Data och Informationsteknik / Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
Magnus Myréen

Modellsvar för
Tentamen för Objektorienterad programvaruutveckling, TDA545

Måndag, 2016-01-04, 08:30-12:30, byggnad H

Ansvarig lärare:	Magnus Myreen; Uno Holmer besöker tentan två gånger
Vitsordsgränser:	3=28p, 4=38p, 5=48p, max 60p.
Hjälpmedel:	på sista sidan av dokument finns ett utdrag ur Javas API
Resultat:	skickas per epost från Ladok, *kommer sent*, men inte senare än 2016-02-05
Lösningar:	kommer att finnas på kurshemsidan
Granskning av rättning:	tider för detta kommer att finnas på kurshemsidan

---

Kom ihåg att inte fastna på en uppgift. Bestäm i förväg din egen tidsgräns per uppgift. ***Lycka till!***

Uppgift 1: [5 poäng totalt] *arv, referensvärden*

a) Förklara *kort* vad arv innebär i Java. [2 poäng]

Arv tillåter en klass att låna kod från en annan klass. Man ärver ofta klasser från Javas API (och andra bibliotek) för att skapa nytt beteende genom att ändra och utvidga det fördefinierade beteendet av bibliotecksklasserna.

b) Vad skrivs ut när följande `Test` program körs? [3 poäng]

```
class T1 {

    private int a = 0;

    public int getA() { return a; }
    public void setA(int a) { this.a = a; }
    public void incA() { a=a+1; }

}

class T2 extends T1 {

    private int b = 0;

    public int getB() { return b; }
    public void incAB() { b=b+1; incA(); }

}

public class Test {

    public static void main(String[] args) {
        T2 t2 = new T2();
        T1 t1 = t2;
        System.out.println(t1.getA() + " " + t2.getA() + " " + t2.getB());
        t1.incA();
        System.out.println(t1.getA() + " " + t2.getA() + " " + t2.getB());
        t1 = new T1();
        t2.incAB();
        System.out.println(t1.getA() + " " + t2.getA() + " " + t2.getB());
    }

}
```

Tips: Var noggrann! Bra att rita "minnesplatser" och "pilar" som vi gjorde på föreläsningarna.

Programmet skriver ut:

```
0 0 0
1 1 0
0 2 1
```

Uppgift 2: [10 poäng totalt] *array, loopar, exceptions*

Denna uppgift ber dig skriva kod som räknar röster.

a) Implementera en metod `mostInARow(a)` som returnerar siffran (dvs en `int`) som upprepar sig de flesta gånger utan avbrott i arrayn `a`. Följande kod bör skriva ut `2` för att det finns fyra tvåor i rad. Inget annat nummer har flera än fyra i rad. [6 poäng]

```
int[] a = { 8, 8, 2, 2, 2, 2, 8, 8, 1, 2, 8, 8 };
System.out.println(mostInARow(a));
```

```
private int mostInARow(int[] a) {
    // antar att array a inte är tom och inte null
    int curr_val = a[0];
    int curr_count = 1;
    int most_val = curr_val;
    int most_count = curr_count;
    for (int i=1; i<a.length; i++) {
        if (curr_val == a[i]) {
            curr_count++;
            if (curr_count > most_count) {
                most_count = curr_count;
                most_val = curr_val;
            }
        } else {
            curr_val = a[i];
            curr_count = 1;
        }
    }
    return most_val;
}
```

b) Anta att det finns en sorteringsmetod `Quicksort.sort(a)`. Använd `mostInARow` och `Quicksort.sort` för att implementera en metod som, givet en array av röster, räknar vilket alternativ har fått mest röster. Metoden bör ha följande form. För exempel arrayn ovan bör din implementation av `winnerOfElection` returnera 8. Metoden bör kasta en beskrivande exception ifall det av någon anledning inte finns röster att räkna. [4 poäng]

```
public int winnerOfElection(int[] a) { ... }
```

Obs: Du kan anta att det *inte* finns lika många röster på två olika alternativ.

```
public int winnerOfElection(int[] a) {
    if (a == null) {
        throw new RuntimeException("Null array passed to winnerOfElection");
    } else if (a.length < 1) {
        throw new RuntimeException("Empty array passed to winnerOfElection");
    }
    Quicksort.sort(a);
    return mostInARow(a);
}
```

