upGrad

Full-Stack Software Development

upGrad

Lecture on: Operators and Conditionals

Instructor: Rajesh Kumar

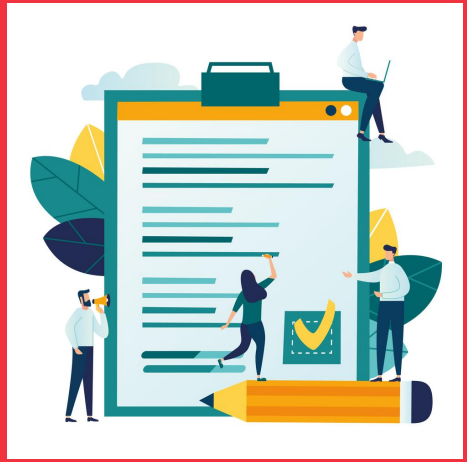
upGrad

Course Roadmap

- Introduction to Java
- **Operators and Conditionals**
- Loops and Methods
- Strings and Arrays

Yesterday's Agenda

1. Introduction to Java
2. Java Development Tools
3. Advantages of Java
4. Main Method
5. Variables in Java
6. Printing Values



Today's Agenda

1. Operators and Their Types
2. Expressions and Statements
3. Operator Precedence and Associativity
4. Conditionals in Java
5. Ternary Operator

Yesterday's Homework

upGrad

You are designing an asset tracking system that allows the user to enter the following details of an asset:

1. Asset tracking number: It is an integer used to identify the asset.
2. Asset name: It is the name of the asset.
3. Asset value: It denotes the price of the asset. An asset value can be 980.75.

Write a program that takes the aforementioned inputs from the user and prints them on the console.

Input Format:
a.The first line contains the asset tracking number, which is an integer
b. The second line contains the asset name, which is a string
c. The third line contains the asset value, which is a float number

Output Format:
a. The first line contains the asset tracking number input by the user
b. The second line contains the asset name input by the user
c. The third line contains the asset value input by the user

Yesterday's Homework Solution

upGrad

```
package com.company;
import java.util.Scanner;
public class Main {

   public static void main(String args[]) {
     Scanner sc = new Scanner(System.in);
     int assetTrackingNumber = sc.nextInt();
     String assetName = sc.next();
     float assetValue = sc.nextFloat();
     System.out.println(assetTrackingNumber);
     System.out.println(assetName);
     System.out.println(assetValue);
    }
}
```

What Is an Operator?

upGrad

You are aware of the variables and the the various ways in which they can be initialised. However, only declaring and initialising the variables is not enough. How do you play around with these variables or get *something* done?

- An operator is a symbol that performs an operation on a variable(s).
- Operators are used to perform operations such as addition, subtraction and value inversion.
- “+”, “*” and “!” are some examples of operators.

Types of Operators

upGrad

Java offers the following types of operators:

a. Arithmetic Operators

b. Assignment Operators

c. Unary Operators

d. Equality and Relational Operators

e. Logical Operators

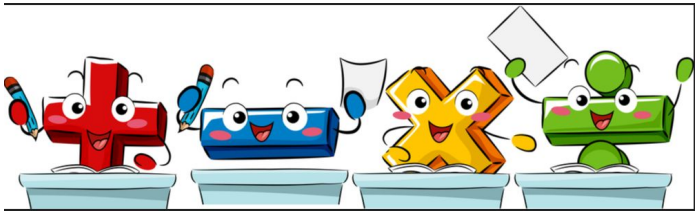
f. Bitwise Operators

Arithmetic Operators

upGrad

In your daily life, you perform various arithmetic operations such as calculating the total cost of the products that you buy from a grocery store.

- In Java, arithmetic operators are used for addition, subtraction, multiplication and division.
- We use +, -, *, / and % for addition, subtraction, multiplication, division and remainder, respectively.
- In case of multiplication, general **BODMAS** rules are followed.



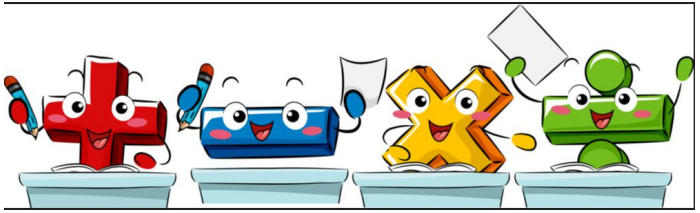
Note:

- An operation performed on variables of int data type will always return an **int** value and hence can lead to information loss.
- You should always use appropriate data types in the first place depending on your purpose to avoid loss of information.

