Chapter 2: Data and Expressions

I dessa uppgifter kommer du att lära dig hur man använder variabler av olika datatyper samt matematiska formel för att lösa enkla problem.
Du lär dig skriva enkla java program.

Del 1: Övningar

Om du aldrig har programmerat är det obligatoriskt att du läser kapitlen 2 i boken samt skriver första programexemplet från boken innan du börjar med övningarna.

Trace

a) Trace the execution of the following program assuming the input stream contains the numbers 10, 3, and 14.3. Use a table that shows the value of each variable at each step. Also show the output (exactly as it would be printed).

```
// FILE: Trace.java
// PURPOSE: An exercise in tracing a program and understanding
// assignment statements and expressions.

import java.util.Scanner;
public class Trace
{
public static void main (String[] args)
{
int one, two, three;
double what;
Scanner scan = new Scanner(System.in);
System.out.print ("Enter two integers: ");
one = scan.nextInt();
two = scan.nextInt();
System.out.print("Enter a floating point number: ");
what = scan.nextDouble() ;
three = 4 * one + 5 * two;
two = 2 * one;
System.out.println ("one " + two + ":" + three);

one = 46 / 5 * 2 + 19 % 4;
three = one + two;
what = (what + 2.5) / 2 ;
System.out.println (what + " is what!");

}
}
```

b) Open a java file called Trace. Copy the program in the file. Put comment to each row of code and explain what it is represent.

Area and Circumference of a Circle

1. Study the program below, which uses both variables and CONSTANTS:

- The first three lines inside *main* are declarations for PI, radius, and area. Note that the type for each is given in these lines: *final double* for PI, since it is a floating point constant; *int* for radius, since it is an integer variable, and *double* for area, since it will hold the product of the radius and PI, resulting in a floating point value.
- These first three lines also hold initializations for PI, radius, and area. These could have been done separately, but it is often convenient to assign an initial value when a variable is declared.
- The next line is simply a print statement that shows the area for a circle of a given radius.
- The next line is an assignment statement, giving variable radius the value 20. Note that this is not a declaration, so the *int* that was in the previous radius line does not appear here. The same memory location that used to hold the value 10 now holds the value 20—we are not setting up a new memory location.
- Similar for the next line—no *double* because area was already declared.
- The final print statement prints the newly computed area of the circle with the new radius.

```
public class Circle
{
  public static void main(String[] args)
 {
   final double PI = 3.14159;

    int radius = 10;
    double area = PI * radius * radius;

   System.out.println("The area of a circle with radius " + radius +
                      " is " + area);

     radius = 20;
     area = PI * radius * radius;

   System.out.println("The area of a circle with radius " + radius +
                      " is " + area);

    }
}
```

2. Modify your program as follows:

- At the very top of the file, add the line

  import java.util.Scanner;

  This tells the compiler that you will be using methods from the Scanner class. In the main method create a Scanner object called ***scan*** to read from System.in.

  Scanner scan=new Scanner (System.in);

- **Instead of initializing** the radius in the declaration, **just declare it without giving it a value**. Now add two statements to read in the radius from the user:

  radius = scan.nextInt(); // Den här raden använder scan objektet (förbindelsen scan) och sitt metod nextInt() som " transporterar" värdet från tangentbordet ditt program

- Compile and run your program.

Painting a Room

File *Paint.java* contains the partial program below, which when complete will calculate the amount of paint needed to paint the walls of a room of the given length and width. It assumes that the paint covers 95 square meter per liter

```
//*************************************************************
//File: Paint.java
//
//Purpose: Determine how much paint is needed to paint the walls
//of a room given its length, width, and height
//*************************************************************
import java.util.Scanner;

public class Paint
{
    public static void main(String[] args)
    {
        final int COVERAGE = 12;  //paint covers 12 sq meter/liter
        //declare integers length, width, and height;
        //declare double totalSqM;
        //declare double paintNeeded;
             //declare and initialize Scanner object

        //Prompt for and read in the length of the room

        //Prompt for and read in the width of the room

        //Prompt for and read in the height of the room

        //Compute the total square meter to be painted--think
        //about the dimensions of each wall

        //Compute the amount of paint needed
```

```
        //Print the length, width, and height of the room and the
        //number of liters of paint needed.
    }
}
```

Save this file to your directory and do the following:

1. Fill in the missing statements (the comments tell you where to fill in) so that the program does what it is supposed to. Compile and run the program and correct any errors.
2. Suppose the room has doors and windows that don't need painting. Ask the user to enter the number of doors and number of windows in the room, and adjust the total square meter to be painted accordingly. Assume that each door is 2 square meter and each window is 1.5 square meter.

