JPL :: Branching

TalentSprint

Licensed To Skill

Version 1.0.4

Learning Objectives

By the end of this presentation, you will be able to:

- Understand IF, IF-ELSE and ELSE-IF
- Design solutions that need conditional execution of statements
- Use logical operators for evaluating more than one condition together for conditional execution
- Use intermediate variables for efficient coding
- Understand Nested-IF

Branching

1. Go to the Bus stop
2. Get into the Bus
3. Get down the Bus at nearest Bus stop to home
4. Get back to home

Branching

1. Go to the Bus stop
2. Get into the Bus
3. Get down the Bus at nearest Bus stop to home

   If you have to buy fruits

   4. Go to the Fruit market to get fruits

5. Get back to home

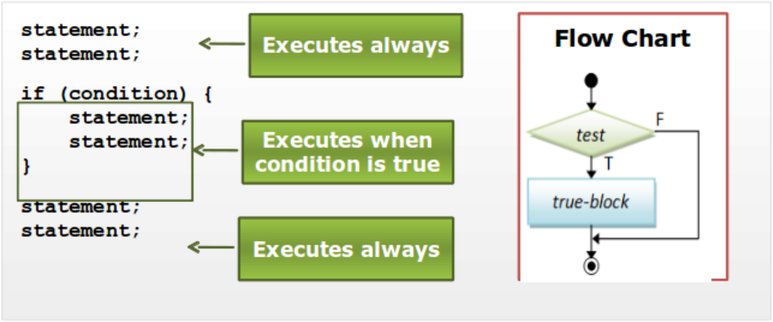
Branching

- A conditional statement is an expression that produces a true or false result. Based on the result, some actions are performed.
- Relational operators are used for conditions. Actions are blocks of statements.

Branching

```
public class DisplayAbsolute {
    public static void main(String[] args) {
        int number = Integer.parseInt (args[0]);
        if  (number < 0) {
            number = −number;
        }
        System.out.println("Number = " + number);
    }
}
```

Branching



```
statement;
statement;

if (condition) {
    statement;
    statement;
}
statement;
statement;
```

Branching

22323020

```
Read the given number and call it X.
Divide X by 2 and record the remainder.
If remainder is 0, print "X is even".
Otherwise, print "X is odd".
```

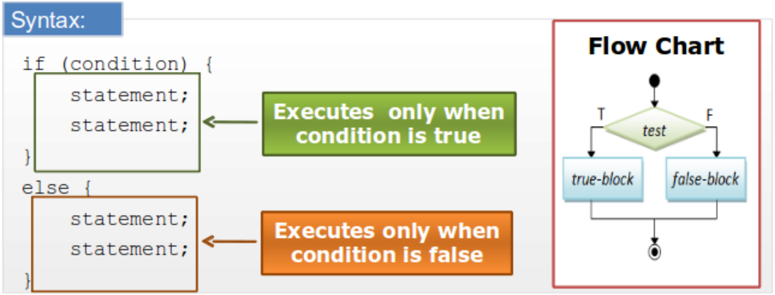
Branching

```
public class EvenOdd {
    public static void main(String[] args) {
        int givenNumber = Integer.parseInt(args[0]);
        if  (givenNumber % 2 == 0) {
            System.out.println("Number is even.");
        }  else {
            System.out.println("Number is odd.");
        }
    }
}
```

Branching

if-else Statement

The **if–else** statement is a control flow statement that tells the program to execute a certain section of code. It depends on condition. If condition evaluates to true **if** block is executed otherwise **else** block.



Syntax:

```
if (condition) {
    statement;
    statement;
}
else {
    statement;
    statement;
}
```

Executes only when condition is true
Executes only when condition is false
Flow Chart
T
F
test
true-block
false-block

Branching

A Few More Operators

Arithmetic Operators %

Relational Operators < > <= >= == !=

Can you name these operators and operations they perform on operands?

