***OBJECT ORIENTED PROGRAMMING THROUGH JAVA***

**Course Objectives:-**

1. Gain knowledge about basic Java language syntax and semantics to write Java programs and use concepts such as variables, conditional and iterative execution methods etc

2. Understand the basic object oriented programming concepts and apply them in problem solving.

3. Illustrate inheritance concepts for reusing the program.

4. Demonstrate on the multi-tasking by using multiple threads.

5. Understand the basics of java console and GUI based programming.

**Course Outcomes:-**

1. Gain knowledge about basic Java language syntax and semantics to write Java programs and use concepts such as variables, conditional and iterative execution methods etc

2. Understand the basic object oriented programming concepts and apply them in problem solving.

3. Illustrate inheritance concepts for reusing the program.

4. Demonstrate on the multi-tasking by using multiple threads.

5. Understand the basics of java console and GUI based programming.

**UNIT-01**

***The Java Language***

**About Java:-**

Java is a general-purpose, class-based, object-oriented programming language designed for having lesser implementation dependencies. It is a computing platform for application development. Java is fast, reliable, and secure. It is also portable, which means that Java code can run on any platform that has a Java Virtual Machine (JVM).

Java is one of the most popular programming languages in the world. It is used to develop a wide variety of applications, including web applications, mobile applications, desktop applications, and enterprise software.

* **Object-oriented:**

Java is an object-oriented language, which means that it is based on the concept of objects. Objects are self-contained entities that contain data and methods.

* **Class-based:**

Java is a class-based language, which means that it is based on the concept of classes. Classes are blueprints for objects. They define the data and methods that objects can have.

* **Portable:**

Java is a portable language, which means that Java code can run on any platform that has a Java Virtual Machine (JVM).

* **Secure:**

Java is a secure language, which means that it is designed to be difficult to exploit for security vulnerabilities.

* **Robust:**

Java is a robust language, which means that it is designed to be reliable and resistant to errors.

**Features:-**

* Java syntax is designed to be simple and easy to understand, which makes it a good choice for beginners.
* It is one of the most popular programming language in the world.
* It is easy to learn and simple to use.
* It is open source and free.
* It is secure, fast and powerful.
* Java is an object oriented language.
* As Java is close to c++ and c#, it makes it easy for programmers to switch to Java or vice versa.
* Java is a platform-independent language, which means that Java code can be run on any platform that has a Java Virtual Machine (JVM).
* Java is a robust and secure language, which means that Java code is less likely to crash or be hacked. This makes Java a good choice for developing enterprise applications.
* Java is a good choice for developing distributed applications, which means that Java code can run on multiple computers simultaneously. This makes Java code more scalable and fault-tolerant.
* Java is a dynamic and extensible language, which means that Java code can be changed and extended at runtime. This makes Java code more flexible and adaptable.

**IDE:-**

IDE-Integrated Development Environment

* Eclipse
* EntellijIDEA
* Net Beans
* J Developer
* Blue J
* J Creator
* Dr Java
* My Eclipse
* J GRASP
* Greenfoot
* Android Studio
* Xcode
* Datagrip
* Visual Studio
* Cloud 9 IDE
* Eclipse Che
* Anjuta
* Nodeclipse

**Types of Errors in Java:-**

1. **Syntax** **Errors:-** These errors occur when the code violates Java’s syntax rulers.
2. **Runtime** **Errors:-** These errors occur when the program is running and encounters a situation that it cannot handle.
3. **Logical** **Errors:-** These errors occur when the program’s logic is incorrect.

**Structure of Java program:-**

|  |
| --- |
| Documentation Section |
| Package Statements |
| Import Statements |
| Interface Statements |
| Class Definitions |
| Main method class  {  Main method definition  } |

**Applications of Java:-**

* Big Data
* Cloud Computing
* Web applications
* Scientific applications
* Embedded systems
* Gaming applications
* Android development
* Artificial Intelligence
* Desktop GUI applications
* Enterprise applications
* Server apps in finance
* J2ME apps

**Steps to Download Java:-**

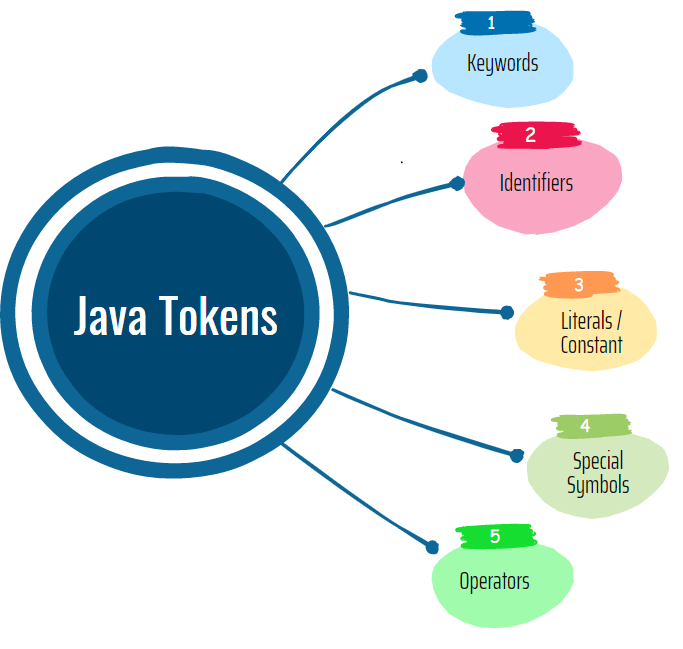
* Go to the Oracle Java Downloads page.
* Select the version of Java you want to download.
* Click JDK Download.
* Accept the license agreement.
* Select the download link for your version of Windows.
* Download the file.
* Launch the installation file.

