Data och Informationsteknik / Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
Erland Holmström

Göteborg 9 okt 2012

Examination in

Övningstenta

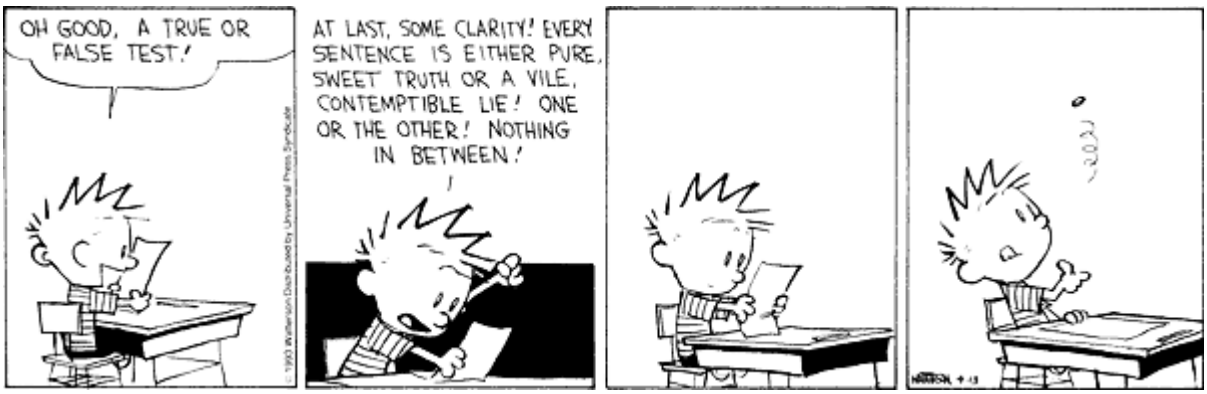
DAY: FRIDAY DATE: 20xx-xx-xx TIME: 8.30-12.30/13.30 ROOM: M

Responsible teacher:	Erland Holmström phone 1007, home 0708-710600
Results:	Are sent by mail from Ladok.
Solutions:	Are eventually posted on homepage.
Inspection of grading:	The exam can be found in our study expedition after posting of results.
	Time for complaints about grading are announced on homepage after the result are published or mail me and we find a time.
Grade limits:	CTH: 3=26/28p, 4=36/38p, 5= 46/48p, max 60p
Aids on the exam:	**See homepage**

Observe:

- Start by reading through all questions so you can ask questions when I come. I usually will come after appr. 2 hours.
- All answers must be motivated when applicable.
- Write legible! Draw figures. Solutions that are difficult to read are not evaluated!
- Answer concisely and to the point.
- The advice and directions given during course must be followed.
- Programs should be written in Java, indent properly, use comments and so on.
- Start every new problem on a new sheet of paper.

Good Luck!



Problem 1. Enkla frågor samt Sant eller falskt? Förklara dina svar. (förklaringen är vanligen viktigare än svaret)

a) Är "5<a && 3>b" detsamma som "! (! (5<a) || !(3>b))"?

b) Vad är värdet av 1653/100 och värdet av 1653%100 :-)

c) Vad skriver följande program ut? Svaret skall motiveras tydligt.

```
public class Minns {
  public static int n = 0;
  public int m = 0;
  public void count() {
    m = m + 1;
    n = n + 1;
  }
}
public class Test {
  public static void main(String[] args) {
    Minns a = new Minns();
    Minns b = new Minns();
    a.count(); a.count(); b.count();
    System.out.println(a.m + " " + b.m + " " + Minns.n);
  }
}
```

(5p)

Problem 2. Testar: rekursion, strängar

a) Skriv en rekursiv funktion `String reverse(string str)` som givet en sträng returnerar strängen baklänges. Om man anger ”topas” skall man alltså få ”sapot” tillbaka.

b) Skriv en funktion `boolean palindrom(String str)` som givet en sträng returnerar true om strängen är en palindrom och false annars. En palindrom är lika oavsett om man läser den fram eller baklänges. Tex ”radar” är en palindrom men ”Mad am I, Madam” är bara en palindrom om man bortser från mellanslag, skiljetecken och stora/små bokstäver. Per definition är tomma strängen och strängar med ett tecken palindromer.

