Branching with if

The if keyword performs a conditional test to evaluate an expression for a Boolean value. A statement following the expression will only be executed when the evaluation is **true**; otherwise the program proceeds onto subsequent code - pursuing the next "branch". The **if** statement syntax looks like this:

if (*test-expression*) *code-to-be-executed-when-true*;

The code to be executed can contain multiple statements if they are enclosed within curly brackets to form a "statement block":

1. Start a new program named "**If**" containing the standard main method

```
Class If
{
        Public static void main (String [] args)
        {
        }
}
```

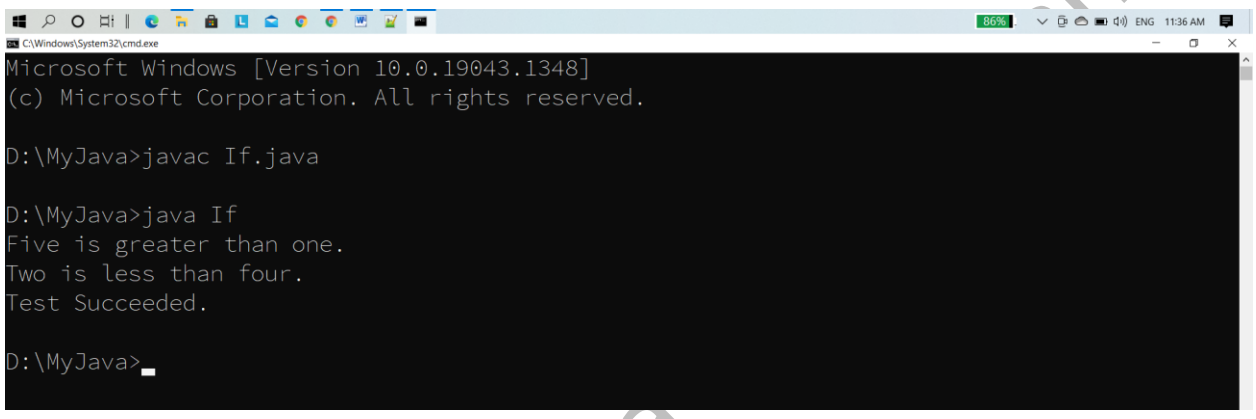
2. Between the curly braces of the main method, insert this simple conditional test the executes a single statement when one number is greater than another **if(5 > 1) System.out.println("Five is greater than one.");**

3. Add a second conditional test, which executes an entire statement block when one number is less than another

```
if ( 2 < 4 )
{
        System.out.println ("Two is less than four");
```

```
        System.out.println ("Test Succeeded");
    }
```

4. Save the program as **If.java** then compile and run the program to see all statements get executed – because both tests evaluate as **true** in this case:



A conditional test can also evaluate a complex expression to test multiple conditions foe a Boolean value. Parentheses enclose each test conditions to establish precedence – so they get evaluated first. The Boolean **&&** AND operator ensures the complex expression will only return **true** when both tested conditions are true:

if ((*test-condition-1*) **&&** (*test-condition-2*)) *execute-this-code*;

The Boolean || OR operator ensures a complex expression will only return **true** when either one of the tested conditions is true:

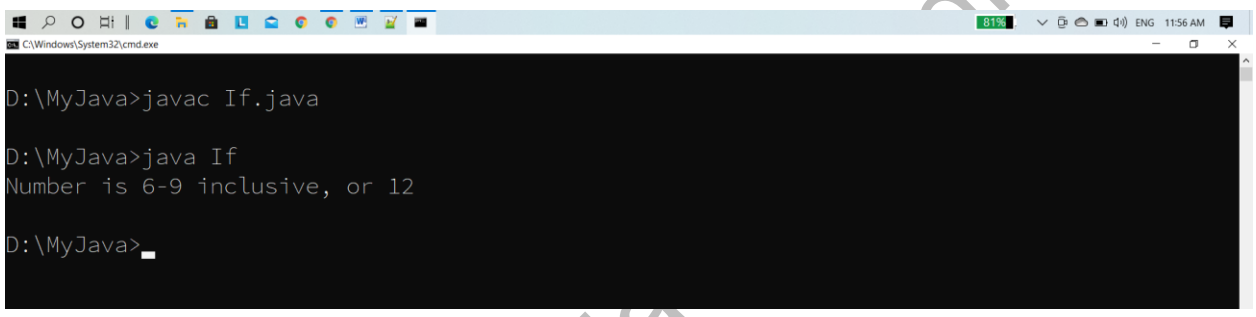
if ((*test-condition-1*) || (*test-condition-2*)) *execute-this-code*;

