************
//   DoodleCanvas.java  Author Lewis & Loftus
//
//  Represents a darwing surface for creating simple doodles
//********************************************************************

import java.awt.*;
import java.awt.event.*;

public class DoodleCanvas extends Canvas implements MouseListener,
MouseMotionListener
{
  private final int CANVASWIDTH = 200;
  private final int CANVASHEIGTH = 200;

  private int lastX, lastY;

  //=============================================================
  //  Constructor : creates an initially empty canvas
  //=============================================================
  public  DoodleCanvas()
  {
    addMouseListener(this);
    addMouseMotionListener(this);

    setBackground(Color.white);
    setSize(CANVASWIDTH, CANVASHEIGTH);
  }// constructor

  //=============================================================
  //  Determines the initial point for a new doodle line
  //=============================================================
  public  void mousePressed(MouseEvent event)
  {
    Point first = event.getPoint();
    lastX = first.x;
    lastY = first.y;
  }// mousePressed

  //=============================================================
  //  Draws a line from the last point to the current point
  //=============================================================
  public  void mouseDragged(MouseEvent event)
  {
    Point current = event.getPoint();
    Graphics page = getGraphics();

    page.drawLine(lastX,lastY,current.x,current.y);
    lastX = current.x;
    lastY = current.y;
  }// mouseDragged

  //=============================================================
  //  Clears the canvas
  //=============================================================
  public  void clear()
  {
    Graphics page = getGraphics();
    page.drawRect(0,0,CANVASWIDTH, CANVASHEIGTH);
    repaint();
  }// mouseDragged

  //=============================================================
  //  Empty definitions for unused event methods
  //=============================================================
  public  void mouseReleased(MouseEvent event) {}// mouseReleased
  public  void mouseClicked(MouseEvent event) {}// mouseClicked
  public  void mouseEntered(MouseEvent event) {}// mouseEntered
  public  void mouseExited(MouseEvent event) {}// mouseExited
  public  void mouseMoved(MouseEvent event) {}// mouseMoved
}// DoodleCanvas
```

```
//*************************************************************************
//   ClearListener.java
//
//  Listener for clear button
//*************************************************************************

import java.awt.*;
import java.awt.event.*;

public class ClearListener implements ActionListener
{
  private DoodleCanvas doodleCanvas;

   //=========================================================
   //  Constructor
   //=========================================================
   public  ClearListener(DoodleCanvas dc)
   {
     doodleCanvas = dc;
   }// constructor

   //=========================================================
   //  Clears the canvas when the clear button is pushed
   //=========================================================
   public void actionPerformed(ActionEvent action)
   {
      doodleCanvas.clear();
   }  // method actionPerformed


}//ClearListener
```