Arithmetic Operators

upGrad

```
public class Main {

  public static void main(String[] args)
  {
      int a = 10;
      double b = 15.45;

      //Addition
     System.out.println(a+b); //25.45

     //Subtraction
     System.out.println(a-b); //-5.449999999999999

     //Multiplication
     System.out.println(a*b); //154.5

     //Division
     System.out.println(b/a); //1.545

     //Remainder
     System.out.println(b%a); //5.449999999999999
  }
     }
```



Assignment Operator

upGrad

- The most common operator that you will use is the assignment operator.
- The assignment operator “=” is used to assign a value to an operand.
- It can also be used to assign *object references**.

**You will learn about this in detail in the course on Object-Oriented Analysis and Design.*

```
public class Main {

  public static void main(String[] args)
  {
      int a = 10;
      double b = 15.45;
      int c = a+b; //25
      String message = "Hello There!";
  }
     }
```

Unary Operators

upGrad

- Unary operators are the operators that require only one operand.
- They perform operations such as increment, decrement and value inversion.

Let's take a look at each unary operator with an example.

1. **Unary Positive:** It indicates that a number is positive. **Note:** A number is positive regardless of whether the + symbol is present or not.

```
//+ Operator
int a = +1;
System.out.println(a);
//1
```

Operator	Description
+	Indicates that a number is positive
-	Indicates that a number is negative
++	Increments a value by 1
--	Decrements a value by 1
!	Inverts the value of a boolean

Unary Operators

Unary Operators

upGrad

2. Unary Negative: It indicates that a number is negative. It converts a negative number to a positive one.

```
//- Operator
int a = -1;
System.out.println(a); //-1
```

3. Increment: It increases the value of a number by 1. It can be used in the following two ways:

a. Pre-increment: It is placed before the operand and increments the value instantly.

```
//Pre-Increment
int a = 1;
++a;
System.out.println(a); //2
```

b. Post-increment: It is placed after the operand, and the value of the operand is increased instantly, but the previous value is retained until the execution of the statements.

```
//Post-Increment
int b= -1;
b++;
System.out.println(b); //0
```

Unary Operators

upGrad

4. Decrement: It decreases the value of a number by 1. It can be used in the following two ways:

a. Pre-decrement: It is placed before the operand and decrements the value instantly.

```
//Pre-Decrement
int a = 1;
--a;
System.out.println(a); //0
```

b. Post-decrement: It is placed after the operand, and the value of the operand is decreased instantly, but the previous value is retained until the execution of the statements.

```
//Post-Decrement
int b= -1;
b--;
System.out.println(b); //-2
```

5. Inversion: It is used to invert the value of a boolean.

```
//Value inversion
Boolean value = true;
System.out.println(!value); //false
```

Relational Operators

upGrad

- Relational operators are used to determine the relation between two operands such as greater than, less than or equal to.
- The two operands are compared, and the result returns a boolean value.
- There are six types of relational operators as mentioned below.

1. **Equal-to (==) operator:** The equal-to operator is used to determine whether or not two numbers are equal to each other.

```
//Equality Operator
int a = 1;
int b = 2;
System.out.println(a==b); //false
```

Note: You must use "==" and not "=" when testing whether or not two primitive values are equal.

Relational Operators

upGrad

2. Not-equal-to (!=) operator: The not-equal-to operator is used to determine whether two numbers are not equal to each other.

```
//Not Equal to Operator
int a = 1;
int b = 2;
System.out.println(a!=b); //true
```

3. Greater than (>) operator: The greater than operator is used to determine whether or not one operand is greater than another one.

```
//Greater than Operator
int a = 1;
int b = 1;
System.out.println(a>b); //false
```

Relational Operators

upGrad

4. Greater than or equal to (>=) operator: The greater than operator is used to determine whether or not one operand is greater than another one.

```
//Greater than or equal to Operator
int a = 1;
int b = 1;
System.out.println(a>=b); //true
```

5. Less than (<) operator: The less than operator is used to determine whether or not one operand is less than another one.

```
//Less than Operator
int a = 1;
int b = 1;
System.out.println(a<b); //false
```

Relational Operators

upGrad

6. Less than or equal to (<=) operator: The less than or equal to operator is used to determine whether or not one operand is less than or equal to another one.

```
//Less than or equal to Operator
int a = 1;
int b = 1;
System.out.println(a<=b); //true
```

Logical Operators

upGrad

- Logical operators take boolean data type as input and give an output of the same data type.
- These operators help in condensing a condition into fewer lines, which simplifies your code.
- They are used extensively for decision-making.

