SI-pass 2 OOP

Handledare: Oscar Söderlund

Mail: soscar@student.chalmers.se

1. Implementera en generell metod vars syfte är att läsa in ett heltal från kommandoraden. Metoden skall ha parametrar som låter den anropande programmeraren specificera inom vilket intervall det inmatade talet får befinna sig, samt vilket eventuellt meddelande användaren ska få innan programmet stannar och väntar på input.

   Om talet användaren matar in inte ligger i det specificerade intervallet, skall metoden fortsätta fråga användaren tills korrekt input erhållits.

   Följande exempelkörningar visar hur er metod skall kunna andropas med olika parameteruppsättningar:

   Exempelkörning1:

   ```
   > Ange ett heltal mellan 1 och 100: 200
   > Ange ett heltal mellan 1 och 100: 42
   ```

   Exempelkörning 2:

   ```
   > Ange ett negativt tal: 5
   > Ange ett negativt tal: -1
   ```

   Tips: Bestäm tidigt hur ni vill att metodhuvudet, dvs parametrar och returvärde, skall se ut.

   Diskutera: Vilka argument bör skickas till metoden om man skall kunna mata in vilket tal som helst? Vad sätter gränsen för hur stora tal som kan matas in?

2. Skriv ett program som läser in fem stycken heltal från kommandoraden och sedan avgör och skriver ut det största talet.

   Exempelkörning:

   ```
   > Ange ett tal: 10
   > Ange ett tal: 20
   > Ange ett tal: 100
   > Ange ett tal: 30
   > Ange ett tal: 40
   > Det största talet: 100
   ```

3. Skriv en metod som låter användaren gissa sig fram till ett okänt slumptal. Talet skall genereras slumpmässigt, och således vara olika varje gång man anropar metoden. Vilket intervall slumptalet får befinna sig i skall specificeras av parametrar till metoden.

   Metoden skall be om gissningar tills att det sökta talet är inmatat och därefter avslutas. Metoden skall hjälpa användaren gissa genom att tala om ifall det senast inmatade talet är större eller mindre än det sökta talet.

(vänd)

Skriv sedan en mainmetod som låter användaren gissa tal mellan 1 och 1000.

Exempelkörning:

```
> Gissa ett tal: 952
> Lägre!
> Gissa ett tal: 12
> Högre!
> Gissa ett tal: 42
> Rätt!
```

4. Fibonaccitalen är en oändlig sekvens av heltal som inleds med 0 följt av 1, där varje nytt tal i sekvensen erhålls genom att ta summan av de två föregående talen.

   Ex: 0 1 1 2 3 5 8 13 21 34 55 89 144 233 377

   Matematiskt kan fibonaccifunktionen defineras:

   ```
   f(0) = 0
   f(1) = 1
   f(n) = f(n-1) + f(n-2), n > 1
   ```

   Skriv en motsvarande funktion i Java, fibonacci, som givet en positiv heltalsparameter n returnerar det n:te talet i fibonaccisekvensen. Anropet fibonacci(6) bör t.ex returnera talet 8.

   Skriv sedan en mainmetod som läser in ett heltal $>= 0$ från kommandoraden och skriver talet på motsvarande position i fibonaccisekvensen.

   Exempelkörning:

   ```
   > Mata in ett heltal >= 0: 6
   > Fibonaccital nr 6: 8
   ```

5. När vi skriver tal använder vi oftast basen 10.

   Detta medför att talet 47915 kan skrivas om som: $4 * 10^4 + 7 * 10^3 + 9 * 10^2 + 1 * 10^1 + 5 * 10^0$

   Skriv ett program som läser in ett positivt heltal från kommandoraden och skriver ut det på den ovan givna formen.

   Exempeloutput:

   ```
   > Mata in ett tal: 385
   > 385 = 3*10^2 + 8*10^1 + 5*10^0
   ```

Tentauppgifter

En ekvation av formen Ax + By + Cz = D, där A, B, C och D kallas en diofantisk ekvation. (Diofantos var en grekisk matematiker som levde omkring 250 f. Kr.) Ekvationen kan ha noll eller flera lösningar. Vi är i denna uppgift bara intresserade av ekvationer i vilka A, B, C och D är heltal som är större än noll och vi söker bara sådana lösningar där x, y och z alla är större än eller lika med noll. Därmed vet vi t.ex. att x <= D/A. Skriv ett program som läser in de fyra talen A, B, C och D och sedan skriver ut alla lösningar enligt ovan och slutligen även en uppgift om antalet lösningar. Så här kan två körningar av programmet se ut:

Ange talen A, B, C och D: 2 4 6 7
0 lösningar!

Ange talen A, B, C och D: 2 4 6 8
0 2 0
1 0 1
2 1 0
4 0 0
4 lösningar!

(10 p)

När man skall spela in ett TV-program på video är det ibland lite knöligt att räkna ut om bandet räcker. Detta skulle bli enklare om man hade ett program som räknade ut hur många minuter det är mellan två klockslag. Skriv ett sådant program.

Programmet skall från två dialogrutor läsa in starttiden respektive sluttiden för programmet som skall spelas in. Tiderna skall anges med formen tt.mm i dialogrutorna. Som resultat skall programmet visa hur många minuter det är mellan de två tidpunkterna. Programmet skall även klara av det fall när man börjar en inspelning före midnatt och avslutar den efter.

Tips: Räkna om de båda tiderna till antalet minuter som gått sedan det aktuella dygnets start.

(10 p)

Ett positivt heltal N kallas lyckligt (happy) om följande förfarande leder till talet 1:

Upprepa: N = numman av kvadraterna på siffrorna i N

T.ex är 44 ett lyckligt tal, eftersom: 440 -> 16+16+0=32 -> 9+4=13 -> 1+9=10 -> 1

Uppenbarligen är 1 ett lyckligt tal. Positiva heltal som inte leder till 1 kallas olyckliga. De leder istället till 4 som vid fortsatt iteration ger en cykel 4 -> 16 -> 37 -> 58 -> 89 -> 145 -> 42 -> 20 -> 4.

a) Skriv en metod int sumOfSqr(int n) som ger efterföljaren till n enligt processen ovan. Dvs resultatet skall vara summan av kvadraterna på siffrorna i n. Det får här och i fortsättningen förutsättas att resultatet är mindre än det största tillåtna heltalet. (Integer.MAX_VALUE)

b) Skriv en metod boolean isHappy(int n) som avgör om n är lyckligt eller ej. Det får förutsättas att antingen 1 eller 4 nås. Utnyttja metoden ovan.

c) Skriv ett fullständigt program som bestämmer och skriver ut antalet lyckliga tal mindre än 1000. Utnyttja metoderna ovan, som naturligtvis inte behöver upprepas.

(10 p)