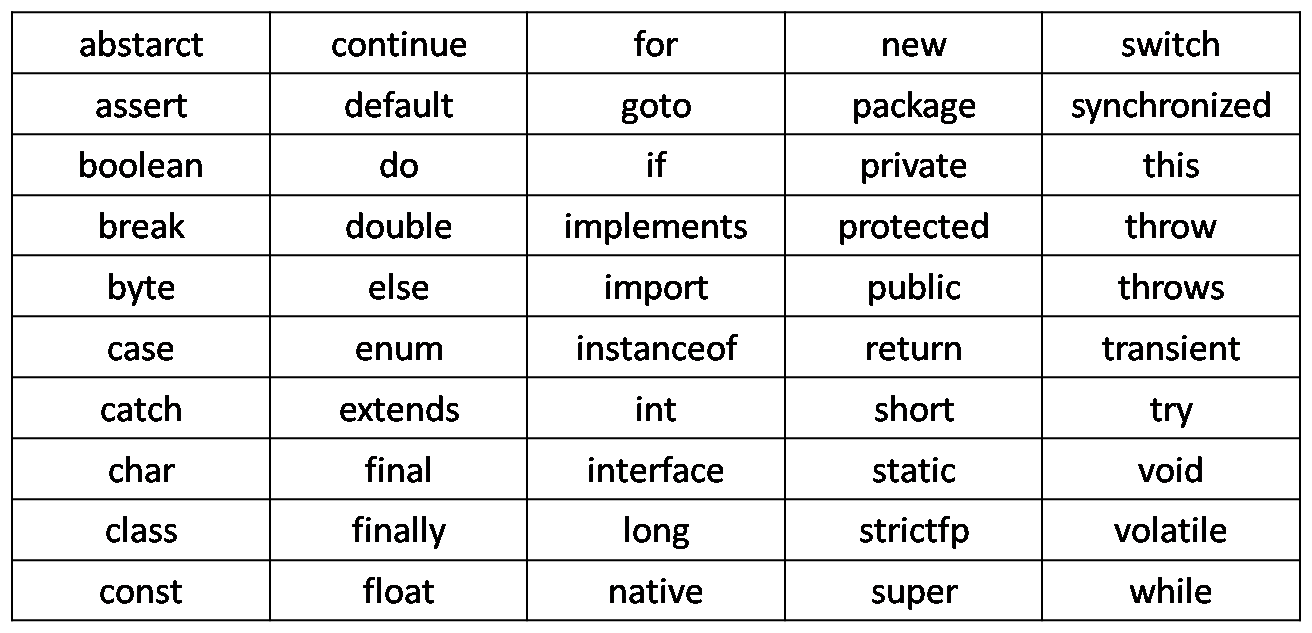
**Set Path in Java:-**

* Open the System Properties window.
* Click on the Advanced tab.
* Click on the Environment Variables button.
* In the System Variables section, find the PATH variable and click on the Edit button.
* Add the path to the JDK bin directory to the end of the PATH variable.
* Click OK to save the changes.

**Java Tokens:-**

The program contains classes and methods. Further, the methods contain the expressions and statements required to perform a specific operation. These statements and expressions are made up of **tokens**. In other words, we can say that the expression and statement is a set of **tokens**. The tokens are the small building blocks of a Java program that are meaningful to the Java compiler. Further, these two components contain variables, constants, and operators. In this section, we will discuss **what is tokens in Java**.



**1.Keywords:-** These are the **pre-defined** reserved words of any programming language. Each keyword has a special meaning. It is always written in lower case. 

**2.Identifiers:-** Identifiers are used to name a variable, constant, function, class, and array. It usually defined by the user. It uses letters, underscores, or a dollar sign as the first character. The label is also known as a special kind of identifier that is used in the goto statement. Remember that the identifier name must be different from the reserved keywords.

**Rules to declare Identifiers:-**

* The variable name must start with a letter, dollar sign ($), or underscore (\_).
* The variable name can contain letters, digits, dollar signs, and underscores.
* The variable name cannot contain spaces.
* The variable name cannot be a Java keyword.

**Examples:**

1. PhoneNumber

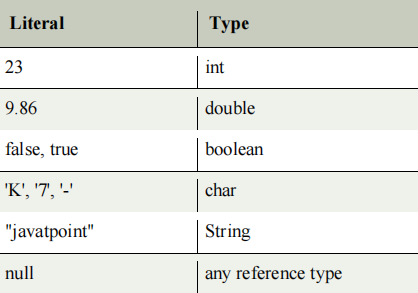
2. PRICE

3. radius

4. a

5.a1

**3.Literals:-** In programming literal is a notation that represents a fixed value (constant) in the source code. It can be categorized as an integer literal, string literal, Boolean literal, etc. It is defined by the programmer. Once it has been defined cannot be changed.



