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

Modellsvar för
Tentamen för Objektorienterad programvaruutveckling, TDA545

Onsdag, 2016-08-24, 08:30-12:30, byggnad M

Ansvarig lärare:	Magnus Myréen 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
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: [10 poäng totalt] *rekursion, arv, referensvärden*

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

Rekursion innebär att man anropar metoden från sig själv. Vissa sorterings algoritmer är lättast implementerade med hjälp av rekursion.

b) Vad skrivs ut när följande `Test` program körs? [8 poäng]

```
class T1 {
    public boolean odd (int n) { if (n == 0) { return false; } { return even(n-1); } }
    public boolean even (int n) { if (n == 0) { return true; } { return odd(n-1); } }
}

class T2 extends T1 {
    public boolean odd (int n) { return false; }
}

public class Test {

    public static void print_it(int i, int[] a, int[] b) {
        if (i == 0) { return; }
        i = i - 1;
        System.out.println(a[i]);
        print_it(i,b,a);
    }

    public static void main(String[] args) {
        System.out.println("Test A:");
        int[] a = {1,2,3,4};
        int[] b = {6,7,8,9};
        print_it(a.length,a,b);
        System.out.println("Test B:");
        T1 t1 = new T1();
        T2 t2 = new T2();
        System.out.println(t1.even(4));
        System.out.println(t1.odd(4));
        System.out.println(t2.even(4));
        System.out.println(t2.odd(4));
    }
```

```
}
```

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

```
Test A:
4
8
2
6
Test B:
true
false
false
false
```

Uppgift 2: [5 poäng totalt] *loopar, arrays*

Implementera en metod som summerar värdena i en två dimensionell matris av heltal. Metoden bör ha följande signatur:

```
public int sum(int[][] input);
```

Obs. Se till att metoden aldrig kastar en exception eller på annat sätt kraschar.

```
public int sum(int[][] input) {
    if (input == null) { return 0; }
    int res = 0;
    for (int i = 0; i < input.length; i++) {
        if (input[i] != null) {
            for (int j = 0; j < input[i].length; j++) {
                res = res + input[i][j];
            }
        }
    }
    return res;
}
```

Uppgift 3: [10 poäng totalt] *loopar, stringar, arrays*

Denna uppgift ber dig skriva en metod som hittar namn på personer som har födelsedagar som faller på samma dag. Metoden bör heta `printMatchingBirthdays`.

Följande testprogram illustrerar vad denna metod ska skriva ut.

```
String[] data = {
    "1963-02-05 Lille Skutt",
    "1979-06-19 Vargen",
    "1942-02-15 Skalman",
    "1997-05-12 Nalle-Maja",
    "1989-05-17 Krösus Sork",
    "1959-09-25 Kapten Buster",
    "1994-02-15 Brum",
    "1963-04-23 Bamse",
    "1992-02-15 Teddy",
    "1938-09-25 Reinard" };

printMatchingBirthdays(data);
```

Testprogrammet ovan ska skriva ut följande:

```
Skalman has the same birthday as Brum.
Skalman has the same birthday as Teddy.
Kapten Buster has the same birthday as Reinard.
```

```
Brum has the same birthday as Teddy.
```

Implementera metoden `printMatchingBirthdays`. [10 poäng]

Obs. Du kan anta att data alltid kommer i precis samma format som i exemplet ovan.

Obs. Ordningen av raderna i utskriften behöver inte vara precis som ovan.

```
public String getName(String str) {
    return str.substring(11);
}

public boolean isMatch(String line1, String line2) {
    String s1 = line1.substring(5,10);
    String s2 = line2.substring(5,10);
    return s1.equals(s2);
}

public void printMatchingBirthdays (String[] a) {
    if (a == null) { return; }
    for (int i=0; i<a.length; i++) {
        for (int j=i+1; j<a.length; j++) {
            if (isMatch(a[i],a[j])) {
                System.out.println(getName(a[i]) +
                   " has the same birthday as " +
                   getName(a[j]) + ".");
            }
        }
    }
}
```

Uppgift 4: [20 poäng totalt] *grafik, händelsehantering, att skriva klass*

Din uppgift är att skriva kod för ett mycket enkelt ritprogram. Ritprogrammet bör:

- Starta med en tom yta i ett fönster. [3 poäng]
- Varje gång man klickar i fönstret bör en liten prick ritas där det klickades. [7 poäng]
- Tidigare ritade prickar får aldrig försvinna. [10 poäng]

