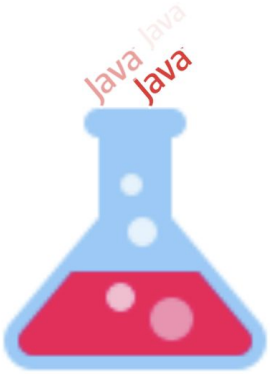

TDDC77 - Objektorienterad programmering

Laboration 2

Institutionen för datavetenskap
Linköpings universitet

2.1 Förberedelseuppgift

Fyll i luckorna och svara på följande frågor och redovisa för din labbhandledare. Kan du inte svaren får du läsa på i boken.

En uppsättning med era variabler i rad kallas en . . . . . och man indexerar en sådan från . . . . . till . . . . . Man kan dela upp funktionaliteten i sitt program/sin klass i flera . . . . . och sådana kan . . . . . nagot till den som anropade.

Skriv en while-loop som skriver ut alla tecken i `char[] tecken = { 'a', 'b', 'c'};`. Utskrift sker med `System.out.println(tecknet);`.
Svar:

Skriv en for-loop som skriver ut alla tecken i en sträng var för sig, Till exempel, ur `String sträng = "Veckans bubblare";`. Utskrift sker med `System.out.print(tecknet);`.
Svar:

(Variabelnamnet `sträng` innehåller ju ett ä. Det borde vara så att olika nationella tecken fungerar utan problem i Java. Om ni alltid använder samma dator/system så borde det fungera, men om ni t.ex. skall flytta kod mellan hemmadatorn och IDA så kan det uppstå problem. Försök undvika variabelnamn liksom `räksmörgås`.

2.2 Få igång Eclipse IDE

Från och med nu får ni (om ni så önskar) utveckla Java-program i ett program(stor editor) som heter *Eclipse*. *Eclipse* är en integrerad utvecklingsmiljö, IDE, som erbjuder er många fördelar när ni programmerar, bland annat syntax highlighting, inkrementell kompilering (kompilering görs hela tiden i bakgrunden), samt stöd för debugging och refactoring. *Eclipse* är dessutom världens mest populära miljö för Java så chansen är stor att ni kommer arbeta med *Eclipse* i framtiden. *Eclipse* är gratis och finns för Windows, Mac, Linux, Solaris med mera. Alltså går det utmärkt att arbeta hemma, gå bara till www.eclipse.org och ladda ner rätt version för er dator. Från terminalen kan ni starta *Eclipse* med:

```
> /opt/eclipse/4.10/eclipse&
```

Ni kommer att bli tillfrågade om var ni vill spara er Eclipse-arbetsyta (workspace). Välj `er_hemkatalog/workspace` och tryck OK.

2.3 Bekanta dig med laborationsmiljön

Denna övning går ut på att lära er in- och utmatning av text samt bekanta er med *Eclipse*. Ju mer ni upptäcker och testar i *Eclipse* nu desto snabbare kommer resten av labbserien går. Längs vägen kommer ett par frågor som ni ska besvara.

Starta *Eclipse* om ni inte har gjort det. Välj Workbench. Till vänster finns en flik med namnet *Package Explorer*. Högerklicka på den tomma, vita ytan och välj *New -> Project...* . I fönstret som kommer upp väljer ni Java Project-wizarden och trycker på *Next*. Döp ert projekt till något lämpligt i stil med *TDDC77Labs* och tryck sen Finish.

Nu ska en projektmapp med bestämde namnet ha kommit upp i *Package Explorer*. Högerklicka på projektmappen och välj *New -> Package...* . Döp paketet till exempelvis lab2 eftersom det är labb 2 ni ska börja arbeta med. Observera att paketnamnet precis som metoder och variabler ska börja med liten bokstav. Om ni börjar paketnamnet med stor bokstav kommer *Eclipse* att varna er *"Discouraged package name. By convention, package names usually start with a lowercase letter*". När paketet är döpt, då trycker ni på Finish. Nu ska ni ha en liten vit paketsymbol i er projektmapp. Ni kommer att skriva klassen AreaCalculator för att bekanta er med *Eclipse*.

Eclipse tutorial:
https://www.vogella.com/tutorials/Eclipse/article.html

2.4 AreaCalculator

Klassen `AreaCalculator` har som uppgift att beräkna ytan på en triangel givet dess bas och höjd, samt beräkna ytan på en cirkel givet dess radie.

Sätt upp ett projekt i Eclipse

Högerklicka på den vita paketsymbolen och välj *New -> Class...* . Nu får ni upp ett fönster med en massa valmöjligheter. Två av textfälten är redan ifyllda *"Source folder"* och *"Package"*. Det är för att ni markerade det paketet i den projektmappen när ni valde att skapa en ny klass.

