PG2100 Oppgaver uke 3, 2014

Oppgave 1

a) En terning (`Die`) kan trilles og vise et antall øyne på siden som vender opp etter hvert kast. Antall øyne kan være fra 1 til 6. Lag en klasse `Die`. Klassen skal ha et attributt `value` som skal lagre antall øyne som vises etter hvert kast. Klassen skal også ha to metoder

- metoden `roll` som triller terningen og genererer et tilfeldig antall øyne i området 1 – 6. Bruk f.eks. `Math.random()` til å produsere dette tilfeldige tallet. Metoden skal ikke returnere noen verdi, men lagre resultatet i terningens attributt `value`. For å produsere et tilfeldig tall fra 1 til 6, kan man benytte følgende "formel":

```
value = (int) (Math.random() * 6) + 1
```

- metoden `getValue()` som returnerer antall øyne som fremkom ved siste "roll".
- metoden `toString()` som returnerer en `String` som viser terningens tilstand – se under:

```
Terningen viser: 6
```

Klassen skal også ha en konstruktør uten parametere som gir attributtet `value` en tilfeldig verdi 1 til 6.

b) Lag et program, `DieTest`, som oppretter et objekt av klassen `Die` og som foretar noen kast med terningen og skriver ut verdien som terningen viste i hvert kast.

c) Tegn et UML klassediagram for klassen `Die`. Se forelesningen for oppsett.

Oppgave 2

En dato består av et dagnummer (`dayNumber`) - `int`, månednummer (`monthNumber`) - `int` og år (`year`) - `int`. En dato kan flyttes til neste dag. Dette kan eventuelt bli en ny måned og eventuelt et nytt år.
Deklarer en klasse `Date` med egenskaper som beskrevet over, samt et klientprogram som tester klassens metoder – se eksempler på kjøring under, der objekter er opprettet med ulike startdatoer for å teste overganger til en ny dag, ny måned og nytt år.
Klassen skal ha en `toString`-metode som returnerer en dato på 'lesbar' form – se under.

```
17. januar 2012
18. januar 2012
19. januar 2012
      ----
31. desember 2011
1. januar 2012
2. januar 2012
      ----
30. april 2011
1. mai 2011
2. mai 2011
      ----
28. februar 2011
1. mars 2011
2. mars 2011
```

Hint: Bruk en `switch`-setning (se side 1154-1156/1144-1146) for å knytte månednummer til månednavn.

Oppgave 3

11. Suppose the following `BankAccount` class has been created:

```
1  // Each BankAccount object represents one user's account
2  // information including name and balance of money.
3  public class BankAccount {
4      String name;
5      double balance;
6
7      public void deposit(double amount) {
8          balance = balance + amount;
9      }
10
11     public void withdraw(double amount) {
12         balance = balance - amount;
13     }
14 }
```

Add a field to the `BankAccount` class named `transactionFee` for a real number representing an amount of money to deduct every time the user withdraws money. The default value is $0.00, but the client can change the value. Deduct the transaction fee money during every `withdraw` call (but not from deposits). Make sure that the balance cannot go negative during a withdrawal. If the withdrawal (amount plus transaction fee) would cause it to become negative, don't modify the balance at all.

12. Add a `toString` method to the `BankAccount` class from the previous exercise. Your method should return a string that contains the account's name and balance separated by a comma and space. For example, if an account object named `yana` has the name `"Yana"` and a balance of `3.03`, the call `yana.toString()` should return the string `"Yana, $3.03"`.

13. Add a `transfer` method to the `BankAccount` class from the previous exercises. Your method should move money from the current bank account to another account. The method accepts two parameters: a second `BankAccount` to accept the money, and a real number for the amount of money to transfer. There is a $5.00 fee for transferring money, so this much must be deducted from the current account's balance before any transfer. The method should modify the two `BankAccount` objects such that "this" current object has its balance decreased by the given amount plus the $5 fee, and the other account's balance is increased by the given amount. If this account object does not have enough money to make the full transfer, transfer whatever money is left after the $5 fee is deducted. If this account has under $5 or the amount is 0 or less, no transfer should occur and neither account's state should be modified. The following are some example calls to the method:

```
BankAccount ben = new BankAccount();
ben.deposit(80.00);
BankAccount hal = new BankAccount();
hal.deposit(20.00);
ben.transfer(hal, 20.00);    // ben $55, hal $40  (ben -$25, hal +$20)
ben.transfer(hal, 10.00);    // ben $40, hal $50  (ben -$15, hal +$10)
hal.transfer(ben, 60.00);    // ben $85, hal $ 0  (ben +$45, hal -$50)
```

(fra 3. utgave)