Now, let's take a look at the logical operators in Java, which are as follows:

1. **Logical AND (&&) Operator**: The AND operator returns true only if both the conditions are true. Even if one condition is false, the operator will return false.

   Syntax: *condition1* && *condition2*

2. **Logical OR (||) Operator:** The OR operator returns true if any one of the conditions is true. Even if one condition is false, the operator will return true.

   Syntax: *condition1* || *condition2*

3. **Logical NOT(!) Operator:** As you learnt in the section on unary operators, the NOT operator returns true if the given condition is not true.

   Syntax: !(*condition)*

Logical Operators

upGrad

```
public class Main {

  public static void main(String[] args) {
     int a = 10;
     int b = 12;
     int c = 30;
     //AND Operator
     System.out.println(a<b && c<b); //false
     //OR Operator
     System.out.println(a<b || c<b); //true
     //NOT Operator
     System.out.println(!(a<b)); //false
  }
}
```

Bitwise Operators

upGrad

- Bitwise operators are used to perform operations on the individual bits of a number.
- They can be used with various data types such as int, char and byte.
- They are not used frequently in coding.

Let's take a look at the bitwise operators in Java, which are as follows:

1. **Bitwise AND (&) Operator**: The bitwise AND operator returns 1 only if both the bits are 1. Otherwise, it returns 0.
2. **Bitwise OR (|) Operator:** The bitwise OR operator returns 1 if any one of the bits is 1. Otherwise, it returns 0.
3. **Bitwise XOR (^) Operator:** The bitwise XOR operator returns 1 if any both bits are same (0 or 1). Otherwise, it returns 0.
4. **Bitwise Complement (~) Operator:** The bitwise complement operator returns the reversed value of an input bit. If the input bit is 0, then it will return 1, and if it is 1, then it will return 0.

Bitwise Operators

upGrad

x	y	x && y	x || y
true	true	true	true
true	false	false	true
false	true	false	true
false	false	false	false

Bitwise Operators

upGrad

```
public class Main {

  public static void main(String[] args) {
     int a = 10;
     int b = 12;
     int c = 30;
     //Bitwise AND Operator
     System.out.println(a & b); //8
     //Bitwise OR Operator
     System.out.println(a | b); //14
     //Bitwise XOR Operator
     System.out.println(a ^ b); //6
     //Complement Operator
     System.out.println(~c); //-31
  }
}
```

Poll 1 (15 sec.)

What will be the output of the following code?

```
int a = 2;
int b = 2;
System.out.println(a=b);
```

1. True
2. 2
3. Compilation error
4. None of the above

Poll 1 (Answer)

What will be the output of the following code?

```
int a = 2;
int b = 2;
System.out.println(a=b);
```

1. True
2. **2**
3. Compilation error
4. None of the above

Poll 2 (15 sec.)

What will be the output of the following code?

```
int a = 10;
int b = ~a;
System.out.println(b);
```

1. 01
2. -11
3. 11
4. 09

Poll 2 (Answer)

What will be the output of the following code?

```
int a = 10;
int b = ~a;
System.out.println(b);
```

1. 01
2. **-11**
3. 11
4. 09

Poll 3 (30 sec.)

What will be the output of the following code?

```
int a = -12;
int b = 2;
System.out.println((a/b)>
0 && (b>0));
```

1. True
2. False

Poll 3 (Answer)

What will be the output of the following code?

```
int a = -12;
int b = 2;
System.out.println((a/b)>0 && (b>0));
```

1. True
2. **False**

Hands-On Exercise 1

upGrad

Take two integers as input and perform the following operations on them:

1. Find their product.
2. Increment the first number by 1.
3. Find out the minimum value.
4. Perform the AND operation between them.