Döp er klass till exempelvis `AreaCalculator`. Notera att klassnamnet börjar med stor bokstav och att det är ett substantiv. Klassnamn bör vara substantiv eftersom det känns naturligt att instansiera objekt som är någon sorts sak, dvs substantiv. Ni kan också kryssa för att ni vill skapa en metodstubbe (eng. Method stub) för metoden `public static void main(String[] args)`. Tryck nu på Finish. Ni bör nu ha fått upp källkoden till er nya klass under fliken AreaCalculator.java:

```
package lab2;
public class AreaCalculator {
	/**
	* @param args
	*/
	public static void main(String[] args) {
	// TODO Auto-generated method stub
	}
}
```

Lägg till kod för att skriva ut någon text till skärmen. Prova att köra igång programmet genom att klicka på menyvalet *Run -> Run As -> Java Application*. Under källkoden finns en flik som heter *Console*. Den fungerar som ett skalfönster, dvs det är där programmet skriver saker och det är där användare kan skriva in saker om programmet ber om det.

När ni skriver kod kan det hända att ni får ett kompileringsfel. *Eclipse* kompilerar hela tiden er kod i bakgrunden och så fort ni får ett kompileringsfel så markeras det med röd understrykning och en glödlampa med ett rött kryss. På samma sätt som med varningar (gul understrykning) så kan ni få reda på vad det är som är felet genom att hålla muspekaren över felmarkeringen.

2.5 GissaTal - ett litet spel*

Nu över till en lite mer intressant uppgift. Ni ska programmera ett litet spel som heter GissaTal. Första spelaren matar in ett heltal mellan 0 och 100. Sen får den andra spelaren gissa vilket tal det var. För varje gissning får andra spelare reda på om gissningen var rätt, för liten eller för stor.

1. Skapa ett program som ber spelaren ett att mata in ett heltal mellan 0 och 100. Därefter ska programmet tömma skalfönstret genom att skriva ut radbrytningar tillräckligt många gånger.
2. Nu ska programmet be spelare två att gissa på ett heltal mellan 0 och 100. Om gissningen är fel så ska programmet be om ett nytt heltal. Om gissningen är rätt ska programmet gratulera till vinsten, skriva ut antalet gissningar som behövdes för att hitta rätta talet och sen avslutas.
3. Sist ska ni införa ledtrådar. Spelare två ska få veta om hans/hennes gissningar för liten eller för stor.

En programkörning efter steg 3 skulle kunna se ut så här:

```
[john@emil23 java]$ java GissaTal
Spelare ett, Mata in ett heltal mellan 0 och 100: 18
*** Skärmen töms ***
Spelare två, chansa på ett heltal mellan 0 och 100: 20
För stort!
Spelare två, chansa på ett heltal mellan 0 och 100: 5
För litet!
Spelare två, chansa på ett heltal mellan 0 och 100: 15
För litet!
Spelare två, chansa på ett heltal mellan 0 och 100: 18
Rätt! Du behövde 4 gissningar.
[john@emil23 java]$
```

Kör programmet för handledaren. Kan ni komma på någon effektiv gissningsstrategi? Hur många gissningar behöver man maximalt med er strategi? Skulle man kunna skriva ett Javaprogram som spelade med er strategi, dvs där en människa matar in ett heltal och sen låter programmet försöka hitta talet med så få gissningar som möjligt (utan att fuska)? Prova!

2.6 Textstatistik - ett ordbehandlingsverktyg*

Förr i tiden skrev man långa dokument på skrivmaskin (har ni provat någon gång?). Fördelarna med att skriva på dator är många och stora. Några av fördelarna är att datorn mycket snabbt kan gå igenom texten och hjälpa till med stavning, grammatik, avstavning och statistik.

Det sistnämnda ska ni nu prova på att programmera - ett program som tar fram statistik på antal ord, antal meningar, antal bokstäver, antal skiljetecken samt medellängd på orden. Som indata tar ni en vanlig textfil som ni skrivit (t.ex. i Atom). En körning av programmet skulle kunna se så här ut:

```
[john@emil23 java]$ java Textstatistik mintextfil.txt
Antal ord: 26
Antal meningar: 6
Antal bokstaver: 97
Antal skiljetecken: 8
Medellangd pa ord: 3.73 bokstaver
[john@emil23 java]$
```

(Obeservera att meddellängden är avrundad till ett vettigt antal decimaler.) Ni får också några tips: Se till att avsedda delar av er kod ser ut som nedan. Ta sedan reda på om det finns några bra metoder i klassen BufferedReader (https://docs.oracle.com/javase/8/docs/api/java/io/BufferedReader.html). Kör och redovisa programmet för handledaren.

```
import java.io.*;
class Textstatistik {
public static void main(String[] args) {
      // Läser in filen som gavs som första argument
      BufferedReader input = new BufferedReader(new FileReader(args[0]));
}
```

Skriv en metod som tar reda på om ett visst tecken finns i en uppsättning av tecken och använd den på något valfritt sätt i programmet. Till exempel kan du använda *memberOf* för att utreda om inlästa tecknet är ett skiljetecken eller inte.

```
public static boolean memberOf(char,String)
```