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

Modellsvar för
Tentamen för Objektorienterad programvaruutveckling, TDA545

Torsdag, 2015-10-29, 14:00-18:00, byggnad H

Ansvarig lärare:	Magnus Myréen, besöker tentan två gånger
Vitsords gränser:	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
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: [8 poäng totalt] *referensvärden, arrays, loopar*

a) Skriv ett *kort* kodexempel som visar att Javas arrayer är referensvärden. [2 poäng]

Ett exempel:

```
int[] a = {0,0,0};
int[] b = a; // referensvärdet kopieras
b[1] = 5;
System.out.println(a[1]); // skriver ut 5, dvs updatering av b påverkade a
```

b) Vad skrivs ut när följande program körs? [6 poäng]

```
public class Hmm {

    public static void swap(int[] a, int i, int j) {
        int a_i = a[i];
        int a_j = a[j];
        a[i] = a_j;
        a[j] = a_i;
    }

    public static void main(String[] args) {

        int[] a = { 0, 1, 2, 3 };
        for (int i=0; i<a.length; i++) {
            swap(a,i,a.length-1-i);
        }
        System.out.println("a: " + a[0] + "," + a[1] + "," + a[2] + "," + a[3]);

        int[] b = { 0, 1, 2, 3 };
        a = b;
        for (int i=0; i<b.length; i++) {
            if (i % 2 == 0) { a[i] = 0; } else { b[i] = 0; }
        }
        System.out.println("b: " + b[0] + "," + b[1] + "," + b[2] + "," + b[3]);

        int k = 20;
        while (0 < k) {
```

```
            k = k / 2;
            System.out.println("k = " + k);
        }

    }

}
```

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

Programmet skriver:

```
a: 0,1,2,3
b: 0,0,0,0
k = 10
k = 5
k = 2
k = 1
k = 0
```

Uppgift 2: [10 poäng totalt] *arrays, loopar/rekursion*

Implementera en metod `pascal` som returnerar en rad av Pascals triangel (se nedan). Metoden tar in radnumret som parameter och bör returnera en array som består av talen för den raden. Metoden bör ha följande form:

```
public static int[] pascal(int n) { ... }
```

Obs: Bry dig inte om att skriva snabb kod. Se till att metoden returnerar korrekt resultat.

De första åtta raderna av Pascals triangel är givna nedan. Observera att varje rad beror endast på talen i raden ovan. Varje internttal är summan av de närmaste talen ovanför.

```
rad 0:                 1
rad 1:               1  1
rad 2:             1  2  1
rad 3:           1  3  3  1
rad 4:         1  4  6  4  1
rad 5:       1  5  10 10 5  1
rad 6:     1  6  15 20 15 6  1
rad 7:   1  7  21 35 35 21 7  1
```

En lösning med rekursion:

```
public static int[] pascal(int n) {
        if (n == 0) {
            int[] a = new int[1];
            a[0] = 1;
            return a;
        } else {
            int[] a = pascal(n-1);
            int[] b = new int[a.length+1];
            b[0] = 1;
            b[b.length-1] = 1;
            for (int i=1;i<a.length; i++) {
                b[i] = a[i-1]+a[i];
            }
            return b;
        }
    }
```

Samma lösning med loopar:

```
public static int[] pascal(int n) {
    int[] a = new int[1];
    a[0] = 1;
    for (int j=0;j<n;j++) {
        int[] b = new int[a.length+1];
        b[0] = 1;
        b[b.length-1] = 1;
        for (int i=1;i<a.length; i++) {
            b[i] = a[i-1]+a[i];
        }
        a = b;
    }
    return a;
}
```

Det går också att skriva en iterativ lösning som endast skapar en array, men då måste man vara mycket försiktig att man inte blandar mellan nya och gamla värden i arrayn. I följande kod går den inre loopen uppifrån neråt så att vi inte blandar nya och gamla värden.

```
public static int[] pascal(int n) {
    int[] a = new int[n+1];
    a[0] = 1;
    for (int j=0;j<n;j++) {
        a[j+1] = 1;
        for (int i=j;0<i; i--) {
            a[i] = a[i-1] + a[i];
        }
    }
    return a;
}
```

Alla lösningar ovan ger fulla poäng. Det finns också andra sätt att komma till fulla poäng.