A combination of these can form longer complex expressions:

5. Inside the main method of **If.java** insert this line to declare and initialize an integer variable named **num int num = 8;**

6. Add a third conditional test that executes a statement when the value of the **num** variable is within a specified range, or when it's exactly equal to a specified range, or when it's exactly equal to a specified vale **if(((num > 5) && (num < 10)) || (num == 12)) System.out.println("Number is 6-9 inclusive, or 12")**;

7. Recompile the program and run it once more to see the statement after the complex expression get executed



8. Change the value assigned to the **num** variable so it is neither within the specified range 6-9, or exactly 12. Recompile the program and run it again to now see the statement after the complex expression is not executed.

Hot tip:

Expression can utilize the **true** and **false** keywords. The test expression **(2 < 4)** is shorthand for **(2 < 4 == true)**.

The range can be extended to include the upper and lower limits using the **>=** and **<=** operators.

Don't forget:

The complex expression uses the **==** equality operator to specify an exact match, not the = assignment operator.

Branching alternatives:

The else keyword is used in conjunction with the if keyword to create if else statements that provide alternative branches for a program to pursue – according to the evaluation of a tested expression. In its simplest form this merely nominates an alternative statement for execution when the test fails:

If (test-expression)

Code-to-be-executed-when-true;

else

Code-to-be-executed-when-false;

Each alternative branch may be a single statement or a statement block of multiple statements – enclosed within curly brackets.

More powerful **if else** statements can be constructed that evaluate a test expression for each alternative branch. These employ nested **if** statements after each **else** keyword to specify each further test. When the program discovers an expression that evaluates as **true** it executes the statement associated with just that test then exists the **if else** statement without exploring any further branches:

1. Start a new program named "Else" containing the standard main method

```
Class Else
{
        Public static void main (String [] args)
        {
        }
}
```

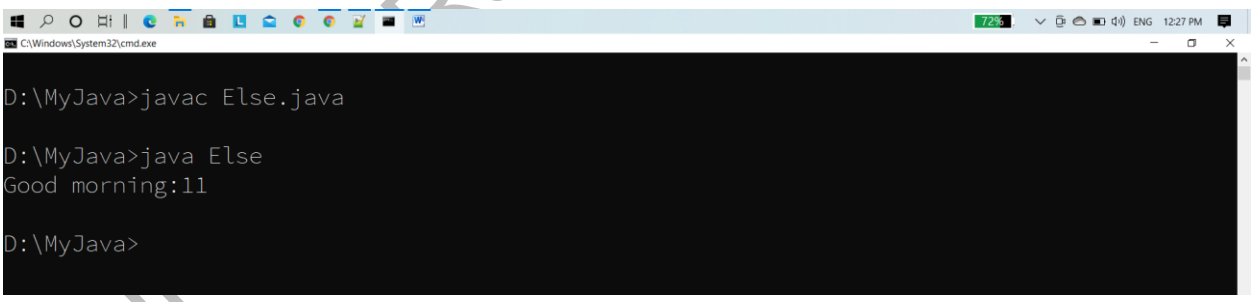
2. Inside the main method, insert this line to declare and initialize an integer variable named **hrs**

   int hrs = 11;

3. Insert the simple conditional test, which executes a single statement when the value of the **hrs** variable is below 13

```
if ( hrs < 13 )
{
        System.out.println("Good morning:" + hrs);
}
```

4. Save the program as **Else.java** the compile and run the program to see the statement get executed

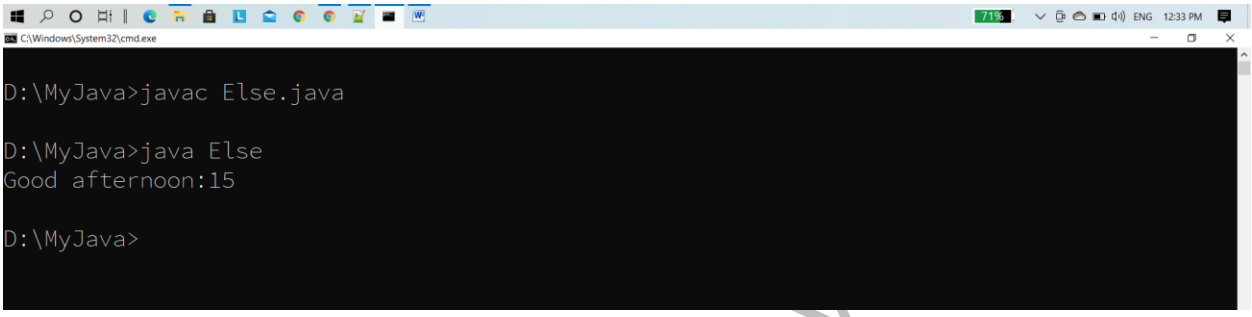


5. Change the value assigned to the **hrs** variable to 15 then add this alternative branch right after the **if** statement

```
else if ( hrs < 18 )
{
        System.out.println("Good afternoon:" + hrs );
```

