**8th June, 2020 JESUS AND MARY SCHOOL & COLLEGE MODULE 2**

**CLASS 10**

**COMPUTER APPLICATION**

**INPUT IN JAVA & MATHEMATICAL LIBRARY METHOD**

**Type of Error in Java**

**There are three kinds of errors: syntax errors, runtime errors, and logic errors.**

**Syntax Erro**r-Syntax Error is due to lack of knowledge in a specific language. It is due to somebody does not know how to use the features of a language. We can know the errors at the time of compilation. It happens at the time of compilation. such errors need to be rectified before proceeding further.

**Logical Erro**r-It is due to the poor understanding of the requirement or problem. It may happen that a program contains no syntax or run-time errors but still it doesn't produce the correct O/P. It is because the developer has not understood the problem statement properly. These errors are hard to detect as well.
e.g. Error caused when any loop is not closed at the right place.

**Run time Erro**r-The exceptions like divide a number by 0, overflow and underflow comes under this. Error occurs at run-time. Such error cause a program to end abruptly or even cause system shut-down. Such errors are hard to detect.e.g. Error caused due to low system memory.

# Java Comments

The [Java](https://www.javatpoint.com/java-tutorial) comments are the statements that are not executed by the compiler and interpreter. The comments can be used to provide information or explanation about the [variable](https://www.javatpoint.com/java-variables), method, [class](https://www.javatpoint.com/object-and-class-in-java) or any statement. It can also be used to hide program code.

## Types of Java Comments

There are three types of comments in Kava.

1. Single Line Comment
2. Multi Line Comment
3. Documentation Comment

**1) Java Single Line Comment**

The single line comment is used to comment only one line.

1. //This is single line comment

**Example:**

1. **public** **class** CommentExample1 {
2. **public** **static** **void** main(String[] args) {
3. **int** i=10;//Here, i is a variable
4. System.out.println(i);

Output:

10

**2) Java Multi Line Comment**

The multi line comment is used to comment multiple lines of code.

**Syntax:**

1. /\*
2. This
3. is
4. multi line
5. comment
6. \*/

**Example:**

1. **public** **class** CommentExample2 {
2. **public** **static** **void** main(String[] args) {
3. /\* Let's declare and
4. print variable in java. \*/
5. **int** i=10;
6. System.out.println(i);

**Output:**

**10**

## 3) Java Documentation Comment

This kind of Java comments is utilized by large code for a programming bundle since it produces a documentation page for reference, which can be utilized for getting data about strategies, its parameters, and so forth.

**Syntax:**

1. /\*\*
2. This
3. is
4. documentation
5. comment
6. \*/

**Example:**

1. /\*\* The Calculator class provides methods to get addition and subtraction of given 2 numbers.\*/
2. **public** **class** Calculator {
3. /\*\* The add() method returns addition of given numbers.\*/
4. **public** **static** **int** add(**int** a, **int** b){**return** a+b;}
5. /\*\* The sub() method returns subtraction of given numbers.\*/
6. **public** **static** **int** sub(**int** a, **int** b){**return** a-b;}
7. }

**Inbuilt Mathematical Function**

**Some important mathematical functions are used in Java -**

### *Math.abs()*

The **Math.abs()** function returns the absolute value of the parameter passed to it. The absolute value is the positive value of the parameter. If the parameter value is negative, the negative sign is removed and the positive value corresponding to the negative value without sign is returned. Here are two **Math.abs()** method examples:

**int abs1 = Math.abs(10); // abs1 = 10**

**int abs2 = Math.abs(-20); // abs2 = 20**

### *Math.ceil()*

The **Math.ceil()** function rounds a floating point value up to the nearest integer value. The rounded value is returned as a **double**. Here is a **Math.ceil()** Java example:

**double c = Math.ceil(7.343); // c = 8.0**

**double c = Math.ceil(-7.343); // c = 7.0**

### *Math.floor()*

The **Math.floor()** function rounds a floating point value down to the nearest integer value. The rounded value is returned as a **double**. Here is a **Math.floor()** Java example:

**double f = Math.floor(7.343); // f = 7.0**

**double f = Math.floor(-7.343); // f = 8.0**

### *Math.min()*

The **Math.min()** method returns the smallest of two values passed to it as parameter. Here is a **Math.min()** Java example:

**int min = Math.min(10, 20);**

After executing this code the **min** variable will contain the value 10.

### *Math.max()*

The **Math.max()** method returns the largest of two values passed to it as parameter. Here is a **Math.max()** Java example:

**int max = Math.max(10, 20);**

After executing this code the **max** variable will contain the value 20.

### *Math.round()*

The **Math.round()** method rounds a **float** or **double** to the nearest integer using normal math round rules (either up or down). Here is a Java **Math.round()** example:

**double roundedDown = Math.round(23.445);**

**double roundedUp = Math.round(23.545);**

After executing these two Java statements the **roundedDown** variable will contain the value **23.0** , and the **roundedUp** variable will contain the value **24.0**.

