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, onsdag, 2022-08-24, 14:00-18:00

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: [5 poäng] *grunder i objektorientering*

Abstrakta klasser är klasser där man kan deklarerar metoder som abstrakta. En metod är abstrakt om dess signatur deklareras men inte implementeras. Exempel:

```
public abstract class MyAbsClass {

    public abstract int method1();

}
```

Interface är en samling av metodsignaturer. Exempel:

```
public interface MyInterface {

    public int method1();

}
```

Det man kan göra med abstrakta klasser som inte går med interface är att abstrakta klasser kan innehålla kod och instansvariabler. Exempel:

```
public abstract class MyAbsClass2 {

    private int n = 0;

    public int inc() { n = n+1; }

    public abstract int method1();

}
```

Det man kan göra med interface som inte går med abstrakta klasser är implementera flera interface med samma klass. Att ha flera superklasser är inte tillåtet i Java.

```
public class MyExample implements MyInterface, MyOtherInterface {

    ...

}
```

Modellsvar för Uppgift 2: [10 poäng totalt] *private, arv, referensvärden*

A. Java kompilatorn ***accepterar inte*** programmet för att klassen `Q2` försöker komma åt den privata instansvariablen `x` från klass `Q1`. Det går inte att komma åt privata variabler från andra klasser (oberoende om man ärver dem eller inte).

B. Java kompilatorn ***accepterar inte*** programmet för att en klassmetod (`main`) försöker anropa en instansmetod (`getStr`) som om den är en klassmetod.

C. Java kompilatorn ***accepterar*** programmet och skriver ut följande. Siffran 2 finns två gånger i utskriften för att både `array[0]` och `array[1]` pekar på samma objekt.

```
2
2
1
```

Modellsvar för Uppgift 3: [20 poäng totalt] *att skriva klasser, användning av standard klasser*

A.

```
public class IntPair {

    private int fst;
    private int snd;

    public IntPair(int fst, int snd) {
        this.fst = fst;
        this.snd = snd;
    }

    public int getFst() {
        return fst;
    }

    public int getSnd() {
        return snd;
    }

    public String toString() {
        return "(" + fst + "," + snd + ")";
    }

    public boolean equals(Object o) {
        if (o == null) { return false;}
        if (o.getClass() != this.getClass()) { return false; }
        IntPair other = (IntPair)o;
        return (this.fst == other.fst && this.snd == other.snd);
    }

    public int hashCode() {
        return fst + snd * 7000;
    }

}
```

B.

```
public class Board2D<A> {

    private HashMap<IntPair,A> map = new HashMap<IntPair,A>();

    public void setCell(int x, int y, A content) {
        map.put(new IntPair(x,y),content);
    }

    public A getCell(int x, int y) {
        return map.get(new IntPair(x,y));
    }

}
```

Modellsvar för Uppgift 4: [10 poäng] *testning, exceptions*

```
public static boolean foo_test(int i) {
    try {
        int ki = foo_fast(i);
        try {
            int kj = foo_slow(i);
            return (ki == kj);
        } catch (Exception e1) {
            return false;
        }
    } catch (Exception e2) {
        try {
            foo_slow(i);
            return false;
        } catch (Exception e3) {
            return true;
        }
    }
}
```

Modellsvar för Uppgift 5: [15 poäng totalt] *grafik, model view controller*

```
public class View extends JPanel {

    private Model m;

    public View(Model m) {
        this.m = m;
    }

    public void paintComponent(Graphics g) {
        int c = 20;
        int w = getWidth() / c;
        int h = getHeight() / c;
        for (int x=0; x<=w; x++) {
            for (int y=0; y<=h; y++) {
                g.setColor(Color.BLACK);
                if (m.get(x,y)) {
                    g.setColor(Color.WHITE);
                }
                g.fillRect(x*c,y*c,c,c);
            }
        }

    }

}
```