Institutionen för
Data- och informationsteknik
JSk

TENTAMEN
2010-05-24

OBJEKTORIENTERAD PROGRAMMERING
för Z1 (TDA540)

OBS! Det kan finnas kurser med samma eller liknande namn på olika utbildningslinjer. Denna tentamen gäller *endast* för den eller de utbildningslinjer som anges ovan. Kontrollera därför noga att denna tentamen gäller för den utbildningslinje du själv går på.

TID 08:30 - 12:30

Ansvarig: Jan Skansholm, tel. 772 10 12 eller 0707-163230

Betygsgränser: Sammanlagt maximalt 60 poäng.
På tentamen ges graderade betyg:
3:a 24 poäng, 4:a 36 poäng och 5:a 48 poäng

Hjälpmedel: Skansholm, *Java direkt med Swing*, valfri upplaga, Studentlitteratur.
(Understrykningar och mindre anteckningar i boken är tillåtna.)

Inga kalkylatorer är tillåtna.

Tänk på:

- att skriva tydligt och disponera papperet på ett lämpligt sätt.
- att börja varje ny (del)uppgift på nytt blad. Skriv endast på en sida av papperet
- Skriv den (anonyma) kod du fått av tentamensvakten på *alla* blad.

De råd och anvisningar som givits under kursen skall följas vid programkonstruktionerna. Det innebär bl.a. att onödigt komplicerade, långa och/eller ostrukturerade lösningar i värsta fall ej bedöms.

Uppgift 1) a) Vad blir det för utskrift när detta program exekveras?

```
public class K {
  public static void main(String[] argv) {
    int j = 1;
    int[] a = {0, 1, 2, 3};
    int[] b = a;
    hm(a, b, j);
    System.out.println(a[1] + j + b[1]);
  }

   public static void hm(int[] x, int[] y, int k) {
     k = 5;
     x[1] = 2*k;
     y[1] = 4*k;
   }
 }
```

(2 p)

b) Ange för var och en av följande rader om den är korrekt eller inte. Om en rad är felaktig så ange vad felet är.

```
/* rad 1 */ LinkedList<String> l1 = new LinkedList<String>();
/* rad 2 */ List<Integer> l2 = new List<Integer>();
/* rad 3 */ List<l1> l3 = new LinkedList<l1>();
/* rad 4 */ List<String> l4 = new LinkedList<String>(l1);
/* rad 5 */ List<double>  l5 = new ArrayList<double>();
/* rad 6 */ LinkedList<String> l6 = new ArrayList<String>();
```

(3 p)

c) I ett program har man lagt följande rader:

```
String s1 = "HEJ";
String s2 = s1;
String s3 = "hej".toUpperCase();
if (s1==s2)
  System.out.println("lika");
else
  System.out.println("olika");
if (s1==s3)
  System.out.println("lika");
else
  System.out.println("olika");
```

Ange vilken utskrift programmet bör ge och *förklara varför*!

(1 p)

Uppgift 2) I programspråket C kan man som du vet ha pekare till funktioner. Detta är inte tillåtet i Java. Man kan ju inte ha referenser till metoder. Men det är i Java möjligt att kapsla in en metod i en klass. Man kan sedan skapa ett objekt av denna klass och referera till objektet. Det går då att anropa metoden via referensen.

Deklarera ett gränssnitt (interface) med namnet `Function` som innehåller en metod med namnet `apply`. Denna metod skall beskriva en allmän matematisk funktion. Den skall alltså ha en **`double`** som parameter och ge ett resultat av samma typ.

Deklarera därefter två klasser `Square` och `Root` som implementerar gränssnittet `Function`. Klassen `Square` skall kapsla in en metod vilken som resultat ger kvadraten på sin parameter och klassen `Root` skall kapsla in en metod som på motsvarande sätt ger kvadratroten ur sin parameter som resultat.

Skriv sedan en statisk metod med namnet `MakeArray`. Denna metod skall som första parameter ha ett fält `a` (en array) med komponenter av typen **`double`**. Metodens uppgift är att skapa och returnera ett nytt fält som har lika många element som fältet `a`. Komponenternas värden i det nya fältet skall vara avbildningar av motsvarande vär-

den i fältet `a`. Avbildningen skall ske med hjälp av en godtycklig matematisk funktion. Vilken funktion som skall användas skall styras med hjälp av ytterligare en parameter till metoden `MakeArray`.

Antag slutligen att man i ett program har skapat ett fält `v` som har komponenter av typen **`double`**. Skriv ett par programrader som deklarerar och skapar två nya fält `w` och `r`. Dessa fält skall innehålla lika många komponenter som fältet `v`. Fältet `w` skall innehålla kvadraterna på alla talen i `v` och fältet `r` kvadratrötterna ur alla talen i `v`. Du *måste* använda dig av metoden `MakeArray` för att skapa de två fälten `w` och `r`.

(10 p)

Uppgift 3) Vid val räknar man hur många röster de olika partierna har fått. Men egentligen är det hur många *mandat* partierna får som spelar roll. I Sverige används en metod som heter *jämkade uddatalsmetoden* för att omvandla röstantal till mandat. Den är beskriven i lagen och går till som följer:

- Mandaten fördelas ett i taget tills alla mandat är fördelade.
- Vid varje fördelning jämför man partiernas så kallade *jämförelsetal*.
- Det parti som för ögonblicket har högst jämförelsetal får mandatet.
- När ett parti fått ett nytt mandat räknas ett nytt jämförelsetal fram för detta parti.