Uppgift 3: [12 poäng totalt] *att skriva klass, strängar*

Din uppgift är att skriva en klass som skriver (till terminalen) kvitton av följande form.

```
vetemjöl, 1 kg ____ 50 kr
bananer, 421 g ____ 35 kr
mjölk _____________ 15 kr
bröd ______________ 10 kr
.......................
Summa _____________ 110 kr
Varav 25 % MOMS ___ 27 kr
```

Din klass `Receipt` bör ha formen:

```
public class Receipt {

    ... privata instansvariabler här ...

    /** Returnerar name + "___…___" + price så att strängen är 25 tecken lång.
        Obs: ifall (name+price).length() > 23, då returnerar den name+price. */
    private String getLine(String name, String price) { ... }

    /** Lägger till en vara. Skriver ej ut något. */
    public void add(Item i) { ... }
```

```
    /** Skriver ut kvittot i stil med exemplet ovan. */
    public void print() { ... }

}
```

Varor är av typen `Item`:

```
public abstract class Item {
    public abstract String getName();
    public abstract int getPrice();
}
```

Din uppgift är att implementera klassen `getLine`, `add` och `print` så att funktionaliteten som är beskriven i kodens kommentarer förverkligas. Metoden `getLine` är avsedd som hjälpmetod för de andra metoderna. Bestäm själv vilka instansvariabler som passar din implementation bäst.

Tips: man kan undvika en `Item`-array (eller dylikt) genom att istället bygga upp delar av utskriften för `print` redan i `add`. `System.out.print(" Rad 1 \n Rad 2 ")` skriver två rader.

```
public class Receipt {

    private String list = "";
    private int total;

    private String getLine(String x, String y) {
        if ((x+y).length() > 23) { return x+y; }
        x = x + " ";
        y = " " + y;
        while ((x+y).length() < 25) {
            y = "_" + y;
        }
        return x + y;
    }

    public void addItem(Item i) {
        list = list + getLine(i.getName(), String.valueOf(i.getPrice()) + " kr") + "\n";
        total = total + i.getPrice();
    }

    public void print() {
        System.out.print(list);
        System.out.println(".........................");
        System.out.print(getLine("Summa", String.valueOf(total) + " kr") + "\n");
        System.out.print(getLine("Varav 25 % MOMS", String.valueOf(total/4) + " kr") + "\n");
    }

}
```

Obs: det blir inte poängavdrag ifall `String.valueOf` saknas.

Poäng: man får max 5 poäng för `getLine` metoden och max 7 poäng för resten av koden, dvs `add`, `print` och deklarationen av instansvariabler.

Uppgift 4: [15 poäng totalt] *swing, händelsehantering, slumptal*

Skriv kod som implementerar ett enkelt spel med följande funktionalitet:

- Spelet består av ett fönster som endast innehåller en knapp. [4 poäng]
- När man trycker på knappen bör fönstret flytta till ett slumpmässigt ställe. [4 poäng]
- När knappen tryckts tio gånger, då slutar programmet. [3 poäng]
- När programmet slutar skriver det ut hur många millisekunder det körde. [4 poäng]

Idén med spelet är alltså att spelaren ska försöka klicka på knappen så fort som möjligt. Fönstret byter plats vid varje klick.

Obs: Använd `Random` för slumptal. Placera fönstret med `JFrames` `setLocation`-metod, men akta att inte fönstret flyttas utanför skärmen!

Tips: Anta att du får skärmens bredd och vidd från `Screen.getWidth()` och `Screen.getHeight()`.

Tips: Man kan använd `System.currentTimeMillis()` för att få veta hur många millisekunder det är sen 1 januari 1970.

Tips: Man kan sluta programmet med ett anrop till `System.exit(0)`.

```
public class Game {

    public static void main(String[] args) {
        JFrame f = new JFrame();
        JButton b = new JButton("Knapp");
        f.add(b);
        long l1 = System.currentTimeMillis();
        b.addActionListener(new ActionListener() {
                int i = 0;
                Random r = new Random();
                public void actionPerformed(ActionEvent e) {
                    if (i < 10) {
                        i = i + 1;
                        int w = Screen.getWidth() - f.getWidth();
                        int h = Screen.getHeight() - f.getHeight();
                        f.setLocation(r.nextInt(w),r.nextInt(h));
                    } else {
                        long l2 = System.currentTimeMillis();
                        System.out.println("Milliseconds: " + (l2-l1));
                        System.exit(0);
                    }
                }
            });
        f.pack();
        f.setVisible(true);
    }

}
```

Ifall man vill köra koden behöver man också följande, men OBS svar som ovan ger fulla poäng.

