TENTAMEN I PROGRAMMERING

Ansvarig:	Jan Skansholm, tel 7721012
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* 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 ditt personnummer 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) Följande (ganska meningslösa) klasser är givna:

```
public class A {
  public int a() {
     return (int) (100*Math.random());
  }
}

public class B extends A {
  public int a() {
    return 0;
  }

  public int b() {
    return (int) (10*Math.random());
  }
}
```

Någon har nu skrivit följande metod i klassen C:

```
1  public class C {
2
3     public static void printValue(A x) {
4        if (x instanceof B)
5           System.out.println(x.b());
6        else
7           System.out.println(x.a());
8     }
9
10 }
```

När man försöker kompilera detta får man följande felmeddelande:

```
C.java:5: cannot resolve symbol
symbol : method b ()
location: class A
           System.out.println(x.b());
                              ^
1 error
```

Förklara vad felet går ut på och skriv om metoden `printValue` så att det fungerar. Raderna är numrerade så du behöver bara ange de rader du ändrar på, lägger till och/eller tar bort.

(4 p)

b) Av dokumentationen av Javas standardklasser mm. kan man utläsa att följande finns i paketet `java.util`:

```
public interface List extends Collection {
  ...
}

public class ArrayList implements List {
  ...
}

public class Vector implements List {
  ...
}
```

Vilka av följande rader är då felaktiga? Beskriv vad felet är för varje felaktig rad.

```
import java.util.*;
class C {
  List l1 = new List();
  List l2 = new ArrayList();
  Vector l3 = new List();
  Vector l4 = new ArrayList();
  Collection l5 = new Vector();
  List l6 = new Collection();
  ArrayList l7 = new Object();
  Object l8 = new ArrayList();
}
```

(3 p)

Uppgift 2) Ett personnummer, t.ex. 561231-4913, består av tio siffror. Efter de sex första siffrorna skall det finnas ett minustecken. Den näst sista siffran är udda för män och jämn för kvinnor. Den sista siffran är en kontrollsiffra. Den beräknas på följande sätt: De nio första siffrorna i personnumret multipliceras omväxlande med 2 och 1, den första med 2 den andra med 1 den tredje med 2 osv. *Siffrorna* i de värden man då får adderas. I personnumret ovan blir det (1+0)+6+2+2+6+1+8+9+2 = 37. (Observera att 10 räknas som 1+0.) Kontrollsiffran bestäms sedan av att den summa man fått plus kontrollsiffran skall vara jämnt delbar med 10. I exemplet blir alltså kontrollsiffran lika med 3.

Skriv en klass `Personnummer` som innehåller ett personnummer. Det skall finnas en konstruktor som får en `String` som parameter. Konstruktorn skall kontrollera att parametern innehåller ett korrekt personnummer (inklusive minustecknet). Det skall också finnas en metod `toString` som returnerar det aktuella personnumret som en `String` (med minustecken) samt två metoder `ärMan` respektive `ärKvinna` vilka ger en **`boolean`** som anger om personen är en man eller en kvinna.

(10 p)

Uppgift 3) Denna uppgift handlar om att skriva ett program som undersöker vilken typ av abonnemang eller kontantkort man skall välja för sin mobiltelefon. (För enkelhets skull skall vi bara bry oss om röstsamtal. Vi bortser därför för kostnader för SMS och MMS.) Antag att man i en textfil `mobildata.txt` har lagt in information om olika typer av abonnemang och kontantkort. Filen kan t.ex. ha innehållet:

```
Normal      99 2.50 2.50 30  0
Natt        79 4.00 0.40  0  0
Maxring    199 0.99 0.99  0  0
Lika        40 2.95 2.95  0  0
Påfyll       0 6.00 0.80  0  0
```

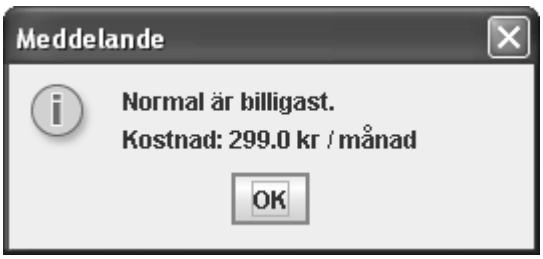
Först på varje rad står namnet på abonnemanget eller kontantkortet. Därefter följer fast kostnad per månad, pris per minut dagtid och pris per minut övrig tid. För vissa abonnemang får man ringa ett visst antal fria minuter per månad utan kostnad. De två sista kolumnerna anger antalet fria minuter under dagtid respektive övrig tid.

a) Konstruera en klass `Abonnemang` som beskriver egenskaperna för en viss typ av abonnemang eller kontantkort. (Ett objekt av denna klass skall alltså innehålla de uppgifter som finns på en viss rad i filen.) I klassen skall det finnas en konstruktor som har som parametrar de data som finns på en rad i filen. Konstruktorn skall kontrollera att alla parametrarna innehåller rimliga värden. Det skall också finnas

två metoder: `avläsNamn` som man kan anropa för att få reda på namnet på abonnemanget eller kontantkortet och `totKostnad` som beräknar den totala kostanden för en månad för att ha ett abonnemang eller kontantkort av det aktuella slaget. Metoden `totKostnad` skall ha två parametrar: antalet minuter man i genomsnitt ringer på en månad dels under dagtid och dels under övrig tid.