Jämförelsetalet beräknas på följande sätt: För ett parti som ännu ej tilldelats något mandat beräknas jämförelsetalet genom att partiets röstantal divideras med 1,4. För ett parti som tilldelats minst ett mandat beräknas jämförelsetalet genom att partiets röstantal delas med $2 \times m + 1$, där *m* är antalet hittills erhållna mandat.

Din uppgift är nu att skriva en metod med huvudet

```
public static int[] antalMandat(int[] antalRöster,
                                int totalaAntaletMandat)
```

Metoden får som parametrar dels ett fält innehållande samtliga partiers röstantal och dels ett heltal som anger det totala antalet mandat som ska fördelas. Metoden skall returnera ett lika långt fält som innehåller hur många mandat respektive parti får. Ordningen mellan partierna ska vara samma i "utfältet" som i "infältet", vilket innebär att man inte behöver veta partiernas namn utan bara kan tänka sig dem som parti 0, parti 1 osv. Utforma metoden så att inte jämförelsetal beräknas i onödan.

(12 p)

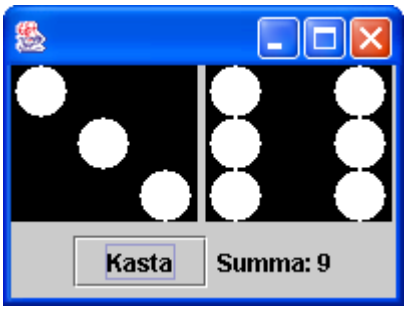
Uppgift 4) Konstruera en klass `Tärning` som skall beskriva en vanlig tärning som man använder t.ex. när man spelar Fia. Följande metoder skall finnas: `kasta`, som ger tärningen ett slumpmässigt värde mellan 1 och 6, `avläsVärde` som avläser tärningens värde och ger det som resultat samt `sättVärde` som har en parameter vilken anger vilket sida på tärningen man vill skall vara upp. Parametern måste förstås ha ett värde mellan 1 och 6. Vidare skall det finnas två konstruktorer, en som initierar tärningen till ett slumpmässigt värde och en som gör det möjligt att ge ett visst första värde till tärningen. Om parametern till den andra konstruktorn eller parametern till metoden `sättVärde` skulle ha ett otillåtet värde skall en exception av typen `IllegalArgumentException` genereras.

Utforma klassen `Tärning` så att objekt av denna klass blir grafiska komponenter som kan visas i fönster. En tärning skall ritas upp med *vita prickar på svart bakgrund*. (Hur tärningar skall ritas upp ser du i figuren till uppgift 5, där två tärningar visas.) Tips: Prickarna kan ritas som separata grafiska komponenter. Om du vill kan du använda dig av följande färdigskrivna klass:

```
import java.awt.*;
import javax.swing.*;
public class Prick extends JPanel {
  public void paintComponent(Graphics g) {
    super.paintComponent(g);
    Insets i = getInsets();
    int w = getWidth()-i.left-i.right;
    int h = getHeight()-i.top-i.bottom;
    int diam = Math.min(h,w);
    int x = i.left + (w-diam)/2;
    int y = i.top  + (h-diam)/2;
    g.fillOval(x, y, diam, diam);
  }
}
```

(12 p)

Uppgift 5) Skriv ett grafiskt program som visar två tärningar:.



När man trycker på knappen `Kasta` skall båda tärningarna kastas så att de får slumpmässiga värden. Dessutom skall summan av värdena visas. Du bör naturligtvis använda dig av klassen `Tärning` från uppgift 4 (även om du inte löst den uppgiften).

(10 p)

Uppgift 6) Ett bilregister finns lagrat i en textfil med namnet `register.txt`. Filen innehåller information om ett stort antal bilar. För varje bil finns det fyra rader i filen. Den första raden innehåller bilens registreringsnummer, den andra bilens fabrikat och modell, den tredje ägarens namn och den fjärde ägarens adress.

Skriv ett program som läser in informationen i filen och lägger den i en avbildningstabell i vilken söknycklarna är registreringsnumren och värdena är den information som finns på de övriga tre raderna för varje bil. Denna information skall internt i programmet sparas i objekt av klassen `Bil` som har följande utseende:

```
class Bil {
  public String modell;
  public String ägare;
  public String adress;

  public Bil(String mo, String äg, String ad) {
    modell = mo; ägare = äg; adress = ad;
  }
}
```

När all information i filen lästs in och lagts i avbildningstabellen skall programmet *upprepade gånger* be användaren ange en bils registreringsnummer. Inläsningen skall ske med hjälp av en dialogruta. För varje angivet registreringsnummer skall programmet slå upp den efterfrågade bilen i avbildningstabellen och visa all information om bilen i en annan dialogruta. Använd en standardklass för att konstruera avbildningstabellen. Välj en klass som gör att sökningarna i tabellen blir så snabba som möjligt. Bilarna behöver inte ligga sorterade i tabellen.

(10 p)