Branching

Operator Description

operator	description
==	Tests whether the expressions on the left and right are equivalent
<=	Tests whether the expression on left less than or equal to the expression on right
<	Tests whether the expression on left less than the expression on right
>	Tests whether the expression on left greater than the expression on right
>=	Tests whether the expression on left greater than or equal to the expression on right
!=	Tests whether the expressions on the left and right are not equal

Branching

Be sure to distinguish between the relational operator == and the assignment operator =

X == Y	Tests, if the contents of variable x are the same as the contents of variable y.
X = Y	Assigns the value stored in variable y to variable x (overwriting what is in x and leaving y unchanged)

Branching

X == Y

```
9 == 8
a == b
a == 6
5+6 == 3+2
5 == a
```

X = Y

```
a = 8
a = b
a = 3+2
5 = 4 // Invalid
```

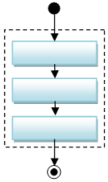
Note

When an operator has two characters (e.g. ==, <=, >=), then there should be no space between them.

Branching

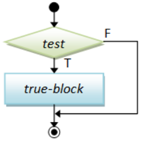
Sequential and Branching statements

Sequential Execution

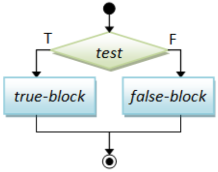


Sequential

Branching Execution



If Statement



If-else Statement

Branching

Another problem involving Conditional Processing

4, 9

$\uparrow$

Which is the largest number?

Solution

- Take the two numbers (n1, n2)
- If n1 is greater than n2, then n1 is largest.
- Otherwise, n2 is largest.

Branching

Let's Structure the Solution

```
n1 = First Number
n2 = Second Number
if (n1 > n2) print("First number is larger")
Otherwise, print("Second number is larger")
```

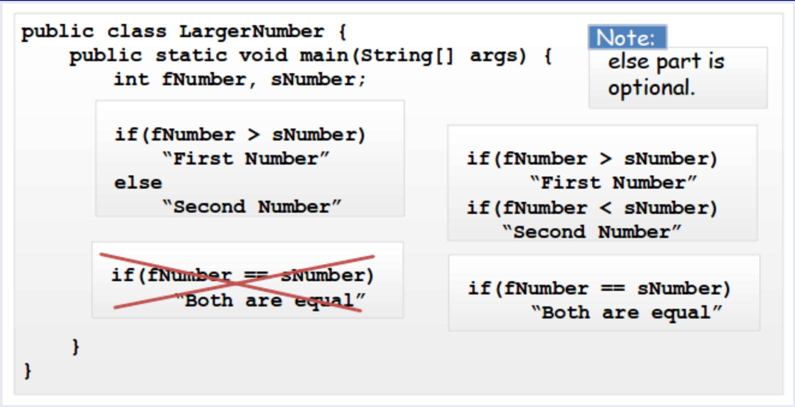
Branching

Program to find largest among two numbers

```
public class LargerNumber {
    public static void main(String[] args) {
        int fNumber, sNumber;
          .........
          .........
          .........
    }
}
```

Branching

Program to find largest among two numbers



```
public class LargerNumber {
    public static void main(String[] args) {
        int fNumber, sNumber;
```

Note:
else part is optional.

```
        if(fNumber > sNumber)
            "First Number"
        else
            "Second Number"
```

```
        if(fNumber > sNumber)
            "First Number"
        if(fNumber < sNumber)
            "Second Number"
```

~~if(fNumber == sNumber)~~
~~"Both are equal"~~

```
        if(fNumber == sNumber)
            "Both are equal"
```

```
    }
}
```

What if *fNumber* is equal to *sNumber*?

Branching

What are the problems with this code?

```
public class LargerNumber {
    public static void main(String[] args) {
        int fNumber, sNumber;
        if  (fNumber > sNumber) {
            "First  Number"
        }
        else if(fNumber < sNumber) {
            "Second Number"
        }  else {
            "Both are equal"
        }
    }
}
```

Branching

First Problem

Second and third conditions are evaluated even if first condition is true

Second Problem

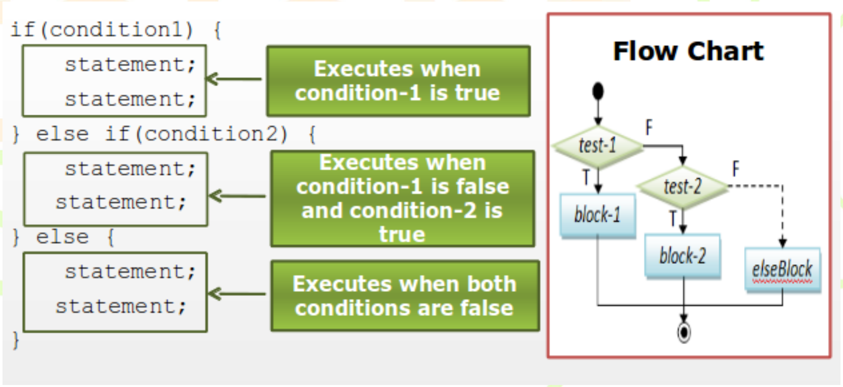
If the first two conditions fail then no need test the third condition at all.