I vårt exempel skall stora/små bokstäver inte spela någon roll så tex ”Radar” och ”radAr” skall bägge vara en palindrom. Mellanslag inuti strängen skall tas hänsyn till men inte i början eller i slutet (radar__ är en palindrom men rad_ar är det inte så tag bort mellanslag i början/slutet.). Skiljetecken skall också spela roll så ”Mad am I, Madam” skall inte betraktas som palindrom.

c) Skriv ett huvudprogram som upprepade gånger läser in en sträng från användaren och sedan skriver ut om det är en palindrom enligt:

Strängen ”indatasträngen” är(är inte) en palindrom

(Fler palindromer: Eve, mumdadmum)

(10p)

Problem 3. *Testar: Enkel klass, val, toString*

Gör en klass, `Lag`, för ett fotbollslag som deltar i en serie. Klassen skall ha utifrån oåtkomliga instansvariabler för *klubbnamnet*, *antal spelade matcher*, *antal vunna*, *antal oavgjorda*, *antal förlorade* och *antal poäng*. Klassen skall ha en konstruktor med vilken vi kan tillverka ett lag med ett visst namn och samtliga antalsuppgifter lika med noll. Dessutom skall det finnas en metod
`public void nyMatch(int vi, int dom)`,
som ändrar antalsuppgifterna utifrån från resultatet vi-dom i en ny match. Vinst ger 3 poäng, oavgjort 1 poäng och förlust 0 poäng. Slutligen en metod som gör att `System.out.println(laget)` skriver ut informationen på formen

HIF 3 2 0 1 6 (antalsuppgifterna i samma ordning som ovan)

(5p)

Problem 4. *Testar:* Snurror och val, slumptal

I denna uppgift skall vi granska påståendet "*Ett positivt heltal är delbart med siffran k om och endast om talets siffersumma är det*". Det är lätt att matematiskt visa att det är sant för $k = 3$ och för $k = 9$ men inte annars. T ex har talet 123111 siffersumman $1+2+3+1+1+1 = 9$ och är därför delbart med 3 och 9. Just nu är du programmerare och inte matematiker, så det begärs att du gör ett program som går igenom fallen $k = 2, 3, 4,\ldots, 9$ och för vart och ett av dem genererar 10 heltalsslumptal i intervallet [0, 999] och testar påståendet för dessa. Programmet skall för varje k-värde räkna antalet avvikelser (brott), dvs då talet är delbart med k men inte siffersumman och vice versa. Om antalet brott är 0, räknar vi påståendet som bevisat (nåja). Ungefär så här skall utskrifterna se ut:

****** Jobbar med k= 2 *****

Talet 589 inte delbart, men siffersumman är det
Talet 15 inte delbart, men siffersumman är det
Talet 875 inte delbart, men siffersumman är det
Talet 672 delbart, men inte siffersumman är det
Talet 323 inte delbart, men siffersumman är det
Antal brott = 5

****** Jobbar med k= 3 *****

Antal brott = 0

Bevisat för 3

.... utelämnade rader

****** Jobbar med k= 9 *****

Antal brott = 0

Bevisat för 9

a) Skriv programmet. Du får använda en färdig metod `int sifsum(int tal)` som beräknar siffersumman av ett tal.

b) Skriv metoden `int sifsum(int tal)` som beräknar siffersumman av ett godtyckligt heltal > 0. Gör bl a upprepade heltalsdivisioner och sluta när kvoten blir 0. Du får inte göra om talet till en sträng/tecken.

(10p)

Du skall skriva en enkel konverterare mellan tum och mm och tvärtom. En tum är 25,4mm. När man startar programmet ser det ut som första figuren nedan. Skriver man i någon av textfälten och trycker return så fylls det andra fältet i. Om man matar in negativt tal eller något som inte är ett flyttal så skall en ”felruta” komma upp med ett meddelande, se nedan.

Problem 5. Testar: klasser, strängar, exceptions

Först skall du skriva en modell-klass, `InchModel`, som kan omvandla mellan tum och millimeter. (ganska mycket detaljer i denna uppgiften, fastna inte)

Du skall lagra tumvärdet. Tum mäts i delar av 1 tum. 1 tum är 25,4mm. Tex så skrivs en kvarts tum som 1/4” och är lika med 1/4*25,4=6,35mm (’”’ är tecknet för tum). Andra värden kan vara 3/4”, 3/8”, 9/16”, 3/64” osv. ”Delarna” är alltid någon av 1, 2, 4, ,8, 16, 32 eller 64. Vi tillåter inte a/0 och ej heller att a eller b i a/b är negativa.