Del 2: Labbar

Gör övningarna innan du börjar med labbarna.

För varje uppgift nedan gäller att du gör följande:
1) Läser hela uppgiften. Försöker få dig en stor bild om vad uppgiften går ut på.
2) Planera uppgiften genom att identifiera
- indata till programmet
- utdata från programmet
- vilka beräkningar som krävs för att utifrån indata realisera utdata
Om du inte lyckats skriva java koden för uppgiften måste du visa handledaren en skriftligt planeringen av uppgiften.

BMI

Skriv ett program för att beräkna BMI för både män och kvinnor. Förenklad beräknas BMI som:
BMI= vikt (kg) / längd * längd (m).
Till exempel kommer BMI för en person 70 kg och 1,68 cm vara ca 25 (70 / 1,68 * 1,68).
Ditt program bör uppmana användaren att ange hans / hennes längd och hans / hennes vikt.
Programmet ska sedan beräkna och skriva ut BMI.
Den allmänna beskrivning av dit program skulle vara följande:

• Deklarera dina variabler (tänk på vilka variabler du behöver, du behöver mata in två bitar av information (vad?)
• Få indata från användaren, vikt och längd
• Beräkna BMI
• Skriv ut svaren

Kompilera och köra programmet.

Timme1

a) Skriv ett program som läser värden som representerar en gången tid i timmar, minuter och sekunder, och sedan skriver ut motsvarande totala antalet sekunder.
Alla tre värden skall läsas in och lagras i variabler av lämplig typ.
Därefter skall konverteringen till sekunder göras samt det totala antalet sekunder beräknas.
Resultatet skall skrivas ut med lämpligt led text.

Timme2 (öppna en ny java fil och en ny klass för punkt b)

b) Skapa en reviderad version av det tidigare programmet genom att vända beräkningen. Gör detta i en ny fil och klass. Kalla klassen Timeme2.
Det du läser in är ett värde representerar antalet sekunder. Därefter skall programmet beräkna och skriva ut motsvarande mängd timmar, minuter och sekunder.
Använd % operator, kolla i boken eller exempel.

Till exempel är 9999 sekunder 2 timmar, 46 minuter och 39 sekunder
Till exempel är 9999 sekunder 2 timmar, 46 minuter och 39 sekunder

Timme3 (öppna en ny fil och klass för punkt 3)

c) Ändra din kod så det blir professionellt.
Skriv följande metod signatur i din klass utanför main (se exemplen från föreläsningen)

```
public static void computeTimme( int seconds){

 // här skall koden från uppgift b skall läggas. Tänk på att variablerna där du lägger timmar, minuter

och sekunder skall deklareras i metoden och inte i main().
Kan du säga varför?
}
```

d) När metoden är färdig skall du ”kalla på” metoden *computeTimme* i main(). Alltså anropa metoden. Se mitt exempel från föreläsningen.

Calories

Öppna ett nytt java-fil, skapa en ny java-klass och namnge det Calories. Klassen skall alltid sparas i en fil med samma namn alltså Calories.java

Ett sätt att mäta mängden energi som förbrukas under träning är att använda metabola ekvivalenter (MET). Här är några MET-s för olika aktiviteter:

Löpning 10 MET-s
Basket 8 MET-s
Sova 1 MET-s

Antalet kalorier som bränns per minut kan uppskattas med formeln:
Kalorier / minut = 0,0175 * MET * vikt (i kg) .

a) Skriv ett program som beräknar det totala antalet kalorier som bränns för en 75 kg person som löper 30 minuter, spelar basket i 30 minuter och sedan sover i 6 timmar.
b) Nu kommer du att göra ditt program mer generellt. Ändra programmet så att användaren kan mata in till programmet sin vikt, och antalet minuter som hon/han spenderar för varje aktivitet. Antalet MET-s skall deklarerar i början av programmet som konstanter. Leta på internet och komplettera ditt program med andra MET-s- variabler.
Programmet skall slutligen skriva ut det totala antalet kalorier förbrukade.

TimeEstimation

Forensiskt arbete handlar bl.a om att inte förstöra eller förändra digital data som man arbetar med. Detta görs genom att först skapa diskavbildningar. Detta kan vara ett långt tråkigt jobb så det gäller att veta hur lång tid det tar.