### *Math.random()*

The **Math.random()** method returns a random floating point number between 0 and 1. Of course the number is not fully random, but the result of some calculation which is supposed to make it as unpredictable as possible. Here is a Java **Math.random()** example:

**double random = Math.random();**

To get a random value between 0 and e.g. 100, multiply the value returned by **Math.random()** with the maximum number (e.g. 100). Here is an example of how that might look:

### *Math.log()*

The **Math.log()** method provides the logarithm of the given parameter. The base for the logarithm is *i* (Euler's number). Thus, **Math.log()** provides the reverse function of **Math.exp()**. Here is a Java **Math.log()** example:

**double log1 = Math.log(1);**

**System.out.println("log1 = " + log1);**

**double log10 = Math.log(10);**

**System.out.println("log10 = " + log10);**

### *Math.pow()*

The **Math.pow()** function takes two parameters. The method returns the value of the first parameter raised to the power of the second parameter. Here is a **Math.pow()** Java example:

**double p1 = Math.pow(2,2);**

**System.out.println("p1 = " + p1);**

**double p2 = Math.pow(2,8);**

**System.out.println("p2 = " + p2);**

### *Output:*

P1= 4.0

P2=256.0

### *Math.sqrt()*

The **Math.sqrt()** method calculates the square root of the parameter given to it. Here are a few Java **Math.sqrt()** example:

**double sqrt4 = Math.sqrt(4);**

**System.out.println("sqrt4 = " + sqrt4); output sqrt4=2.0**

### *Math.cbrt()*

The **Math.cbrt()** method calculates the sine value of some angle value in radians. Here is a java  **Math.cbrt()**  example:

**double b = Math.cbrt(27);**

**System.out.println("cube root = " + b);**

**Output: cube root = 3**

### *Math.sin()*

The **Math.sin()** method calculates the sine value of some angle value in **radians.** Here is a Java **Math.sin()** example:

**double sin = Math.sin(Math.PI);**

**System.out.println("sin = " + sin);**

### *Math.cos()*

The **Math.cos()** method calculates the cosine value of some angle value in **radians.** Here is a Java **Math.cos()** example:

**double cos = Math.cos(Math.PI);**

**System.out.println("cos = " + cos);**

### *Math.tan()*

The **Math.tan()** method calculates the tangens value of some angle value in **radians**. Here is a Java **Math.tan()** example:

**double tan = Math.tan(Math.PI);**

**System.out.println("tan = " + tan);**

# Decision Making

Decision making structures have one or more conditions to be evaluated or tested by the program, along with a statement or statements that are to be executed if the condition is determined to be true, and optionally, other statements to be executed if the condition is determined to be false.

There are various types of if statement in Java.

* if statement
* if-else statement
* if-else-if ladder
* nested if statement

## Java if Statement

The Java if statement tests the condition. It executes the if block if condition is true.

**Syntax:**

1. **if**(condition){
2. //code to be executed
3. }

**Example:**

1. //Java Program to demonstate the use of if statement.
2. **public** **class** IfExample {
3. **public** **static** **void** main(String[] args) {
4. //defining an 'age' variable
5. **int** age=20;
6. //checking the age
7. **if**(age>18){
8. System.out.print("Age is greater than 18");  }
9. }

Output:

Age is greater than 18

**Java if-else Statement**

The Java if-else statement also tests the condition. It executes the *if block* if condition is true otherwise *else block* is executed.

**Syntax:**

1. **if**(condition){
2. //code if condition is true
3. }**else**{
4. //code if condition is false
5. }

**Example:**

1. //A Java Program to demonstrate the use of if-else statement.
2. //It is a program of odd and even number.
3. **public** **class** IfElseExample {
4. **public** **static** **void** main(String[] args) {
5. //defining a variable
6. **int** number=13;
7. //Check if the number is divisible by 2 or not
8. **if**(number%2==0){
9. System.out.println("even number");
10. }**else**{
11. System.out.println("odd number");
12. }
13. }
14. }

Output: odd number

**Java if-else-if ladder Statement**

The if-else-if ladder statement executes one condition from multiple statements.

**Syntax:**

1. **if**(condition1){
2. //code to be executed if condition1 is true
3. }**else** **if**(condition2){
4. //code to be executed if condition2 is true
5. }
6. **else** **if**(condition3){
7. //code to be executed if condition3 is true
8. }
9. ...
10. **else**{
11. //code to be executed if all the conditions are false
12. }

**Example:**

**Program to check POSITIVE, NEGATIVE or ZERO:**

1. **public** **class** PositiveNegativeExample {
2. **public** **static** **void** main(String[] args) {
3. **int** number=-13;
4. **if**(number>0){
5. System.out.println("POSITIVE");
6. }**else** **if**(number<0){
7. System.out.println("NEGATIVE");
8. }**else**{
9. System.out.println("ZERO");
10. }
11. }
12. }

Output: NEGATIVE

**Java Nested if statement**

The nested if statement represents the *if block within another if block*. Here, the inner if block condition executes only when outer if block condition is true.