Obs. Se till att prickarna blir kvar fast man förstorar fönstret.

```
import java.awt.*;        // denna rad inte vikitigt i tentan
import java.awt.event.*;  // denna rad inte vikitigt i tentan
import javax.swing.*;     // denna rad inte vikitigt i tentan
```

```
public class Dots extends JFrame {

    private class Dot {
        private int x, y;
        private Dot next;
        public Dot (Dot next, int x , int y) {
            this.x = x; this.y = y;
            this.next = next;
        }
        public void draw(Graphics g) {
            g.fillOval(x-3,y-3,6,6);
            if (next != null) { next.draw(g); }
        }
    }

    private class Surface extends JPanel
                          implements MouseListener {
        Dot d = null;
        public void paintComponent(Graphics g) {
            super.paintComponent(g);
            if (d != null) { d.draw(g); };
        }
        public void mouseClicked(MouseEvent e) {
            d = new Dot(d,e.getX(),e.getY());
            this.repaint();
        }
        public void mouseEntered(MouseEvent e) {}
        public void mouseExited(MouseEvent e) {}
        public void mousePressed(MouseEvent e) {}
        public void mouseReleased(MouseEvent e) {}

    }

    public Dots() {
        Surface panel = new Surface();
        panel.addMouseListener(panel);
        add(panel);
        setVisible(true);
    }

    public static void main(String[] args) {
        new Dots();
    }

}
```

Uppgift 5: [15 poäng totalt] *swing, att skriva klass, händelsehantering, Timer*

Denna uppgift ber dig skriva kod för ett program som simulerar ett trafikljus, som består av en röd, en gul och en grön lampa. Så här bör detta program bete sig:

A. När programmet startar bör endast den röda lampan vara tänd.

B. 28 sekunder senare bör den gula lampan tändas. Den röda är ännu tänd.

C. Två sekunder senare slocknar den röda och gula lampan; samtidigt kommer den gröna på.

D. Efter att den gröna lampan varit på i 30 sekunder ska den slockna och den gula ska tändas.

E. Den gula ska lysa i 3 sekunder sedan slocknar den. Den röda tänds. Stegen repeteras från A.

Deluppgifter:

a) Skriv en klass för lampor. Kalla den `Light` och se till att den ärver `JLabel` så att den ser ut ungefär som i exemplen ovan. Klassen bör ha en konstruktör som tar en string som beskriver färgen på lampan när den lyser. När lampan är släkt ska det stå ”svart” i `JLabel`ens textområde. Lampan ska börja i släckt tillstånd. Man ska kunna släcka och tända lampan med instansmetoder. [5 poäng]

```
public class Light extends JLabel {

    String colour;

    public Light (String colour) {
        super("svart",SwingConstants.CENTER);
        this.colour = colour;
    }

    public void turnOn() { this.setText(colour); }
    public void turnOff() { this.setText("svart"); }

}
```

b) Skriv klassen för trafikljuset. Kalla klassen `TrafficLight`.

- `TrafficLight`-klassen bör visa ett fönster på rutan och placera tre lampor (instanser av `Light`) som i illustrationerna ovan. [3 poäng]

- Skapa en `Timer`-instans och se till att trafikljusen beter sig exakt som beskrivet ovan. *Förklara din lösning*. [7 poäng]

```
public class TrafficLight extends JFrame {

    public TrafficLight() {
        JPanel main = new JPanel(new GridLayout(3,1));
        final Light red = new Light("  röd  ");
        final Light amber = new Light("  gul  ");
        final Light green = new Light("  grön  ");
        main.add(red);
        main.add(amber);
        main.add(green);
        add(main);
        red.turnOn();
        pack();
        setVisible(true);
        Timer t = new Timer(1000, new ActionListener() {
                private int time = 0;
                public void actionPerformed(ActionEvent e) {
                    if (time == 28) {
                        amber.turnOn();
                    } else if (time == 30) {
                        red.turnOff();
                        amber.turnOff();
                        green.turnOn();
                    } else if (time == 60) {
                        amber.turnOn();
                        green.turnOff();
                    } else if (time == 65){
                        red.turnOn();
                        amber.turnOff();
                        time = 0;
                    }
                    time = time + 1;
                }
            });
        t.start();
    }

    public static void main(String[] args) {
        new TrafficLight();
    }

}
```

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

`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 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 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 JLabel extends JComponent extends Container extends Component extends Object`
`JLabel(String text)`
Creates a JLabel instance with the specified text.
`void setText(String text)`
Defines the single line of text this component will display.

`Class JPanel extends JComponent extends Container extends Component extends Object`
`JPanel()`
Creates a new JPanel with a double buffer and a flow layout.

`Class MouseEvent extends InputEvent extends ComponentEvent … extends Object`
`int getX()`
Returns the vertical x position of the event relative to the source component.
`int getY()`
Returns the vertical y position of the event relative to the source component.

`Interface MouseListener`
`void mouseClicked(MouseEvent e)`
Invoked when the mouse button has been clicked (pressed and released) on a component.

`Class String extends Object`
`int length()`
Returns the length of this string.
`String substring(int beginIndex, int endIndex)`
Returns a new string that is a substring 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.