Operator Precedence and Associativity

upGrad

- Precedence and associative rules are used when dealing with equations involving more than one type of operator.
- These rules determine which part of the equation is to be considered first, as there can be many different valuations for the same equation.
- Operator precedence determines which operator should be evaluated first when there are multiple operators.
- In case there are two operators of the same precedence, they are solved according to the associativity rule, i.e., either from right to left or from left to right.
- The table in the next slide depicts the precedence of operators in decreasing order, as the magnitude on the top represents the highest precedence and the one at the bottom shows the lowest precedence.

Operator Precedence and Associativity

upGrad

Operators	Precedence	Associativity
postfix	`expr++ expr--`	Right to left
unary	`++expr --expr +expr -expr ~ !`	Right to left
multiplicative	`* / %`	Left to Right
additive	`+ -`	Left to Right
shift	`<< >> >>>`	Left to Right
relational	`< > <= >= instanceof`	Left to Right
equality	`== !=`	Left to Right
bitwise AND	`&`	Left to Right
bitwise exclusive OR	`^`	Left to Right
bitwise inclusive OR	`|`	Left to Right
logical AND	`&&`	Left to Right
logical OR	`||`	Left to Right
ternary	`? :`	Right to left
assignment	`= += -= *= /= %= &= ^= |= <<= >>= >>>=`	Right to left

Poll 4 (30 sec.)

What will be the value of ‘e’ in the following code?

```
int a = 2;
int b = 3;
int c = 4;
double e = ++b*c/a;
```

1. 8.0
2. 7.0
3. 3.25
4. 6.0

Poll 4 (Answer)

What will be the value of ‘e’ in the following code?

```
int a = 2;
int b = 3;
int c = 4;
double e = ++b*c/a;
```

1. **8.0**
2. 7.0
3. 3.25
4. 6.0

Poll 5 (30 sec.)

What will be the value of ‘e’ in the following code?

```
int a = 2;
int b = 3;
int c = 4;
int e = a^b*c-a;
```

1. 2
2. 8
3. 1
4. None of the above

Poll 5 (30 sec.)

What will be the value of ‘e’ in the following code?

```
int a = 2;
int b = 3;
int c = 4;
int e = a^b*c-a;
```

1. 2
2. **8**
3. 1
4. None of the above

Expressions

upGrad

- An *expression* is a construct made up of variables, operators and method invocations that evaluates to a single value.
- An expression returns a single value, and the data type of the value returned by an expression depends on the elements used in the expression.

```
//Expressions
int a = 10;
int b = 12;
int c;
c = a +b;
```

- In the code given above, ‘a + b’ is an expression.

Statements

upGrad

- A statement forms a complete unit of execution.
- You can consider statements as equivalents of English statements.
- A statement in Java can be of the following two types:
  - **Declarative Statements:** They are used for declaring variables.

```
int a = 10;
```

  - **Expression Statements:** They are expressions terminated by a semicolon (;).

```
++a;
a--;
```

Flow Control Statements

upGrad

Whenever you have to go out, how do you decide which outfit to wear? Let's take a look at the following general factors that you may usually consider:

1. Temperature outside
2. The place that you are going to visit
3. Your style
4. Occasion

Your decision is governed by these factors that help you in arriving at a conclusion. Similarly, in programming, we have to make some decisions such as whether a number is even or odd and whether a number is divisible by 2.

We use flow control statements to control the execution of a program based on some predefined conditions.

Java supports the following types of flow control statements:

1. If
2. Else
3. Switch

If Statement

upGrad

- An if statement is the simplest decision-making statement.
- It is used to determine whether or not a statement or a group of statements is executed.
- There is a condition defined in the if statement, and if that condition is true, then the block is executed; otherwise, it is not executed.

Syntax:

```
if(condition) {
  //statement(s) to be executed when the condition is met
}
```

Note:

- The condition should always return a boolean value.
- If the {} are not provided, then only the statement that is below the if statement would be executed.

Poll 6 (30 sec.)

