Objektorienterad Programmering
Tentamen

2003-03-14
kl. 09.00-13.00

- Tillåtet hjälpmedel är kursboken *Object Oriented Software Development using Java* av X. Jia (första eller andra upplaga).
- Tentamen ger maximalt 20 poäng. För att få betyg 3 behövs 8 poäng, för betyg 4 12 poäng och för betyg 5 16 poäng.
- Ansvarig för tentamen är Verónica Gaspes, telefon 167 380.

- Läs noggrant!
- Skriv Tydligt!

Lycka Till!

1. I kursboken utvecklas ett program för att rita funktionsgrafer. Det finns en återanvändbar klass `MultiPlotter` som är lätt att använda för att få en fungerande plotter. Här ser Du ett exempel där vi ärver från `MultiPlotter` för att rita grafer för funktionerna sinus och cosinus:

```
public class SinCos extends MultiPlotter{
    protected void initMultiPlotter(){
        Function sin = new Sin();
        Function cos = new Cos();
        addFunction(sin,java.awt.Color.red);
        addFunction(cos,java.awt.Color.blue);
    }
}
```

Nyckeln i utveckling av detta program är interfacet

```
public interface Function{
    public double apply(double x);
}
```

som implementeras bland annat av klasserna `Sin` och `Cos`.

I denna uppgift skall Du göra det möjligt att även rita grafer för *polynom*. Exempel av polynom är $4x^2 + 3$ och $10x^3 + 2x + 1$. Man kan beräkna deras värde för givna värde för x. Till exempel, kan man beräkna $4x^2 + 3$ för 2 som resulterar i 19. Detta betyder att vi kan se polynom som funktioner! Du skall använda Dig av klassen `Poly` som implementerar polynom i Java:

```
class Poly{
    // konstruerare. Skapar polynomet 0:
    public Poly()

    // konstruerare. Skapar polynomet coeff * x ^exp:
    public Poly(int exp, double coeff)

    // sumerar två polynom:
    public static Poly add(Poly p, Poly q)

    // deriverar en polynom:
    public static Poly derivate(Poly p)

    // beräknar värdet av polynomet för argumentet:
    public double eval(double x)
```

```
    public String toString()
}
```

(a) *3 p.* Definiera en klass som implementerar interfacet `Function` för att kunna använda polynom som funktioner i denna sammanhang. Du får inte göra ändringr i klassen `Poly`!

(b) *2 p.* Använd `MultiPlotter` och din klass för att rita grafen av polynomet $2x^3$ och dess derivata.

2. Interfacet `java.util.Iterator` är

```
public interface Iterator{
   public boolean hasNext();
   public Object next();
   public void remove();
}
```

Genom att tillhandahålla en iterator kan många olika typer av samlingar av objekt ge ett likformigt sätt att gå igenom alla element.

I många fall kan det vara av intresse att *bara* gå igenom de element som uppfyller en egenskap (till exempel, givet en samling av heltal, gå igenom alla jämna tal i samlingen).

(a) *3 p.* Definiera ett generellt klass för att göra detta enligt följande: klassen `FilterIterator` gör det möjligt att gå igenom alla element som genereras av en annan iterator och som dessutom uppfyller en villkor (som till exempel att vara jämn). Klassen skall alltså implementera `Iterator` och den skall innehålla en `Iterator` som genererar de element som skall filtreras i din klass. Bara de element som uppfiller villkoren skall produceras av din `FilterIterator`. Konstrueraren har argument `Property prop, Iterator iter`.

För denna uppgift har Du även tillgång till interfacet

```
public interface Property{
   public boolean test(Object x)
}
```

Klasser som implementerar `Property` definierar en egenskap hos vissa objekt. Till exempel

```
class Even implements Property{

    public boolean test(Object x){
      if(x instanceof Integer){
          int xv = ((Integer)x).intValue();
          return xv % 2 == 0;
      }
      else throw new ClassCastException();
    }
}
```

där metoden `test` returnerar true för jämna tal och flase för udda tal.

(b) *3 p.* Skriv även en liten applikation som visar hur man kan använda din klass för att gå genom alla jämna tal i en mängd av heltal (en `Set` av `Integer`)
Obs! Du behöver inte ha löst första delen för att lösa denna!

3. Definitioner för `AlgorithmAnimator`, `SortAlgorithm` och `SortDisplay` hittar Du som bilaga. Dessa klasser och interface bygger ett litet ramverk för att skriva program som animerar sorteringsalgoritmer.

   (a) *3 p.* Vilka design mönster har använts. Ange var och med vilket syfte.

   (b) *3 p.* Ändra `AlgorithmAnimator` så att ett fält av heltal kan sorteras enligt många algoritmer. Se till att klassen har ett fält med sorterings algoritmer, att sorterings algoritmer kan läggas till detta fältet och att alla algoritmer i fältet används efter varandra för att sortera samma fält av heltal.

   (c) *3 p.* Ändra `AlgorithmAnimator` så att det hela tiden sorteras nya fält av heltal med samma algoritm.

```
import java.awt.*;

public interface SortDisplay {
  int getArraySize(Dimension d);
  void display(int a[], Graphics g, Dimension d);
}
```

```
public abstract class SortAlgorithm extends java.util.Observable {

    abstract void sort(int a[]);

    protected void pause(){
        setChanged();
        notifyObservers();
    }

    protected static void swap(int a[], int i, int j) {
        int T;
        T = a[i]; a[i] = a[j]; a[j] = T;
    }
}
```

```
import java.util.*;
import java.awt.*;
public abstract class AlgorithmAnimator
                extends java.applet.Applet
                implements Runnable,Observer{

    protected int delay;
    protected Thread animationThread;
    private   boolean finished;
    protected SortAlgorithm theAlgorithm;
    protected int [] arr;
    protected SortDisplay theDisplay;

    public void start(){
        if(animationThread == null){
            animationThread = new Thread(this);
            finished = false;
            animationThread.start();
        }
    }

    public void stop(){
        finished = true;
    }

    public void run(){
        theAlgorithm.sort(arr);
    }


    final public  void update(Observable obs, Object o) {
        if (!finished) {
            try {
                Thread.currentThread().sleep(delay);
            } catch (InterruptedException e) {}
            repaint();
        }
    }

    public void setDelay(int d){
        delay = d;
    }
```

```
    public final void init() {
        setDelay(Integer.parseInt(getParameter("delay")));

        initAnimator();
        scramble();
        theAlgorithm.addObserver(this);
    }

    protected abstract void initAnimator();

    protected void scramble() {
        int size = theDisplay.getArraySize(getSize());
        arr = new int[size];
        for (int i = arr.length; --i >= 0;) {
            arr[i] = i;
        }
        for (int i = arr.length; --i >= 0;) {
            int j = (int)(i * Math.random());
            SortAlgorithm.swap(arr, i, j);
        }
    }

    public void paint(Graphics g) {
        theDisplay.display(arr,g,getSize());
    }

}
```