Övning Klasser och Objekt

Tänk noga igenom vilket publikt gränssnitt klassen du skriver ska ha gentemot övriga klasser, d.v.s. vilka publika metoder som ska finnas.
Använd inkapsling, med hjälp nyckelordet private, av datamedlemmar samt av metoder som är för internt bruk inom klassen.

Skriv för varje uppgift en main-metod där du skapar objekt av klassen och testar funktionaliteten.

Övningsuppgifter

Uppgift 1, Tärning

Skriv en klass, Dice, som modeller en tärning.
En tärning vänder alltid en viss sida upp (ett tal 1 - 6). Detta representeras med en datamedlem, som jag kallar value.
En tärning kan kastas och då slumpas en ny sida fram. Det ska vara möjligt att skriva ut information om tärningen snyggt. Vi behöver alltså följande metoder:

- En konstruktor. Låt en nyskapad tärning vända sidan 1 upp.
- En metod, throwDice, för att kasta tärningen, d.v.s. slumpa ett nytt värde.
- En metod, getValue, för att avläsa värdet på en tärning.
- En metod, toString, för att få textinformation om tärningens tillstånd.

För att slumpa tal kan man använda sig av Math.random() som returnerar ett slumptal i intervallet [0.0..1.0[. Heltal i intervallet 1 – 6 fås med `value = (int)(Math.random()*6)+1;`

Skriv också ett main där 2 tärningar kastas ett antal gånger och resultatet skrivs ut. Skriv även kod som kastar en tärning 1000 gånger och beräknar frekvensen för de olika sidorna.

Uppgift 2, Student

Skriv en klass som representerar information om en student. En student har ett namn, går på en viss utbildning och har tagit ett antal poäng. Det ska finnas metoder för att lägga till poäng, addCreditPointss, byta utbildning, setProgram, samt skriva ut information om en student.

Skriv också en main-metod där du skapar några studenter och testar metoderna.

Uppgift 3, Account

Hämta filen Account.java, som innehåller definitionen av bankkonto, från föreläsningen. Studera klassen. Lägg till eller ändra följande:

- En privat datamedlem, typ `long`, som representerar kontonumret.
- Ändra konstruktorerna så att ett objekt ges värde på kontonumret när det skapas.
- Lägg till en accessmetod för att avläsa kontonumret.
- Uppdatera metoden toString().

Skriv en main-metod där du skapar några olika Account-objekt och testar de nya metoderna.

Uppgift 4, Spelkort

Skriv en klass, Card, som är en mall för spelkort.
Klassen ska ha 2 datamedlemmar, färg (suit) respektive valör (rank). Enklast representeras dessa som heltal, 1 – 4 respektive 1 – 13.

Klassen bör ha följande metoder:

- Konstruktor som tar emot 2 heltal. Fundera över hur felaktiga indata ska hanteras.
- Accessmetoder: getRank, getSuit.
- toString() som returnerar en snyggt formaterad textsträng med information om kortet.

Observera att ett skapat kort inte ska kunna förändras, d.v.s. det ska vara immutable[1]. Är det så här? Vad finns det för poäng med att låta datamedlemmarna vara privata (när man ändå kan komma åt dessa via publika accessmetoder)?

Skriv sedan en klass, PlayCards, där du skapar några Card-objekt och testar metoderna.

Lite knepigare

Försök också skriva, och testa, dessa metoder:

- boolean equals(Card c) som tar ett annat kort som parameter och undersöker om korten har lika färg och valör
- int compareRank(Card c), returnerar skillnaden i valör. Metoden kan användas för att rangordna kort i ett spel.

Uppgift 5, Post-it-lappar

Skapa en klass PostIt, som modellerar kom-ihåg-lappar. Klassen ska ha datamedlemmarna:

- String note, själva meddelandet
- 3 heltal år, mån, dag för datumet då lappen skrevs.

Skriv en konstruktor, en metod append(String s), som lägger till ny text till notisen, samt metoden toString().

Skriv ett testprogram där du skapar PostIt-objekt, lägger till text och skriver ut.

Uppgift 6, Post-it-lappar, fortsättningen

Utöka klassen PostIt med följande:

Lägg till en konstruktor som bara tar texten som parameter. Konstruktorn ska initiera med dagens datum.
Dagens datum får du från klassen GregorianCalendar, se API-dokumentationen:

```
GregorianCalendar c = new GregorianCalendar();
år  = c.get( GregorianCalendar.YEAR );
mån = c.get( GregorianCalendar.MONTH ) + 1;
dag = c.get( GregorianCalendar.DAY_OF_MONTH );
```

[1] Klassen String är ett annat exempel på en klass vars objekt är immutable. En textsträng kan alltså i Java inte förändras när den väl skapats; metoder som string.toUpperCase ändrar alltså *inte* i befintlig text utan skapar istället ett helt nytt String-objekt.

Uppgift 7, Post-it-lappar, fortsättningen på fortsättningen

Lägg till en datamedlem som representerar kom-ihåg-lappens nummer i PostIt-klassen.
Gör så att varje nytt PostIt-objekt automatiskt numreras när det skapas. Detta kan åstadkommas med hjälp av klassdata, en *static* variabel, som håller reda på hur många objekt som skapats. Denna räknas upp i klassens konstruktor (som ju anropas när ett objekt skapas).
Skriv också en static metod som returnerar antalet objekt som skapats, samt ändra toString.

Uppgift 8, Räkna med bråk

Skriv en klass Rational, som lagrar information om ett bråk (rationellt tal), d v s två heltal, täljare och nämnare. Notera att nämnaren aldrig får vara 0.
Skriv konstruktorer så att man kan skapa ett bråk med två, en eller ingen parameter (i sista fallet lagras 0/1).

Skriv metoden toString() så att en snygg utskrift fås, accessmetoder samt en metod `add( Rational r )` som adderar två bråk och returnerar ett nytt (förkortat) bråk, summan[2].

Fundera själv över vilka ytterligare metoder som är bra att ha och skriv dessa.

Uppgift 9, Mera bråk

Det är lämpligt att ett bråk alltid representeras på förenklad form (t ex 10/15 = 2/3). Vilka metoder måste du ändra för att detta villkor alltid ska vara uppfyllt? Gör dessa ändringar och testa klassen.

Tips: För att på ett enkelt sätt alltid hålla bråket på enklaste form kan du skriva en privat hjälpmetod som dividerar täljare och nämnare med största gemensamma delare som t ex fås så här:

```
private int gcd( int m, int n )  { // Greatest Common Divisor
  m = Math.abs( m );
  n = Math.abs( n );
  while( m != n ) {
    if( m > n )
      m -= n;
    else
      n -= m;
  }
  return m;
```

[2] $\frac{a}{b} + \frac{c}{d} = \frac{ad + bc}{bd}$