Uppgift 3: [18 poäng totalt] *att använda klass, rekursion*

Anta att du har en klass för rationella tal `RatNum` som i `RatNum`-labben, men som använder `BigInteger` internt (dvs overflow är omöjligt).

```
public class RatNum {
    public RatNum(int n, int d) { ... }  // skapar nytt rationellt tal n/d
    public RatNum add(RatNum r) { ... }  // adderar
    public RatNum sub(RatNum r) { ... }  // subtraherar
    public RatNum mult(RatNum r) { ... } // multiplicerar
    ...
}
```

a) Implementera en metod `pi`. Metoden bör ha följande signatur:

```
private RatNum pi(int k) { ... }
```

och bör implementera följande rekursiva beräkning av ett närmevärde för π.

$$\begin{aligned} \texttt{pi}(0) &= 0 \\ \texttt{pi}(k+1) &= \texttt{pi}(k) + \frac{1}{16^k}\left(\frac{4}{8k+1} - \frac{2}{8k+4} - \frac{1}{8k+5} - \frac{1}{8k+6}\right), \quad \text{if } k \geq 0 \end{aligned}$$

Ju större värdet på `k` är desto närmare π kommer denna funktion. [10 poäng]

```
private RatNum pow(RatNum r, int power) {
    RatNum t = new RatNum(1,1);
    for (int i=0; i<power; i++) {
        t = t.mult(r);
    }
    return t;
}

private RatNum pi(int k) {
    if (k == 0) { return new RatNum(0,1); }
    k = k-1;
    RatNum t = new RatNum(4,8*k+1);
    t = t.sub(new RatNum(2,8*k+4));
    t = t.sub(new RatNum(1,8*k+5));
    t = t.sub(new RatNum(1,8*k+6));
    t = t.mult(pow(new RatNum(1,16),k));
    return t.add(pi(k));
}
```

b) Skriv ett helt program `Pi` som skriver ut ett givet antal decimaler av π. Programmet bör ge utskrifter som exemplet nedan. I exemplet nedan körs programmet först med `3` som input data och sedan med indata `50`. Anta att indata alltid finns och går att läsa in.

```
$ java Pi 3
pi = 3.141...

$ java Pi 50
pi = 3.14159265358979323846264338327950288419716939937510...
```

Anta att `pi(3)` är tillräckligt nära π för att räkna de tre första decimalerna av π, likaså `pi(50)` är tillräckligt nära π för att räkna de första 50 decimalerna, osv.

Anta att `RatNum` klassen har en instansmetod `floorIntegerAsString()` som returnerar en sträng av heltalsdelen av det rationella talet. Exempel: (new RatNum(8,3)).`floorIntegerAsString()` returnerar `"2"`.

Tips: pseudokod ((2/3) * 1000).`floorIntegerAsString()` blir `"666"`. [8 poäng]

```
public class Pi {

    private RatNum pow(RatNum r, int power)  // som ovan ...

    private RatNum pi(int k)  // som ovan ...

    public static void main(String[] args) {
        int decimals = Integer.valueOf(args[0]);
        if (decimals < 0) { decimals = 0; }
        RatNum p = (new Pi()).pi(decimals);
        RatNum ten = new RatNum(10,1);
        p = p.mult(pow(ten,decimals));
        String str = p.floorIntegerAsString();
        str = "3." + str.substring(1) + "...";
        System.out.println("pi = " + str);
    }

}
```

Uppgift 4: [15 poäng totalt] *swing, händelsehantering, att skriva klass*

Din uppgift är att skriva kod för en klass som kan användas som en del av ett ritprogram. Klassen som du skriver bör:

- Ha en konstruktor som tar en array av `Color`-objekt som parameter. [2 poäng]
- Konstruktorn bör få ett fönster att synas. [2 poäng]
- Fönstret bör ha en färgad knapp för varje färg som kom i `Color` arrayn. [3 poäng]
- När man trycker på en knapp då bör knappens text bli `SELECTED`. Det får endast finnas en knapp med texten `SELECTED`. Övriga knappar ska inte ha text. [3 poäng]
- Klassen ska ha en `getSelectedColor()`-metod som returnerar färgen av knappen där det står `SELECTED`. [3 poäng]
- När fönstret skapas ska det se ut som den första knappen blivit tryckt. [2 poäng]

```
public class ColorFrame extends JFrame {

    int selected = -1;
    Color[] colors;
    JButton[] buttons;

    public ColorFrame(Color[] colors) {
        this.colors = colors;
        buttons = new JButton[colors.length];
        JPanel p = new JPanel();
        add(p);
        for (int i=0;i<colors.length; i++) {
            JButton b = new JButton("");
            b.setBackground(colors[i]);
            final int j = i;
            b.addActionListener(new ActionListener() {
                    public void actionPerformed(ActionEvent e) {
                        selectColor(j);
                    }
                });
            p.add(b);
            buttons[i] = b;
        }
        selectColor(0);
        pack();
        setVisible(true);
    }

    public Color getSelectedColor() {
        return colors[selected];
    }
```

```
    public void selectColor(int j) {
        if (selected != -1) {
            buttons[selected].setText("");
        }
        selected = j;
        buttons[selected].setText("SELECTED");
    }

}
```

Uppgift 5: [12 poäng totalt] *grafik, att skriva klass*

För denna uppgift lönar det sig att använda sig av följande klass som vi sett i föreläsningarna.

```
class Rotate2D {

    // Koden nedan roterar en punkt (x,y) runt nollstället (0,0)
    // med en given vinkel (angle). Den nya punktens koordinater
    // är (getX(x,y,angle), getY(x,y,angle)).

    public static double getX(double x, double y, double angle) {
        return x * Math.cos(angle) - y * Math.sin(angle);
    }

    public static double getY(double x, double y, double angle) {
        return x * Math.sin(angle) + y * Math.cos(angle);
    }

}
```

a) Din uppgift är att *skriva ett program* som får ett fönster att synas. En *spiral ska ritas i fönstret* som i bild (o) nedan. Spiralen ska ha sin mittpunkt i mitten av fönstret. Spiralen skall fylla hela fönstret. [9 poäng]

Obs. Det finns flera sätt att lösa denna uppgift. Här är en lösning:

```
public class Spiral extends JPanel {

    private int intX(double x) { return (int)x + getWidth()/2; }

    private int intY(double y) { return (int)y + getHeight()/2; }

    private void drawThickLine(double x1, double y1, double x2, double y2, Graphics g) {
        double s = 1.6; // thickness must be > 1.0
        int[] xPoints = {intX(x1),intX(s*x1),intX(s*x2),intX(x2)};
        int[] yPoints = {intY(y1),intY(s*y1),intY(s*y2),intY(y2)};
        g.fillPolygon(xPoints,yPoints,4);
    }

    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        double x = 0.5;
        double y = 0.0;
        double prev_x = x;
        double prev_y = y;
        double w = (double)(getWidth()) / 2.0;
        double h = (double)(getHeight()) / 2.0;
        while (x*x + y*y < w*w + h*h) {
            x = Rotate2D.getX(prev_x*1.01, prev_y*1.01, 0.05);
            y = Rotate2D.getY(prev_x*1.01, prev_y*1.01, 0.05);
            drawThickLine(prev_x,prev_y,x,y,g);
            prev_x = x;
            prev_y = y;
        }
    }
```

```
    public static void main(String[] args) {
        JFrame f = new JFrame();
        f.add(new Spiral());
        f.setSize(400,400);
        f.setVisible(true);
    }

}
```

b) *Förklara* hur din kod ser till att spiralen ritas rätt och fyller hela fönstret. [3 poäng]

Koden ritar en spiral från insidan ut åt. Den börjar med en punkt (x, y) som är nära noll. För varje iteration av loopen räknar den en ny punkt som är 1 % längre från noll och roterad med 0.05. Koden håller koll på vad den föregående punkten var och ritar ett sträck från den förra punkten till den nya.

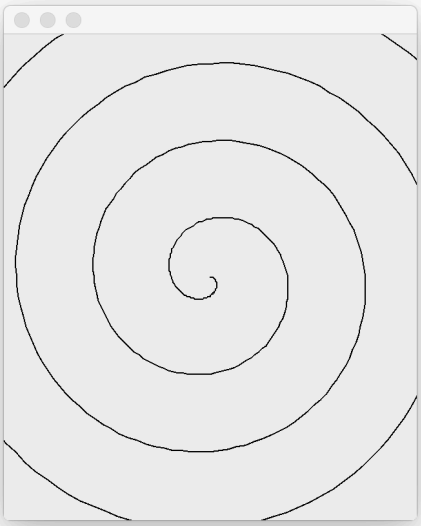
Denna loop körs tills punkten (x, y) ligger förlångt från noll för att synas i fönstret. Koden använder sig av pytagoras sats för att kolla hur långt en punkt är från noll. Om x*x + y*y < w*w + h*h är sant, då är punkten (x, y) närmare noll än punkten (w,h).