}

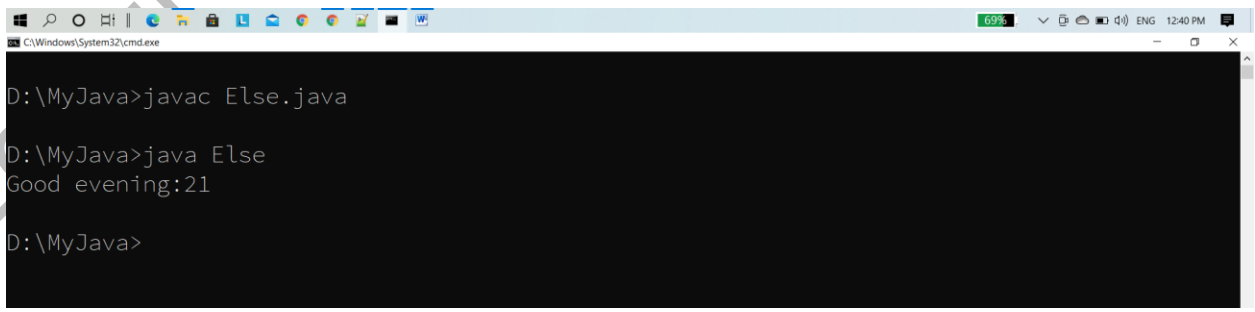
6. Save the changes, recompile, and run the program again to see just the alternative statement get executed



It is sometimes desirable to provide a final **else** branch, without a nested **if** statement, to specify a "default" statement to be executed when no tested expression evaluates as **true:**

7. Change the value assigned to the **hrs** variable to 21, then add this default branch to the end of the **if else** statement **else System.out.println("Good evening:" +hrs);**

8. Save the changes, recompile, and run the program once more to see just the default statement get executed



Beware:

Notice that the first statement is terminated with a semicolon, as usual, before the else keyword.

Don't forget:

Conditional branching is the fundamental process by which computer programs proceed.

Now let us see the programs discussed in the video:

If statement:

Example: 1

```
class IfDemo
{
    public static void main(String[] args)
    {
        if(18 > 20)
        {
            System.out.println("20 is greater than 18");
        }
    }
}
```

Output:

```
D:\MyJava>javac IfDemo.java

D:\MyJava>java IfDemo

D:\MyJava>
```

Example: 2

```java
class IfDemo
{
    public static void main(String[] args)
    {
        if(20 > 18)
        {
            System.out.println("20 is greater than 18");
        }
    }
}
```

Output:

```
D:\MyJava>javac IfDemo.java

D:\MyJava>java IfDemo

D:\MyJava>javac IfDemo.java

D:\MyJava>java IfDemo
20 is greater than 18

D:\MyJava>
```

If Else:

Example: 1

```java
class IfElse
{
    public static void main(String[] args)
    {
        int x = 20;
        int y = 18;

        if(y > x)
        {
            System.out.println("Y is greater than X");
        }
        else
        {
            System.out.println("X is greater than Y");
        }
    }
}
```

Output:

```
D:\MyJava>javac IfElse.java

D:\MyJava>java IfElse
X is greater than Y

D:\MyJava>
```

Else If:

Example: 1

```java
class ElseIf
{
    public static void main(String[] args)
    {
        int marks = 95;
        if(marks < 50)
        {
            System.out.println("Fail");
        }
        else if(marks>=50 && marks<60)
        {
            System.out.println("D grade");
        }
        else if(marks>=60 && marks<75)
        {
            System.out.println("C grade");
        }
        else if(marks>=75 && marks<90)
        {
            System.out.println("B grade");
        }
        else if(marks>=90 && marks<100)
        {
            System.out.println("A+ grade");
        }
        else
        {
            System.out.println("Invalid");
        }
    }
}
```

Output:

Lectures by Ravula Govardhan (SUBSCRIBE & Click BELL)

If you found our videos are helpful, please #Subscribe and hit the #Bell. Share our videos with your friends. Happy Learning.