Chapter 2: Data and Expressions
Lab Exercises

Area and Circumference of a Circle

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

```
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);

    }
}
```

Some things to notice:

- 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.

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:

  A read statement that actually reads in the value. Since we are assuming that the radius is an integer, this will use the **nextInt()** method of the Scanner- object (called **scan** in your case)

  radius = scan.nextInt(); // Den här raden använder scan objektet (förbindelsen scan)
  och sitt metod nextInt() som " transporterar" värdet du skvev in 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.

Ideal Weight (BMI)

Write a program to compute the BMI for both males and females. According to one study, the BMI is counted as:
weight(kg) / length* length (m) . For example, the BMI for a person 70 kg and 1,68 cm will be about 25 (70/ 1.68*1.68) . Your program should ask the user to enter his/her height and his/her weight. It should then compute and print the BMI. The general outline of your main function would be as follows:

- Declare your variables (think about what variables you need—you need to input two pieces of information (what?), then you need some variables for your calculations (see the following steps)
- Get the input (height in meter) from the user
- Compute the BMI
- Print the answers

Plan your program, then type it in, compile and run it. Be sure it gives correct answers.

Calories

Open a new java file and class and name it Calories. Save in a file Calories.java

One way to measure the amount of energy that is expended during exercise is to use the metabolic equivalents (MET). Here are some METS for various activities:

Running	10 METS
Basketball	8 METS
Sleeping	1 MET

The number of calories burned per minute may estimate using the formula:
Calories/Minute= 0.0175*MET*weight in kg.

a) Write a programme that calculate and outputs the total number of calories burned for a 75 kg person who runs for 30 minutes, play basket for 30 minutes and then sleep for 6 hours.

b) Now you will make your programme more general. Change the programme so the user will input his/hers values for weight, different activities METS and sleep hours.

Time

a) Write an application that reads values representing a time duration in hours minutes and seconds, and then prints the equivalent total number of seconds.

b) Create a revised version of the previous program that reverse the computation. That is, read a value representing the number of seconds, than print the equivalent amount of time as a combination of hours, minutes and seconds. For example, 9999 seconds is 2 hours, 46 minutes and 39 seconds .

Volume

Start by open **a new java file and class**. Save it as Volume.java.
Your programme should ask the user to input the length, breadth and deep for a swimming pool.

Than the program should calculate the volume and how much liter water you need to fill the pool and how long time it take. How much das it cost?
Print out the results.

Drawing Shapes (Ej obligatoriskt)

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 (Ej obligatoriskt)

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 .