**Syntax:**

1. **if**(condition){
2. //code to be executed
3. **if**(condition){
4. //code to be executed
5. }
6. }

**Example:**

1. //Java Program to demonstrate the use of Nested If Statement.
2. **public** **class** JavaNestedIfExample {
3. **public** **static** **void** main(String[] args) {
4. //Creating two variables for age and weight
5. **int** age=20;
6. **int** weight=80;
7. //applying condition on age and weight
8. **if**(age>=18){
9. **if**(weight>50){
10. System.out.println("You are eligible to donate blood");
11. }
12. }
13. }}

Output: You are eligible to donate blood

# Java Switch Statement

The Java switch statement executes one statement from multiple conditions. It is like [if-else-if](https://www.javatpoint.com/java-if-else) ladder statement. The switch statement works with byte, short, int, long, enum types, String and some wrapper types like Byte, Short, Int, and Long. Since Java 7, you can use [strings](https://www.javatpoint.com/java-string) in the switch statement.

In other words, the switch statement tests the equality of a variable against multiple values.

#### Points to Remember

* There can be one or N number of case values for a switch expression.
* The case value must be of switch expression type only. The case value must be literal or constant. It doesn't allow [variables](https://www.javatpoint.com/java-variables).
* The case values must be unique. In case of duplicate value, it renders compile-time error.
* Each case statement can have a break statement which is optional. When control reaches to the [break statement](https://www.javatpoint.com/java-break), it jumps the control after the switch expression. If a break statement is not found, it executes the next case.
* The case value can have a default label which is optional.

**Syntax:**

1. **switch**(expression){
2. **case** value1:
3. //code to be executed;
4. **break**;  //optional
5. **case** value2:
6. //code to be executed;
7. **break**;  //optional
8. ......
9.
10. **default**:
11. code to be executed **if** all cases are not matched;
12. }

**Example:**

1. **public** **class** SwitchExample {
2. **public** **static** **void** main(String[] args) {
3. //Declaring a variable for switch expression
4. **int** number=20;
5. //Switch expression
6. **switch**(number){
7. //Case statements
8. **case** 10: System.out.println("10");
9. **break**;
10. **case** 20: System.out.println("20");
11. **break**;
12. **case** 30: System.out.println("30");
13. **break**;
14. //Default case statement
15. **default**:System.out.println("Not in 10, 20 or 30");
16. }
17. }
18. }

Output: 20

**WORKSHEET**

Q1.What is the difference between Logical and Run time error?

Q2. What is difference between single line and multi line comment statement?

Q3. What is the difference between if else and switch statement?

Q4. What is the important of default statement in switch statement?

Q5**. Give output**-

 class output {

 public static void main(String[] args) {

 int i = 7;

 int j = -9;

 double x = 72.3;

 double y = 0.34;

 System.out.println("|" + i + "| is " + Math.abs(i));

 System.out.println("|" + j + "| is " + Math.abs(j));

 System.out.println("|" + x + "| is " + Math.abs(x));

 System.out.println(x + " is approximately " + Math.round(x));

 System.out.println("The ceiling of " + j + " is " + Math.ceil(j));

 System.out.println("The ceiling of " + x + " is " + Math.ceil(x));

 System.out.println("The floor of " + j + " is " + Math.floor(j));

 System.out.println("The floor of " + y + " is " + Math.floor(y));

 System.out.println("min(" + x + "," + y + ") is " + Math.min(x,y));

 System.out.println("max(" + i + "," + x + ") is " + Math.max(i,x));

 System.out.println("max(" + y + "," + j + ") is " + Math.max(y,j));

 System.out.println("Pi is " + Math.PI);

 System.out.println("e is " + Math.E);

 // Convert a 45 degree angle to radians

 double angle = 45.0 \* 2.0 \* Math.PI/360.0;

 System.out.println("cos(" + angle + ") is " + Math.cos(angle));

 System.out.println("sin(" + angle + ") is " + Math.sin(angle));

 System.out.println("pow(2.0, 2.0) is " + Math.pow(2.0,2.0));

 System.out.println("pow(10.0, 3.5) is " + Math.pow(10.0,3.5));

 System.out.println("pow(8, -1) is " + Math.pow(8,-1));

 The square root of " + j + " is " + Math.sqrt(j));

 }

 **Note: -** Please do all this work in your copies which will be check when school be re-open.

 **Please consider this important**