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

DAT050 Objektorienterad programmering

Modellsvar för tentamen, tisdag, 2023-01-03, 08:30-12:30

Vitsordsgränser: 3=24p, 4=36p, 5=48p, max 60p.

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

Modellsvar för Uppgift 1: [8 poäng totalt] *grunder i objektorientering och Java*

A. Klassvariabler är globala variabler som tillhör klassen. Instansvariabler är variabler som finns i varje instans av klassen. Det finns alltså en egen version av varje instansvariabel i varje instans av klassen.

```
private static int klassVar;   // klassvariabler deklareras med static
private int instVar;           // instansvariabler deklareras utan static
```

B. En Frame är ett fönster där man kan lägga endast en komponent. En JPanel är en yta man kan rita på och som man kan lägga flera komponenter på. Vanligtvis lägger man en JPanel inuti en JFrame och sedan många komponenter i JPaneln, inklusive andra JPanel instanser.

Modellsvar för Uppgift 2: [15 poäng totalt] *att skriva klass, användning av standard klasser*

A.

```
public class PupilDatabase {

    private class PupilInfo {
        private String name, className;
        public PupilInfo(String name, String className) {
            this.name = name; this.className = className;
        }
        public String getName() { return name; }
        public String getClassName() { return className; }
    }

    private HashMap<Integer,PupilInfo> db;

    public PupilDatabase() {
        db = new HashMap();
    }

    public void setPupilInfo(int pupilID, String name, String className) {
        db.put(pupilID, new PupilInfo(name,className));
    }

    public String getPupilName(int pupilID) {
        PupilInfo i = db.get(pupilID);
        if (i == null) { return null; }
        return i.getName();
    }

    public String getPupilClassName(int pupilID) {
        PupilInfo i = db.get(pupilID);
        if (i == null) { return null; }
        return i.getClassName();
    }

}
```

B.

```
    public List<String> pupilsInClass(String className) {
        List<String> l = new LinkedList<String>();
        for (PupilInfo pi : db.values()) {
            if (className.equals(pi.getClassName())) {
                l.add(pi.getName());
            }
        }
        return l;
    }
```

Modellsvar för Uppgift 3: [10 poäng] *interface, iterators, listor, exceptions*

```
import java.util.*;

public class NextPrevOfList<A> implements NextPrev<A> {

    private List<A> list;
    private int pos;

    public NextPrevOfList(List<A> list) {
        this.list = list;
        this.pos = 0;
    }

    public boolean hasCurr() {
        return (0 <= pos) && (pos < list.size());
    }

    public A curr() {
        if (!hasCurr()) { throw new IllegalArgumentException(); }
        return list.get(pos);
    }

    public void next() { pos++; }
    public void prev() { pos--; }

}
```

Modellsvar för Uppgift 4: [12 poäng totalt] *specifikation och testning*

A. Testkoden kollar inte om elementen som finns i arrayn efter att sort har kört har något samband med elementen som fanns i arrayn innan sort körde. Det kan ju hända att sort har tappat bort element eller bara skrivit nollor in i hela arrayn.

B.

```
public static void sort(int[] a) {
    for (int i=0; i<a.length; i++) {
        a[i] = 0;
    }
}
```

C. Testkoden borde göra en kopia av array a innan den ger arrayn till sort. Efter att sort har kört bör den kolla att alla element i den ursprungliga versionen av a ännu finns i array a efter att sort har kört. Detta gör man enklast med att köra en pålitlig version av sortering på kopian av a och sen jämför man resultatet av sort med den pålitligt sorterade versionen av det ursprungliga innehållet av a.

Modellsvar för Uppgift 5: [15 poäng] *grafiskt användargränssnitt, ritning*

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

public class Histrogram extends JPanel {

    private double[] data;

    public Histrogram(double[] data) {
        this.data = new double[data.length];
        for (int i=0; i<data.length; i++) {
            this.data[i] = data[i];
        }
    }

    public void paintComponent(Graphics g) {
        int w = getWidth();
        int h = getHeight();
        g.setColor(Color.WHITE);
        g.fillRect(0,0,w,h);
        g.setColor(Color.BLACK);
        int w1 = w / data.length;
        for (int i = 0; i < data.length; i++) {
            int h1 = h - (int)(((double)h)*data[i]);
            g.fillRect(w1*i+2,h1,w1-4,h);
        }
    }

}
```