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

class Screen {
    public static int getWidth() {
        return (int)(Toolkit.getDefaultToolkit().getScreenSize().getWidth());
    }
    public static int getHeight() {
        return (int)(Toolkit.getDefaultToolkit().getScreenSize().getHeight());
    }
}
```

Uppgift 5: [15 poäng totalt] *grafik, gränssnitt, arv*

a) Förklara hur man ritar i Java. Mera specifikt: förklara hur koordinatsystemet fungerar och hur man bör använda `Graphics`, `paintComponent` och `repaint`. [4 poäng]

Komponenter ritas av `paintComponent` metoden. För att rita egen grafik är det lättast att överskugga den metoden och på det sättet ersätta den med egen kod. [1 poäng]

Denna metod får som parameter ett objekt av typen `Graphics`. Det är via detta objekt man ritar, t.ex. med att anropa `g.fillRect(0,0,5,5)`. [1 poäng]

Koordinatsystemet skiljer sig från det vi är vana vid från matematik. Nollpunkten är i det övre vänstra hörnet och y-axeln växer neråt. [1 poäng]

Om man vill rita om en komponent ska man inte kalla på `paintComponent`, istället bör man anropa `repaint()`. [1 poäng]

b) För den här delen bör du implementera en klass som ritar kurvan av en given matematisk funktion. Kalla den nya klassen `FuncPanel` och se till att den ärver `JPanel`. Din klass `FuncPanel` skall ha en konstruktor som tar en `MathFunc m` som parameter. `FuncPanel` ska rita funktionens kurva. [11 poäng]

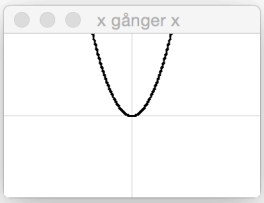
Obs: `FuncPanel` ska rita funktionen på ett sätt som är bekant från matematik: x-axeln ska växa mot höger, y-axeln (dvs funktionens värden) ska växa uppåt, och nollstället för båda axlarna ska vara i mitten av panelen.

Gränssnittet `MathFunc` är:

```
public interface MathFunc {
    public double f(double x);
}
```

Nedan finns det testkod som bör få din implementation av `FuncPanel` att se ungefär ut som bilderna till höger.

```
JFrame f1 = new JFrame(" x gånger x ");
JPanel p1 = new FuncPanel(new MathFunc() {
        public double f(double x) {
            return x*x;
        }
    });
f1.add(p1); f1.setSize(200,150); f1.setVisible(true);
```




```
JFrame f2 = new JFrame(" sinusfunktionen ");
JPanel p2 = new FuncPanel(new MathFunc() {
        public double f(double x) {
            return Math.sin(x);
        }
    });
f2.add(p2); f2.setSize(200,150); f2.setVisible(true);
```

```
public class FuncPanel extends JPanel {

    private MathFunc f;
    private final double SCALE = 15.0; // zoom-faktor

    public FuncPanel(MathFunc f) {
        this.f = f;
    }

    private int getX(double x) {
        int w = getWidth();
        // den matematiska x-koordinaten * zoom-faktorn
        // + hälften av brädden (så att nollpunkten kommer i mitten)
        return (int)(x * SCALE) + (w / 2);
    }

    private int getY(double y) {
        int h = getHeight();
        // det negativa värdet av den matematiska y-koordinaten * zoom-faktorn
        // + hälften av höjden (så att nollpunkten kommer i mitten)
        return (int)(-y * SCALE) + (h / 2);
    }

    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        // ritar den vita bakgrunden
        int w = getWidth();
        int h = getHeight();
        g.setColor(Color.WHITE);
        g.fillRect(0,0,w,h);
        // ritar en grå linje för x-axeln och y-axeln
        g.setColor(new Color(230,230,230));
        g.drawLine(0,getY(0),w,getY(0));
        g.drawLine(getX(0),0,getX(0),h);
        // ritar punkter på kurvan i båda riktingarna från x=0 punkten
        g.setColor(Color.BLACK);
        double x = 0.0;
        double eps = 1.0 / (SCALE * 100.0);
        while (getX(x) < w) {
            x = x + eps;
            // här har vi try-catch ifall beräkningen av funktionsvärdet
            // inte fungerade, t.ex. sqrt(-3.0).
            try {
                g.fillOval(getX(x)-1,getY(f.f(x))-1,2,2);
            } catch (Exception e) { }
            try {
                g.fillOval(getX(-x)-1,getY(f.f(-x))-1,2,2);
            } catch (Exception e) { }
        }
    }

}
```

Det blir poäng avdrag (max 4 poäng) ifall man glömmer att zooma. På bilderna är det tydligt att sinusfunktionen går mera än 1 pixel upp från noll-linjen.

Obs: det blir inte poängavdrag ifall `try-catch` saknas.

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.

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 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 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.

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 Random extends Object

`Random()`
Creates a new random number generator.
`int nextInt()`
Returns the next pseudorandom, uniformly distributed int value from this random number generator's sequence.
`int nextInt(int n)`
Returns a pseudorandom, uniformly distributed int value between 0 (inclusive) and the specified value (exclusive), drawn from this random number generator's sequence.

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.