Is there any better way?

if...
else if...
else...

Branching

Syntax of **if** – **else** – **if**



```
if(condition1) {
      statement;
      statement;
} else if(condition2) {
      statement;
     statement;
} else {
      statement;
     statement;
}
```

Executes when condition-1 is true

Executes when condition-1 is false and condition-2 is true

Executes when both conditions are false

Branching

Write Java code to find largest among four numbers. The class name and basic structure is given below:

```
public class LargestNumber {
    public static void main(String[] args) {
        int next, largestSoFar;
         .........
         .........
         .........
    }
}
```

Branching

Write a Java program to find the grade of a student (given the marks) using the following rules. Print “Marks: Grade” 91-100: A grade; 81-90: B grade; 71-80: C grade; 61-70: D grade; 51-60: E grade; < 51: Fail

```
public class StudentGrade {
    public static void main(String[] args) {
        int marks = Integer.parseInt(args[0]);
          .........
          .........
          .........
    }
}
```

Branching

Given two numbers, print ‘true’ if they are equal; ‘false’ otherwise.

```
public class TwoNumsEqual {
    public static void main(String[] args) {
        ....
        ....
        if (num1 == num2)
            System.out.print ("true");
        else
            System.out.print ("false");
    }
}
```

Branching

Add Elegance to the Code
Can we write the code more elegantly?

Try this

```
public class TwoNumsEqual {
    public static void main(String[] args) {
        ....
        System.out.print(num1 == num2);
    }
}
```

Branching

Nested If

Let's see how we can write a solution to find the smallest number among three numbers:

```
if (n1 < n2)
    if (n1 < n3)
        n1 is smallest
    else
        n3 is smallest
else  // n2 is smaller than n1
    if (n2 < n3)
        n2 is smallest
    else
        n3 smallest
```

```
if ((n1 < n2) && (n1 < n3))
    n1 is smallest
else if (n2 < n3)
    n2 is smallest
else
    n3 is smallest
```

Branching

Now, if we want to find largest among four numbers:

4, 9, 2, -27

$\uparrow$

Which is the largest number?

So, how many levels of nested condition do we have to write?
Or, how many conditions do we have to logically combine?

Branching

```
n1 = Integer.parseInt(args[0]);
n2 = Integer.parseInt(args[1]);
n3 = Integer.parseInt(args[2]);
n4 = Integer.parseInt(args[3]);
if ((n1 > n2) && (n1 > n3) && (n1 > n4))
    n1 is largest
else if ((n2 > n3) && (n2 > n4))
    n2 is largest
else if ((n3 > n4))
    n3 is largest
else
    n4 is largest
```

Branching

Solution ? Use an intermediate variable.

```
n1 = First Number
n2 = Second Number
if (n1 > n2) largestSoFar = n1; otherwise, largestSoFar = n2;
n3 = Third Number
if (n3 > largestSoFar) largestSoFar = n3
n4 = Fourth Number
if (n4 > largestSoFar) largestSoFar = n4
Print largestSoFar
```

Branching

We can do even better. Because we need only one of `n1, n2, n3` and `n4` at anytime, we can use only one variable for all the four numbers, holding them one at a time.

```
largestSoFar = First Number
next = Second Number
if (next > largestSoFar) largestSoFar = next
```

Branching

```
next = Third Number
if (next > largestSoFar) largestSoFar = next
next = Fourth Number
if (next > largestSoFar) largestSoFar = next
Print largestSoFar
```

Branching

Use an intermediate variable called grade and see how it helps.

```
public class StudentGrade {
    public static void main(String[] args) {
        char grade;
         .........
         .........
         .........
    }
}
```

Branching