Which line in the following code is a statement?

```
float a; //1
a = 12.5f; //2
double b = a; //3
```

1. 1
2. 2
3. 3
4. There is no statement in the code

Poll 6 (Answer)

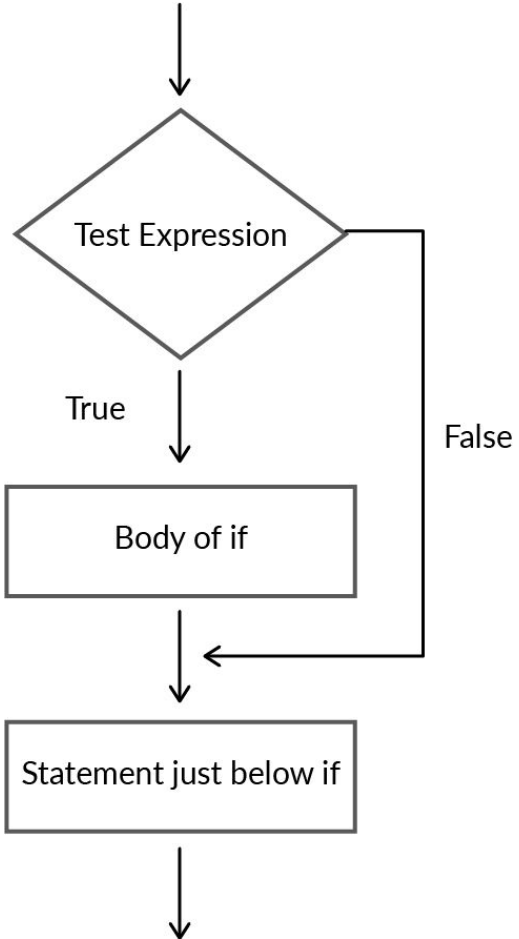
Which line in the following code is a statement?

```
float a; //1
a = 12.5f; //2
double b = a; //3
```

1. 1
2. 2
3. **3**
4. There is no statement in the code

If Statement: Execution

upGrad



if Statement

upGrad

```
package com.company;
import java.util.Scanner;

public class Main {

  public static void main(String[] args) {
     //if Statement
     int a = 10;
     Scanner sc = new Scanner(System.in);
     int b = sc.nextInt();
     if(a<b)
        System.out.println("b is greater");
  }
}
```

If-Else Statement

upGrad

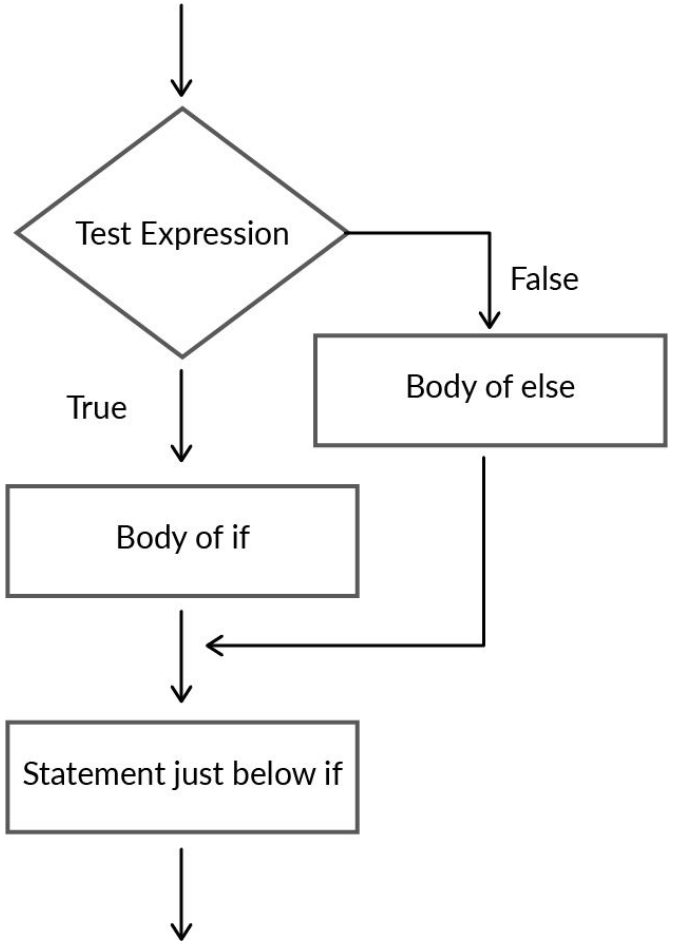
- The if statement has only one condition, which, if true, executes some statements; otherwise, it does not.
- There are cases when we need to execute something else in case the if condition is not satisfied.
- The else statement is used to specify what is to be done in case the if condition is not satisfied.