Klassen skall ha följande konstruktorer/metoder:

a) `public InchModel(int a, int b)`

som skapar ett InchModel objekt. Om inte förutsättningarna ovan är uppfyllda så kastas en NumberFormatException med lämplig text. Man sparar också alltid tumvärdet med minsta gemensamma faktorer i täljaren och nämnaren. För att kunna göra denna förkortning behövs en klassmetod som beräknar den största gemensamma divisorn till två heltal m och n, sgd(m, n), som du kan anta finns färdig.

b) `public InchModel(Double d, int precision)`

Från Wikipedia:

“Convert mm to fractional inches for example 6.35 to ¼ as follows

d is the length in mm for the object of interest.
precision is one of 1, 2, 4, ,8, 16, 32 or 64

-Divide d by 25.4 to get the inches in decimal form.

-Take the decimal portion of the result of Step 2 and multiply it by “precision”.

Example with d = 120 and precision = 16:
The point is to determine the nearest 16th of an inch.
120 mm divided by 25.4 yields 4.7244 inches.
Multiplying the 0.7244 by 16 gives 11.591. Round this to 12/16, or 3/4.
The result, therefore, is 4 3/4 inches (= 19/4 inches).

The exact length of a mm when expressed as an inch in decimal form is 0.0393700787.

Note that the smallest difference between two fractional values is 1/64" = 0,396875mm (≈0,4mm). That means that 6.35, 6.45, 6.55, 6.65 are all stored as 1/4 even in the highest precision and 6.75 is stored as 17/64 = 6.746874898...mm.”

c) `public InchModel(String s)`

som bara anropar parse nedan dvs this(InchModel.parse(s));
(så den behöver du inte skriva)

```
public static InchModel parse(String s)
```

tar en sträng och omvandlar till en InchModel. För att förenkla det något så kan du kan anta att strängen är på formen ”a/b” (eller ”a” om b=1) men inte ”c a/b”. Man antas alltså anropa med 9/4 och inte 2 1/4.

d) `public boolean equals(Object obj)`

som avgör om två InchModel objekt är lika.

e) `public String toString()`

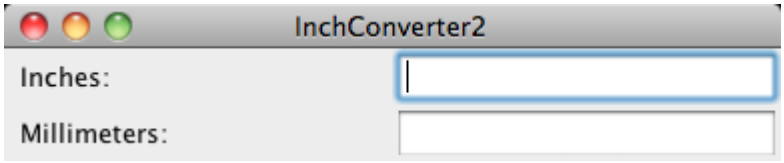
som resturnerar värdet i tum dvs tex ”1/4” utan tum-tecken. Det är ok att returnera tex 57.15 (2 1/4”) som 9/4 (men det är ju snyggare som just 2 1/4”) och noll skall returneras som ”0”.

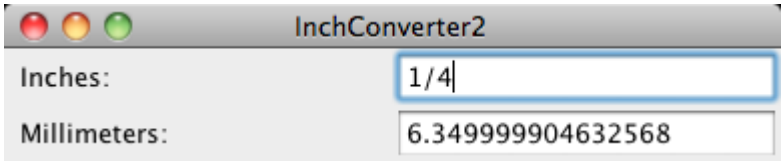
Du kan också anta att det finns en metod `double toMetric()` som returnerar värdet i mm.

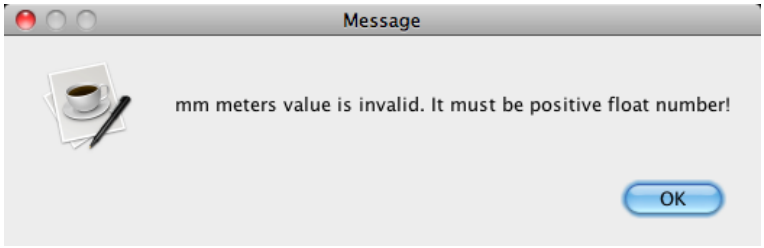
4+4+4+4+2 =18p

Problem 6. *Testar: Klasser, Swing, händelsehantering, exceptions*

Vi vill ju också ha ett snyggt gränssnitt till vår omvandlare. När man startar programmet ser det ut som första figuren nedan. Skriver man i någon av textfälten och trycker return så fylls det andra fältet i. Om man matar in negativt tal eller något som inte är ett inchvärde/ett flyttal så skall en ”felruta” komma upp med ett meddelande, se nedan. Du skall använda en gridlayout och InchModel klassen ovan.







(12p)