A **block** is a bundle of statements in a computer program that can include declarations and executable statements. A programming language is **block structured** if it (1) allows blocks to be created and (2) allows blocks to be nested within other blocks. Many modern programming languages, including Java, are block structured.

Java blocks are identified and delimited by curly braces ({}), and nesting can be carried on to several levels – statement blocks can be nested within method blocks; method can be nested within class blocks; and class blocks may be nested within other class blocks.

When a block is nested within another, we call the surrounding block the **outer block** and the block nested within it the **inner block**. By drawing a line around each block and omitting the details (such as code and declarations) you can clarify the hierarchical block structure of code. Such a simplified drawing is called a **contour map**.

**Example**

Connecting the delimiting braces of each block on this Java code leads to the contour map shown on the right. The two if statement blocks are inner to the **do-while** statement block but separate from each other. The **do-while** statement is outer to both if statement blocks.
Example
This application has four blocks labeled A, B, C and D. The contour map is shown to the right.

```java
import java.util.Scanner;
import java.text.DecimalFormat;
import static javax.swing.JOptionPane.*;

public class PayWithOverTime
{
    static DecimalFormat df = new DecimalFormat( " $#,###.00" );

    public static void main( String [] args )
    {
        // declare data
        String input, prompt, output;
        prompt = "Enter name, wage, hours separated by commas";
        // read until user cancels dialog
        while ( ( input = showInputDialog( prompt ) ) != null )
        {
            // user entered data so build scanner and set delimiter
            Scanner scan = new Scanner( input );
            scan.useDelimiter( "," );
            // scan worker's name, hourly wage and hours worked
            String name = scan.next( );
            double wage = scan.nextDouble( );
            double hours = scan.nextInt( );
            // calculate worker's regular pay
            double pay = hours * wage;
            if ( hours > 40 ) // if overtime
            {
                // add half pay for each overtime hour
                double extraPay = (hours - 40) * wage * 0.5;
                pay = pay + extraPay;
            }
            output = "Pay " + name + df.format( pay );
            showMessageDialog( null, output );
        }
    }
}
```
Example
This application has five blocks. In its contour, the four method blocks B, C, D and E are stacked on top of each other and surrounded by the class block A.

```java
import static javax.swing.JOptionPane.*;
import java.text.DecimalFormat;

public class CircleMethods {
    static DecimalFormat df = new DecimalFormat( "0.000" );

    public static void main( String [] args )
    {
        // declare data
        double radius, area;
        radius = inputRadius( );
        area = calcArea( radius );
        printResults( radius, area );
    }

    public static double inputRadius( )
    {
        String msg = "Enter circle's radius";
        return Double.parseDouble( showInputDialog( msg ) );
    }

    public static double calcArea( double radius )
    {
        return Math.PI * Math.pow( radius, 2 );
    }

    public static void printResults( double r, double a )
    {
        String out = "Circle with radius " + df.format( r )
                       + "\nhas area = " + df.format( a );
        showMessageDialog( null, out );
    }
}
```
Example

Application **DiceToss** has six blocks, including an *inner class* nested within the application class. The contour map at right pictures how blocks E and F are inner to D; B, C and D are inner to A; and

```java
public class DiceToss
{
    public static void main( String [] args )
    {
        new DiceToss(); // call app constructor

        public DiceToss()  // app constructor
        {
            Die die1 = new Die();  // build one die
            Die die2 = new Die();  // build another
            die1.roll(); // roll them
            die2.roll();
            int sum = die1.faceUp + die2.faceUp;
            System.out.println( "Roll is " + sum );

            // A Die object models a 6-sided die used for dice games.
            public class Die
            {
                public int faceUp;  // number shown facing up

                public Die()  // explicit constructor
                {
                    roll();
                }

                public void roll()  // roll the die
                {
                    // randomly select a number from 1 to 6
                    faceUp = (int)(Math.random( ) * 6 + 1);
                }
            }
        }
    }
}
```
Exercises

1. Mark each block in the following Java program. HINT: any curly brace ({} delimiters) is a block because it could potentially contain data declarations even if it doesn’t. Draw its contour map.

```java
public class DukeandEarl {
    public static void main( String [] args )
    {
        final double DUKE_FTP = 0.5;
        final double EARL_FTP = 0.5;
        String shooter, winner;
        // Duke and Earl flip a coin to see who shoots first
        shooter = ( Math.random( ) < 0.5 ) ? "Duke" : "Earl";
        winner = "nobody";
        while ( winner.equals( "nobody" ) )
        {
            System.out.print( shooter + " shoots and " );
            if ( shooter.equals( "Duke" ) )
            {
                System.out.println( "HITS!" );
                winner = "Duke";
            }
            else
            {
                System.out.println( "misses" );
                shooter = "Earl";
            }
        }
        System.out.println( winner + " WINS!!" );
    }
}
```