Syntax:

```
if(condition){
  //statement(s) to be executed when the condition is met
} else {
   //statement(s) to be executed when the if condition is not met.
}
```

If-Else Statement: Execution

upGrad



If-Else Statement

upGrad

```
public class Main {

  public static void main(String[] args) {
     //if - else  statements
     int a = 2;
     Scanner sc = new Scanner(System.in);
     int b = sc.nextInt();
     if(a<b)
        System.out.println("b is greater");
     else
        System.out.println("a is greater");
  }
}
```

Hands-On Exercise 2

upGrad

Take a number as an input from the user and find out whether the number is even or odd.

Else-If Statement

upGrad

So far, we have seen cases with only two options, i.e., either the if condition gets satisfied or the else statement is executed.

However, there could be cases where we have more than two options. For example, whether a number is positive, zero or negative.

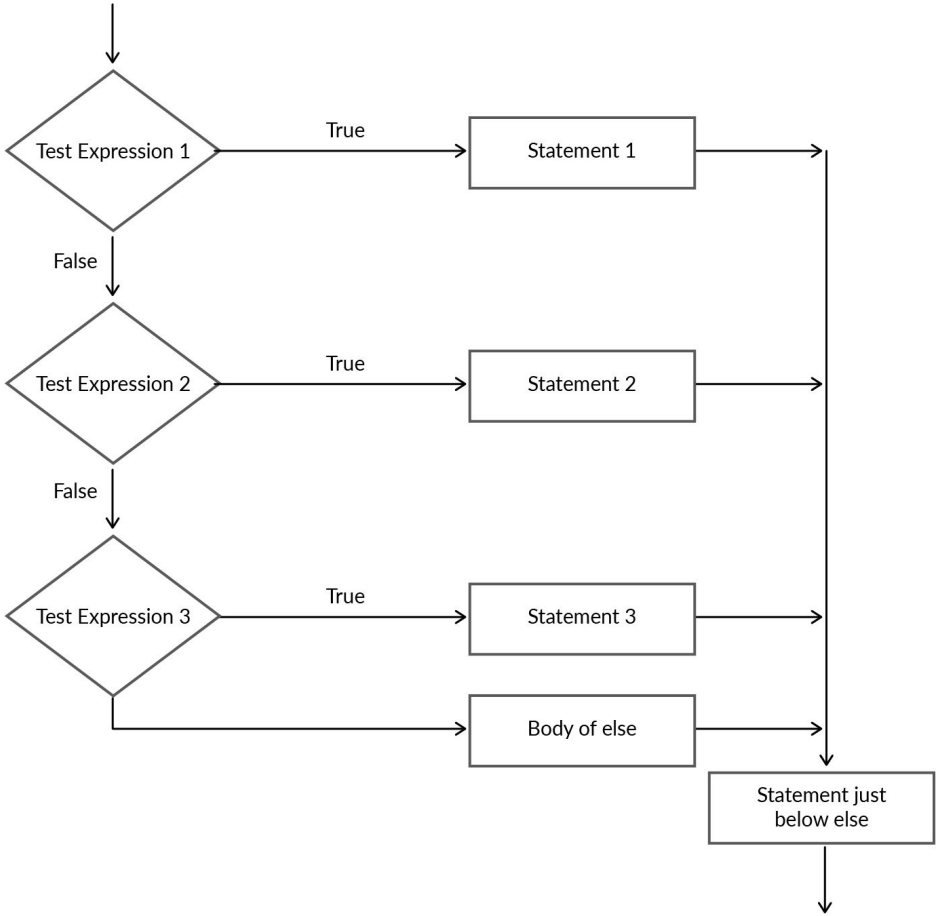
In such cases, we use the else-if statement for checking the conditions. Only those statements inside the conditional block whose condition is satisfied are executed.

If none of the conditions is satisfied, then the else block is executed.

Syntax:

```
if(condition1){
  //statement(s) to be executed when the condition is met
} else if(condition 2) {
   //statement(s) to be executed when the condition 2 is met.
} else if(condition 3) {
   //statement(s) to be executed when the condition 3 is met.
} else {
   //statement(s) to be executed when none of the conditions is met.
}
```

Else-If Statement: Execution



Else-If Statement: Execution

upGrad

```
public class Main {

  public static void main(String[] args) {
     //if - else  Statements
     Scanner sc = new Scanner(System.in);
     int a = sc.nextInt();
     if(a > 0) {
        System.out.println("a is a positive number");
     } else if( a < 0) {
        System.out.println("a is a negative number");
     } else {
        System.out.println("a is zero");
     }
  }
}
```