(6 p)

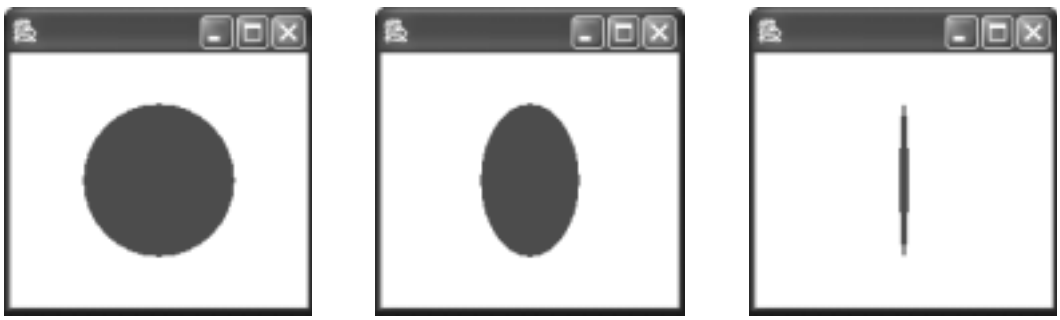
b) Skriv ett fullständigt program som först läser in informationen i filen `mobildata.txt` till programmet. Varje rad i filen *skall* beskrivas med ett objekt av klassen `Abonnemang` ovan. Därefter skall programmet ge den som kör programmet möjlighet att undersöka vilket abonnemang eller kontantkort som är billigast. Det kan se ut på följande sätt när man kör programmet:



Programmet skall klara flera undersökningar i följd. Ovanstående skall alltså kunna upprepas ett godtyckligt antal gånger, ända tills man klickar på knappen 'Avbryt'.

(8 p)

Uppgift 4) Skriv ett program som visar en roterande disk, enligt bildsekvensen nedan. Diametern på disken får sättas till ett konstant värde, t.ex. 100 pixlar.
Tips: Animeringen kan göras med hjälp av en oval, vars diameter ändrar storlek



(12 p)

Uppgift 5) Du har kanske vid något tillfälle när du besökt Liseberg eller något annat nöjesfält spelat det spel som går ut på att man med en klubba skall slå ekorrar i huvudet. I en stock finns ett antal hål. En ekorre i taget dyker upp ur ett slumpmässigt hål och det gäller att hinna slå på ekorren innan den försvinner ner i hålet igen. Uppgiften är att skriva en något mindre dramatisk variant av detta spel. När man kör spelet kan det se ut som i nedanstående figur.

I fönstret finns överst en rad med knappar. När man startar spelet genom att klicka på knappen *Nytt spel*, kommer en knapp i taget att vara aktiverad i den övre raden. Efter en viss tid kommer den aktiverade knappen att deaktiveras och en *annan* slumpmässigt vald knapp kommer att aktiveras i stället. Det gäller förstås för användaren att hinna klicka på den aktiverade knappen innan den deaktiveras. Varje gång man lyckas med detta får man ett poäng. De uppnådda poängen visas i fönstret längst ner.

Varje spel skall pågå i 30 sekunder. Efter denna tid skall samtliga knappar i den övre raden deaktiveras. Användaren kan när som helst starta ett nytt spel genom att klicka på knappen *Nytt spel*. Poängen skall då nollställas.

Den klass som beskriver fönstret skall utformas på ett generellt sätt. Detta innebär att antalet knappar i övre raden skall kunna varieras mellan olika körningar. Den tid som varje knapp är aktiverad skall också kunna varieras. Klassens konstruktor skall därför ha två parametrar, en som anger antalet knappar och en som anger hur många millisekunder en knapp skall vara aktiverad. Utforma metoden `main` så att antalet knappar och antalet millisekunder kan ges som parametrar på kommandoraden när man startar programmet. (Du får alternativt använda dig av dialogrutor i metoden `main` för att läsa in denna information.)

Tips: Använd ett fält med knappar. Använd också två `Timer`-objekt, ett som efter 30 sekunder signalerar att spelet är slut, och ett som signalerar när det är dags att deaktivera en knapp och aktivera en annan.

Om användaren lyckas klicka på en knapp som är aktiverad skall programmet ge ifrån sig ett ljud (beep). Den aktuella knappen skall då deaktiveras och en *annan* slumpmässigt vald knapp aktiveras i stället. Timern skall också startas om så att den knapp som blir aktiverad kommer att vara aktiverad rätt tid.

Om användaren inte hinner klicka på en aktiverad knapp skall en annan slumpmässigt vald knapp aktiveras i stället.

För att användaren tydligt skall kunna se vilken knapp som är aktiverad skall texten på denna visas med röd färg.

(17 p)