CHALMERS TEKNISKA HÖGSKOLA
Datavetenskap TDA546

Tentamen i Objektorienterad programmering E

Fredagen 20 augusti 2010, 8.30 – 12.30.
Jourhavande lärare: Björn von Sydow, tel 1040.

Inga hjälpmedel.

Lösningar till uppgifterna behöver ej kommenteras. Ej heller behöver nödvändiga importer anges. Triviala syntaxfel tolereras utan poängavdrag vid rättningen. Skriv läsligt!

Skrivningen kan ge maximalt 40 p; 20 p ger med säkerhet godkänt. 27 p ger betyget 4 och 33 p betyget 5.

1. Betrakta följande Java-program:

```
public class Uppgift1 {

  private static void p(int n, int [] v) {
    n = 2;
    v[0] = 8;
  }

  public static void main(String[] args) {
    int [] a = {3,7,11};
    int x = 7;
    p(x,a);
    System.out.println("x="+x);
    System.out.println("a[0]="+a[0]);
  }
}
```

   Vilken utskrift fås när programmet körs? Förklara kortfattat, gärna med hjälp av en figur över variabler och minnesceller. (5 p)

2. Du ska definiera en klass `Uppgift2` som består av en statisk subrutin och en `main`-rutin som testar subrutinen:

   (a) Definiera en subrutin med signaturen

   ```
   public static void reverse(int[] a)
   ```

   som åstadkommer effekten att vända på ordningen av talen i argumentfältet `a` så att första talet kommer sist, det andra näst sist osv. (4 p)

   (b) Definiera en `main`-rutin som läser ett antal heltal från kommandoraden och samlar dem i ett fält, därefter anropar `reverse` och slutligen skriver ut resultatet. Ett exempel på körning kan se ut på följande sätt:

```
> java Uppgift2 3 5 7 9 11
11 9 7 5 3
>
```

(4 p)

3. Här ska du definiera en klass `Uppgift3` som används på precis samma sätt som programmet i föregående uppgift, men som är uppbyggt på annat sätt.

   (a) Definiera en statisk funktion med signaturen

   ```
   public static int[] reverse1(int[] a)
   ```

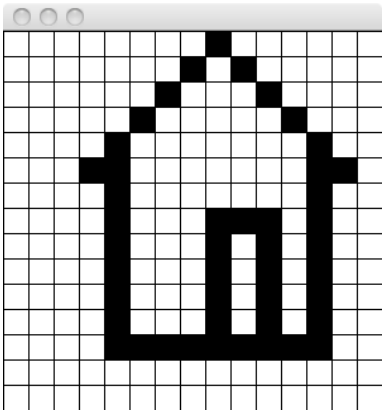
   som, utan att ändra på argumentet `a`, beräknar och returnerar ett fält där elementen kommer i omvänd ordning. (4 p)

   (b) Skriv en `main`-rutin som använder funktionen `reverse1`, och som används som program på precis samma sätt som föregående program:

   ```
   > java Uppgift3 3 5 7 9 11
   11 9 7 5 3
   >
   ```

   (3 p)

4. I denna uppgift ska vi betrakta ett program för att konstruera en *bitmap*, dvs en rektangulär bild uppbyggd av små svarta eller vita kvadrater (pixels). Ett exempel på hur ett användargränssnitt kan se ut är



   Användaren interagerar med programmet genom att klicka med musen i någon av de små kvadraterna; denna byter då färg från svart till vit eller tvärtom.

   Ett program som detta är användbart för att tillverka små ikonbilder som kan användas i knappar eller menyer. Ett användbart program skulle förutom ovanstående funktion ha en möjlighet att spara en bild på en fil i något lämpligt format, men vi bortser här från detta.

   Detta program kan byggas upp av tre klasser:

- En modellklass `BitmapModel` vars objekt som innehåller representationen av en bitmap som en matris av booleska värden (vitt eller svart). I konstrueraren anges bitmappens storlek, dvs antalet rader och kolumner. Klassen har i övrigt metoderna
  `int getWidth()` som ger antalet kolumner i matrisen.
  `int getHeight()` som ger antalet rader.
  `boolean getPixel(int x, int y)` som ger värdet av pixeln i rad x, kolumn y.
  `void flip(int x, int y)` som byter färg på pixeln i rad x, kolumn y.
- En vyklass som ger gränssnittet; denna klass är

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

public class BitmapView extends JPanel {
  private static final int SIZE = 20;
  private final int width, height;

  private BitmapModel model;

  public BitmapView(BitmapModel model) {
    this.model = model;
    width = model.getWidth();
    height = model.getHeight();
    setPreferredSize(new Dimension(width*SIZE,height*SIZE));
    setBackground(Color.WHITE);
    setOpaque(true);
    addMouseListener(new MouseHandler());
  }

  public void paintComponent(Graphics g) {
    super.paintComponent(g);
    for(int i=0;i<width;i++)
      g.drawLine(i*SIZE,0,i*SIZE,height*SIZE);
    for(int j=0;j<height;j++)
      g.drawLine(0,j*SIZE,width*SIZE,j*SIZE);
    for(int i=0;i<width;i++)
       for(int j=0;j<height;j++)
         if(model.getPixel(i,j))
           g.fillRect(i*SIZE,j*SIZE,SIZE,SIZE);
  }


  private class MouseHandler extends MouseAdapter {
    public void mouseClicked(MouseEvent e) {
    // Ännu ej implementerad!
    }
  }
}
```

- En huvudklass som innehåller en lämplig `main`-rutin.

Din uppgift är att

(a) implementera modellklassen. (4 p)

(b) implementera huvudklassen (3 p).

(c) komplettera `mouseClicked` ovan (3 p).

5. Ett enkelt interface för mängder av naturliga tal är

```
public interface IntSet {
  public void insert(int n);
  public boolean isMember(int n);
}
```

Detta kan implementeras på många olika sätt.

(a) Du ska definiera en klass `SmallIntSet` som implementerar interfacet genom att använda ett booleskt fält (eng. array) för att representera mängden; talet n ingår i mängden om motsvarande fältelement är `true`. Klassens konstruerare ska som argument ta ett positivt heltal k, vilket anger det största tal som kan sättas in. Denna implementation är därför bara lämplig om man på förhand kan ange en övre gräns för de heltal som kan komma att ingå i mängden. (5 p)

(b) Vi utökar interfacet med ytterligare en metod:

```
public java.util.Iterator iterator();
```

som returnerar en iterator för att gå igenom mängdens element. Utöka implementationen så att den nya metoden också implementeras. (5 p)