Hands-On Exercise 3

upGrad

Write a program that decides whether a person is diabetic or not depending on their sugar level as per the following ranges:

1. If less than 100 and fasting - normal
2. If less than 130 before a meal - normal
3. If less than 180 after taking a meal - normal
4. If none of the above - diabetic

Nested If-Else Statement

upGrad

Suppose you want to buy an ice cream from a local shop near your house. You would probably follow the below steps:

1. Go to the shop
2. Ask for ice cream
3. Select the flavour
4. Pay the price

Now, what if the shopkeeper is out of ice cream? Would you still execute the next steps? No.'There are situations where you are not required to execute all the conditions to arrive at a decision. Another example would be checking whether a number is divisible by 8. If the number is not divisible by 2, then it means that it is not divisible by 8.

For such cases, we use the nested if-else statements. ‘Nested if-else’ basically means that there is an if-else statement inside another if-else statement.

Nested If-Else Statement

upGrad

Syntax:

```
   if (condition1) {
     if (condition1A) {
        Action1A;
     }
     else {
        Action1B;
     }
  }
  else {
     Action2;
  }
}
```

Hands-On Exercise 4

upGrad

An online car rental application has the following steps to rent a car:

a. Is the user looking to rent a car in Mumbai?
b. If yes, is the number of hours requested by the user greater than 3?
c. If yes, is the total bill higher than ₹300?
d. If yes, the car is booked.
e. Otherwise, the car is not booked.

Write a program that takes the following inputs from the user:

1. Whether they are looking to rent a car in Mumbai
2. Number of hours for which the car is needed

Note: The bill is calculated by multiplying the number of hours by 100.

Switch-Case Statement

upGrad

- Consider a scenario of grading students in a college. A student can have one of the fixed grades depending on his/her marks.
- In situations like these where there are multiple fixed alternatives, we can use a switch-case statement.
- Thus, switch-case statements help us to solve everyday life scenarios that require us to choose between different alternatives.
- A switch-case statement executes different parts of code based on the value of an expression.
- Expression can be of type byte, short, int, char, enumeration or String.
- We use the break statement to terminate the sequence.
- There is also a default statement that is similar to the else statement.

Switch-Case Statement

upGrad

Note:

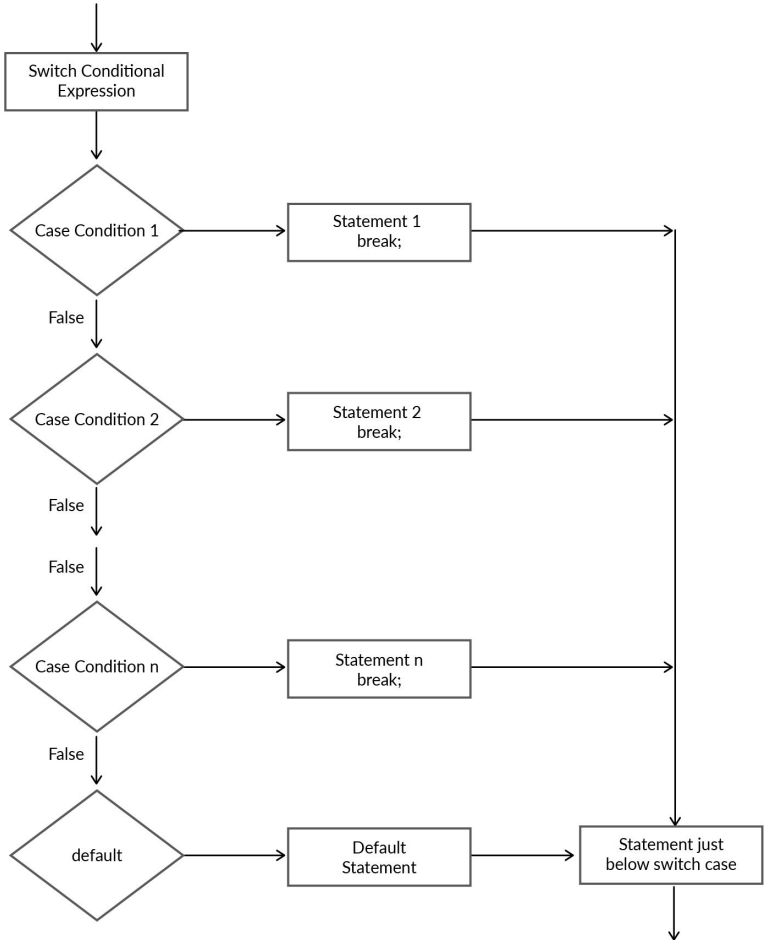
- Duplicate case values are not allowed.
- The default statement is optional.
- The break statement is optional. If omitted, the execution will continue in the next case.