Koden får spiralen att se tjock ut genom att rita en polygon istället för ett enkelt sträck. Polygonen har två punkter enligt sträcket och två punkter på ett sträck som 60 % längre från noll. Innan en koordinat används för ritning justeras den så att noll är i mitten av fönstret.

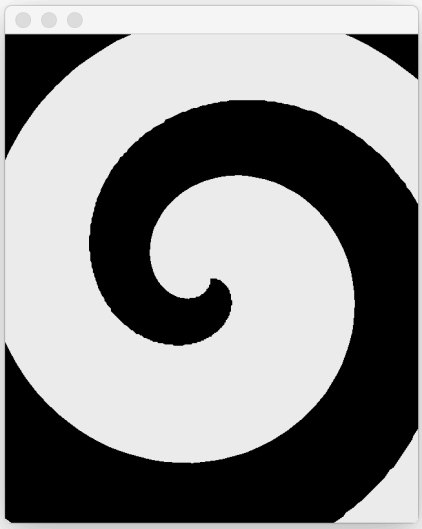
Tips: Det är kanske lättast att först fundera hur man ritar en enkelare spiral som i bild (x).

Tips: för (o), se `fillPolygon` i `Graphics` klassen på nästa sida.

(x)



(o)



Utdrag ur Javas API. ***Obs.*** Man behöver inte använda alla dessa klasser. Man får också använda annat från Javas API.

Class AbstractButton extends JComponent extends Container extends Component extends Object

`public void addActionListener(ActionListener l)`
Adds an ActionListener to the button.

`public void setText(String text)`
Sets the button's text.

Interface ActionListener

`void actionPerformed(ActionEvent e)`
Invoked when an action occurs.

Class ActionEvent extends AWTEvent extends EventObject extends Object

`String getActionCommand()`
Returns the command string associated with this action.

Class Color extends Object

`static Color BLACK`
The color black.

Class Component extends Object

`int getHeight()`
Returns the current height of this component.

`int getWidth()`
Returns the current width of this component.

`void repaint()`
Repaints this component.

Class Container extends Component extends Object

`Component add(Component comp)`
Appends the specified component to the end of this container.

Class Graphics extends Object

`void drawLine(int x1, int y1, int x2, int y2)`
Draws a line, using the current color, between the points (x1, y1) and (x2, y2) in this graphics context's coordinate system.

`abstract void fillPolygon(int[] xPoints, int[] yPoints, int nPoints)`
Fills a closed polygon defined by arrays of x and y coordinates.

`abstract void fillOval(int x, int y, int width, int height)`
Fills the specified oval.

`abstract void setColor(Color c)`
Sets this graphics context's current color to the specified color.

Class Integer extends Number extends Object

`static int parseInt(String s)`
Parses the string argument as a signed decimal integer.

Class JButton extends AbstractButton extends JComponent extends Container extends Component …

`JButton(String text)`
Creates a button with text.

Class JComponent extends Container extends Component extends Object

`protected void paintComponent(Graphics g)`
Calls the UI delegate's paint method, if the UI delegate is non-null.

`public void setBackground(Color bg)`
Sets the background color of this component.

Class JFrame extends Frame extends Window extends Container extends Component extends Object

`Container getContentPane()`
Returns the contentPane object for this frame.

Class JPanel extends JComponent extends Container extends Component extends Object

`JPanel()`
Creates a new JPanel with a double buffer and a flow layout.

Class String extends Object

`int length()`
Returns the length of this string.

`static String valueOf(int i)`
Returns the string representation of the int argument.

Class Timer extends Object

`Timer(int delay, ActionListener listener)`
Creates a Timer and initializes both the initial delay and between-event delay to delay milliseconds.

`void start()`
Starts the Timer, causing it to start sending action events to its listeners.

Class Window extends Container extends Component extends Object

`public void setLocation(int x, int y)`
Moves this component to a new location. The top-left corner of the new location is specified by the x and y parameters in the coordinate space of this component's parent.

`void setVisible(boolean b)`
Shows or hides this Window depending on the value of parameter b.