Institutionen för
Data- och informationsteknik
JSk

TENTAMEN
2008-10-24

**OBJEKTORIENTERAD PROGRAMVARUUTVECKLING**

**för IT1 (TDA545) och GU (DIT710)**

*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

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

Betygsgränser: Sammanlagt maximalt 60 poäng.
På tentamen ges graderade betyg:.
CTH: 3:a 24 poäng, 4:a 36 poäng och 5:a 48 poäng
GU: G 24 poäng, VG 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 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) Här ser du ett felaktigt program. Avsikten är att det skall läsa in en text och kontrollera att texten bara innehåller siffror. Programmet innehåller tre olika fel. Det finns ett syntaxfel. Om man rättar detta fel och provkör programmet visar det sig att det innehåller ytterligare två fel: ett logiskt fel som gör att programmet inte utför det som det skall och ett fel som kan resultera i ett exekveringsfel. Ange vilka de tre felen är och ge för varje fel förslag på hur man kan skriva för att rätta felet!

```
import javax.swing.*;

public class FinnTreFel {
  public static void main (String[] arg) {
    String t = JOptionPane.showInputDialog("Ett tal?");
    for (int i=1; i<=t.length(); i++)
      if (t.charAt(i) >= '0' &&  <= '9')
        JOptionPane.showMessageDialog(null,"Talet är OK");
      else
        JOptionPane.showMessageDialog(null,"Inget tal");
    System.exit(0);
  }
}
```

(6 p)

Uppgift 2) a) Skriv en klass `Verktyg` som innehåller en klassmetod med namnet `platsFör`. Metoden `platsFör` skall ha två heltalsparametrar `k` och `n`. Du får förutsätta att parametern `k` innehåller ett positivt ensiffrigt värde, dvs. ett värde i intervallet 0 till 9, och att `n` >= 0. Metoden skall undersöka om, och i så fall var, den siffra som finns i `k` förekommer i talet `n`. Som resultat skall placeringen för siffran `k` anges. Placeringen räknas från höger i talet. Numreringen börjar på 0. Om en siffra förekommer flera gånger skall den första förekomsten (från höger räknat) anges. Om siffran `k` inte finns i talet `n` skall resultatet -1 ges. Det är i denna uppgift inte tillåtet att göra om heltalen till typen `String`.

Några exempel: k=6 och n=2356 skall ge resultatet 0, k=6 och n=6350 resultatet 3, k=6 och n=6360 resultatet 1 samt k=6 och n=4350 resultatet -1.

(5 p)

b) Utöka klassen `Verktyg` med en klassmetod med namnet `innehåller`. Denna metod skall ha samma parametrar som metoden `platsFör`. Samma förutsättningar för parametrarna gäller också. Metoden `innehåller` skall *förutom* att beräkna placeringen för siffran `k` i `n` även undersöka hur många gånger siffran `k` förekommer i `n`. Den skall alltså beräkna två resultat. Du får själv hitta en lösning för hur detta skall gå till. Det är inte heller i denna uppgift tillåtet att göra om heltalen till typen `String`

Några exempel: Om k=6 och n=2356 skall resultaten 0 och 1 ges. Om k=6 och n=6350 skall resultaten 3 och 1 ges. Om k=6 och n=6360 skall resultaten 1 och 2 ges. Om k=6 och n=4350 skall resultaten -1 och 0 ges.

(5 p)

Uppgift 3) Man har i en textfil samlat uppgifter om ett antal personer. För varje person finns två rader i filen. På första raden står personens namn och adress och på andra raden finns personens ålder, längd och vikt. Längden anges i cm och vikten i kg. Man vill göra en medicinsk studie av överviktiga personer och söker därför personer vilkas s.k. body mass index (BMI) överstiger 30. BMI beräknas enligt formeln $m/h^2$ där $m$ är vikten i kg och $h$ längden i m. Skriv ett program som läser filen med personuppgifter. Programmet skall skapa en ny textfil som bara innehåller uppgifterna för dem vilkas BMI överstiger 30. Filnamnen skall läsas från dialogrutor.

Uppgift 4) Uppgiften är att konstruera en klass `ProduktId` vars uppgift är att hålla reda på unika id-nummer för olika produkter. Varje objekt av denna klass skall innehålla ett namn (en `String`) och ett id-nummer (en **`int`**). Id-numren skall vara unika så att inte två eller flera produkter har samma nummer. Om man skriver siffror (slarvigt) för hand kan lätt siffrorna 4 och 9 förväxlas. Samma sak gäller för siffrorna 1 och 7. Av denna anledning har man bestämt att produkternas id-nummer inte skall få innehålla siffrorna 4 och 7.

Klassen skall innehålla instansmetoderna `avläsNamn` och `avläsId` som ger en viss produkts namn resp. id-nummer. Dessutom skall det finnas en klassmetod `avläsAntal` som ger det totala antalet objekt av typen `ProduktId`.