Syntax:

```
switch (variable) {
        case match1:
           Action1;
           break;

        case match2:
           Action2;
           break;
   .
   .
   .
        case matchN:
           ActionN;
           break;

        default:
           DefaultAction;
           break;
```

Switch-Case Statement: Execution

upGrad



Switch-Case Statement

upGrad

```
public class Main {

  public static void main(String[] args) {
     String light = "Green";
     String lightString;
     switch (light) {
        case "Green":  lightString = "You can drive";
           break;
        case "Yellow":  lightString = "Slow down";
           break;
        case "Red":  lightString = "Stop";
           break;
        default: lightString = "Invalid color";
           break;
     }
     System.out.println(lightString);
  }
}
```

Hands-On Exercise 4

upGrad

You want to find out which is the current month on the calendar. Take an integer input from the user and find out which month is it. **Note:** The default should be January.

Poll 7 (30 sec.)

upGrad

What would be the output of the following code?

```
int num1 = 25;
int num2 = 5;
char operator = '/';
switch (operator){
  case '/':
     System.out.println(num1 / num2);
  case '+':
     System.out.println(num1 + num2);
     break;
  case '-':
     System.out.println(num1 - num2);
     break;
  case '*':
     System.out.println(num1 * num2);
     break;
}
```

1. 5
2. 5
   30
3. 30
4. None of the above

Poll 7 (Answer)

upGrad

What would be the output of the following code?

```
int num1 = 25;
int num2 = 5;
char operator = '/';
switch (operator){
  case '/':
     System.out.println(num1 / num2);
  case '+':
     System.out.println(num1 + num2);
     break;
  case '-':
     System.out.println(num1 - num2);
     break;
  case '*':
     System.out.println(num1 * num2);
     break;
}
```

1. 5
2. **5**
   30
3. 30
4. None of the above

Ternary Operator

upGrad

- The ternary operator (?:) is a conditional statement similar to the if-else statement.
- It is a one-liner replacement for the if-then-else statement.
- It takes extremely less space and makes the code short.

Syntax: variable x = (expression) ? value if true: value if false

```
public class Main {

  public static void main(String[] args) {
     int x = 10;
     int y = 12;
     int maxValue = (x > y) ? x : y;
     System.out.println(maxValue); //12
  }
}
```

Hands-On Exercise 5

upGrad

Find the absolute value of a number using the ternary operator.

Poll 8 (30 sec.)

Suppose you are designing a colour detector that prints the name of an input colour. There can be only two colours: black and white. If the input colour is black, then you need to print ‘black’, and if the input colour is white, then you need to print ‘white’. Which of the following would be the best to use?

1. If statement
2. If-else statement
3. Switch-case statement
4. Ternary operator

Poll 8 (Answer)

Suppose you are designing a colour detector that prints the name of an input colour. There can be only two colours: black and white. If the input colour is black, then you need to print ‘black’, and if the input colour is white, then you need to print ‘white’. Which of the following would be the best to use?

1. If statement
2. **If-else statement**
3. Switch-case statement
4. **Ternary operator**

Poll 9 (30 sec.)

A program needs to make a decision based on the value of an expression. Which of the following would be the best to use for the program?

1. If statement
2. If-else statement
3. Switch-case statement
4. Ternary operator

Poll 9 (Answer)

A program needs to make a decision based on the value of an expression. Which of the following would be the best to use for the program?

1. If statement
2. If-else statement
3. **Switch-case statement**
4. Ternary operator

Today's Homework

upGrad

Find out the largest number among the three given numbers.

Input Format:
The first line contains the first number
The second line contains the second number
The third line contains the third number

Output Format:
The first line contains the value of the largest number

Sample Input 1:
12
4
5

Sample Output 1:
12

Tasks to Complete After the Session

Homework Questions
MCQs
Coding Questions

upGrad

Next Class

1 Loops in Java with break and continue statements

2 Methods in Java

upGrad

#RahoAmbitious

Thank you!