Skriv ett program som kommer att uppmana användaren att mata in värdet
på mediets kapacitet samt läshastigheten som gäller för att kopiera
data till det nya lagringsmediet.
Därefter skall programmet beräkna tiden som krävs för avbildning med formel :
tiden = kapacitet / bandbredd
Ta reda på läshastigheter för Tx. USB, HDD, minneskort.
Tänkt på att användaren brukar tala om Gbyte när det gäller minneskapacitet
och Mbit/sec när det gäller hastighet.
Du måste konvertera indata så att du jobbar med samma måttenhet.

Skriv ut tiden med ledande text.

SwimmingPool

Börja med att öppna en ny java-fil och klass. Spara den som SwimmingPool.java.
Ditt program skall fråga användaren om att ange längd, bredd och djup för en swimmingpool. Därefter skall programmet beräkna:

a) Volymen av swimmingpoolen.
b) Hur många liter vatten som krävs för att fylla poolen.
c) Anta att vattenkranen ger en liter vatten/ sekund. Hur lång tid tar det att fylla poolen? Ange resultatet i timmar, minuter och sekunder. Kan du återanvända den kod som du har skrivit i uppgift Timme3?

Punkt d) ej obligatoriskt, men prova på. Om det känns svårt nu skall hu göra uppgiften till labb3.

d) Nu märker ni att ert program består av olika ”delar” som har olika funktion. Dessa funktioner kan användas i andra program om vi gör dessa funktioner generella. Se mitt exempel från föreläsningen där jag flyttar beräkning av hastigheten till en separat funktion (metod). Samma sak gör du i uppgift Time3.

Den första metoden som beräknar volymen kan se ut som nedan.

```
public static double volume ( double lenght, double witdh, double deep)
{
 double volume = length* width* deep;
 return volume;
}
```

Skriv färdig följande metoder :
public static double konvertVToLiters(double volume){ ??? }
public static double calculateTimmeToFill(double liters, int liter_ sec){???}
public static int calculatePrice(double lites, double kr_liter) {};

Ändra nu programmet **SwimmingPool** så att det anropar metoderna. Programmet gör i princip samma sak som tidigare men på ett mer professionellt sätt. Detta gör också att din kod kan återavändas.

Del 3: Ej obligatoriskt men roligt

Drawing Shapes

The following is a simple applet that draws a blue rectangle on a yellow background.

```
import javax.swing.JApplet;
import java.awt.*;

public class Shapes extends JApplet
{
    public void paint (Graphics page)
    {


        // Declare variables
        int x, y;    // x and y coordinates of upper left-
corner of each shape
        int width, height; // width and height of each shape

        // Set the background color
        setBackground (Color.yellow);

        // Set the color for the next shape to be drawn
        page.setColor (Color.blue);

        // Assign the corner point and width and height
        x = 200;
        y = 150;
        width = 100;
        height = 70;

        // Draw the rectangle
        page.fillRect(x, y, width, height);
    }
}
```

Study the code, noting the following:

- The program imports javax.swing.JApplet because this is an applet, and it imports java.awt.* because it uses graphics.
- There is no *main* method—instead there is a *paint* method. The paint method is automatically invoked when an applet is displayed, just as the main method is automatically invoked when an application is executed.
- Most of the methods that draw shapes (see the list in Figure 2.12) require parameters that specify the upper left-hand corner of the shape (using the coordinate system described in Section 2.9) and the width and height of the shape. You can see this in the calls to *fillRect*, which draws a rectangle filled with the current foreground color.

1. Compile Shapes.java, but don't run it—this is an applet, so it is run through a browser or a special programs.

2. Run the program through the jGrasp by using **" the strawberry button "** which is actually the **"run applet for current file button " .**

4. Now change the width to 200 and the height to 300. Save, recompile and run to see how this affects the rectangle.

5. Modify the program so that it draws four rectangles in all, as follows:
   - One rectangle should be entirely contained in another rectangle.
   - One rectangle should overlap one of the first two but not be entirely inside of it.
   - The fourth rectangle should not overlap any of the others.

6. One last touch to the program ... Change the colors for at least three of the shapes so the background and each of the three shapes are different colors (a list of colors is in Figure 2.10 of the text). Also change two of the *fillRect* methods to *fillOval* so the final program draws two rectangles and two ovals. Be sure that the overlap rules are still met.

Creating a Pie Chart

Write an applet that draws a pie chart showing the percentage of household income spent on various expenses. Use the percentages below:

Rent and Utilities	35%
Transportation	15%
Food	15%
Educational	25%
Miscellaneous	10%

Each section of the pie should be in a different color, of course. Label each section of the pie with the category it represents—the labels should appear outside the pie itself.
Tip! Use the method fillArc(), se the course book or java doc, class Graphics .