Det måste naturligtvis finnas en konstruktor för klassen `ProduktId`. Denna skall som enda parameter få en `String` med produktens namn. Konstruktorn skall hålla reda på hur många objekt av typen `ProduktId` som skapas. Konstruktorn skall dessutom själv tilldela ett nytt, unikt id-nummer till varje nytt objekt. För att få fram ett nytt id-nummer kan man prova med att addera 1 till det id-nummer man gav det senaste `ProduktId`-objektet. Det nya id-numret får emellertid inte innehålla siffrorna 4 och 7. För att prova om ett nummer innehåller en viss siffra får du gärna använda metoden `platsFör` i klassen `Verktyg` från uppgift 2 a. Om man provar ett nummer men finner att det innehåller en otillåten siffra är det i de flesta fall ineffektivt att bara öka numret med 1 och göra ett nytt prov. Antag t.ex. det förra numret man tilldelade ett `ProduktId`-objekt var 33999. Man ökar då numret med 1 och finner att 34000 är otillåtet. Det är förstås ineffektivt att fortsätta att prova med 34001 osv. Istället skall man addera 1000 och prova med 35000 som visar sig vara tillåtet. Gör på det sättet (inte bara för tusental utan för alla potenser av tio).

(10 p)

Uppgift 5) 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 den s.k. *jämkade uddatalsmetoden* för att omvandla röstantal till mandat. Den går till som följer:

- Mandaten fördelas ett och ett tills alla mandat är slut.
- 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.

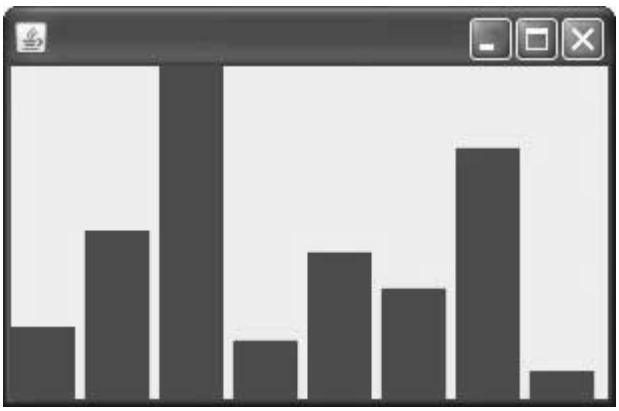
Din uppgift är nu att skriva en metod med huvudet

```
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 fält som innehåller hur många mandat respektive parti får med jämkade uddatalsmetoden. Fältet skall ha samma längd som fältet `antalRöster`. Ordningen mellan partierna ska vara samma i "utfältet" som i "infältet", vilket innebär att man inte behöver veta vilka partier det rör sig om utan bara kan tänka sig dem som parti 0, parti 1 osv.

(12 p)

Uppgift 6) Uppgiften är att konstruera en egen grafisk komponent som visar ett histogram. Klassen skall heta `Histogram` och vara en subklass till standardklassen `JPanel`. I figuren visas ett fönster vilket som enda komponent innehåller en komponent av typen `Histogram` (Men uppgiften är inte att skriva ett fullständigt program.)



Klassen `Histogram` skall internt innehålla en lista av typen `List<Double>`. Denna skall innehålla värdena för staplarna i histogrammet. Det skall kunna finnas godtyckligt många staplar. Följande skall dessutom finnas i klassen `Histogram`:

- En defaultkonstruktor som skapar ett tomt histogram, utan staplar. Den interna listan initieras till en tom lista.
- En konstruktor som har en parameter av typen **`double`**`[]`. Konstruktorn initierar den interna listan så att den innehåller de tal som finns i parametern. Därefter ser den till att staplarna i diagrammet placeras ut.
- En metod `setValues` som har en parameter av typen **`double`**`[]`. Metoden tar bort de gamla värdena ur listan och ersätter dem med de värden som finns i parametern. Därefter ser den till att nya staplar placeras ut.
- En metod `addValue` som lägger till ett nytt värde sist i listan. Därefter ser den till att en ny stapel placeras ut till höger i histogrammet.

Utplaceringen av staplarna görs lämpligen i en privat metod `redrawHistogram`. Det skall finnas lika många staplar som antalet element i listan. Lämna ett litet utrymme mellan varje stapel, så som visas i figuren. Stapeln för det största talet skall precis fylla ut det tillgängliga vertikala utrymmet. De övriga staplarna skall vara skalade i relation till detta.

I din lösning får du förutsätta att det finns en *färdig* klass `Stapel`. (Skriv inte denna klass.) Klassen `Stapel`, som är en subklass till `JPanel`, kan visa vertikala eller horisontella staplar. Den har en konstruktor med en parameter av typen **`boolean`** som anger om en stapel skall vara vertikal eller horisontell. (**`true`** betyder vertikal). I klassen `Stapel` finns en metod, `sättProcent`, som man anropar för att tala om hur stor del av stapeln (dvs. hur många procent av den totalt tillgängliga höjden för en vertikal stapel) som skall vara ifylld. Metoden `sättProcent` skall ha en **`double`** mellan 0 och 100 som parameter. Den icke ifyllda delen av en stapel visas med bakgrundsfärgen. En stapels storlek bestäms av den `LayoutManager` som används.

Klassen `Histogram` skall vara så utformad att en komponent av denna typ kan visa olika histogram vid olika tillfällen. Vare gång någon av metoderna `setValues` och `addValue` anropas skall de tidigare visade staplarna tas bort och nya visas istället

(12 p)