**4. Separators:-**

The separators in Java is also known as **punctuators**. There are nine separators in Java, are as follows:

1. separator <= ; | , | . | ( | ) | { | } | [ | ]

o **Square Brackets []:** It is used to define array elements. A pair of square brackets represents the single-dimensional array, two pairs of square brackets represent the two-dimensional array.

o **Parentheses ():** It is used to call the functions and parsing the parameters.

o **Curly Braces {}:** The curly braces denote the starting and ending of a code block.

o **Comma (,):** It is used to separate two values, statements, and parameters.

o **Assignment Operator (=):** It is used to assign a variable and constant.

o **Semicolon (;):** It is the symbol that can be found at end of the statements. It separates the two

statements.

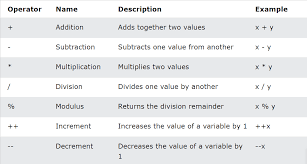
o **Period (.):** It separates the package name form the sub-packages and class. It also separates a

variable or method from a reference variable.

**5.Operators:-** In programming, operators are the special symbol that tells the compiler to perform a special operation. Java provides different types of operators that can be classified according to the functionality they provide.

* Arithmetic Operators
* Assignment Operators
* Bitwise Operators
* Relational Operators
* Ternary Operators
* Logical Operators
* Unary Operators
* Shift Operators

**Arithmetic Opertors:-** Arithmetic operators are the symbols used to perform basic mathematical operators data.



**Program:-**

//Arithmetic operators

class arith

{

public static void main(String args[])

{

int a=10,b=5;

System.out.println(“a+b=”+(a+b));

System.out.println(“a-b=”+(a-b));

System.out.println(“a\*b=”+(a\*b));

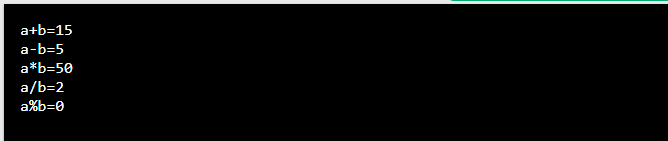
System.out.println(“a/b=”+(a/b));

System.out.println(“a%b=”+(a%b));

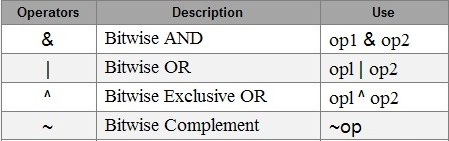
}

}

**Output:-**



**Bitwise** **Operator:-** Bitwise operators are used to performing the manipulation of individual bits of a number. They can be used with any integral type (char, short, int, etc.). They are used when performing update and query operations of the Binary indexed trees.



**Truth Table:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| X | Y | X&Y | X|Y | X^Y | ~X |
| 1 | 1 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 1 |

**Program:-**

//Bitwise operators

class bitwise

{

public static void main(String args[])

{

int a=3, b=4;

System.out.println(“a&b=”+(a&b));

System.out.println(“a|b=”+(a|b));

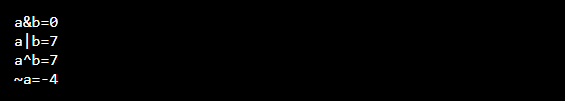
System.out.println(“a^b=”+(a^b));

System.out.println(“~a=”+(~a));

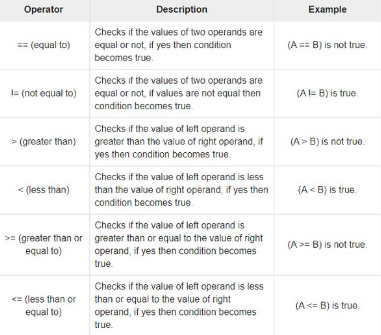
}

}

**Output:-**



**Relational** **Operators:- Java Relational Operators** are a bunch of binary operators used to check for relations between two operands, including equality, greater than, less than, etc. They return a boolean result after the comparison and are extensively used in looping statements as well as conditional if-else statements and so on.



**Program:-**

//Relational operators

import java.util.\*;

class comp

{

public static void main(String args[])

{

Scanner obj=new Scanner(System.in);

int a=obj.nextInt();

int b=obj.nextInt();

System.out.println(“a>b “+(a>b));

System.out.println(“a<b “+(a<b));

System.out.println(“a>=b “+(a>=b));

System.out.println(“a<=b “+(a<=b));

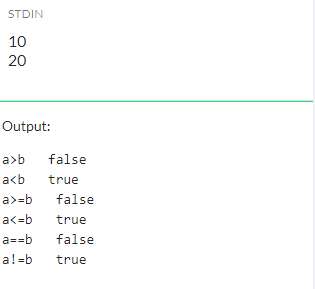
System.out.println(“a==b “+(a==b));

System.out.println(“a!=b “+(a!=b));

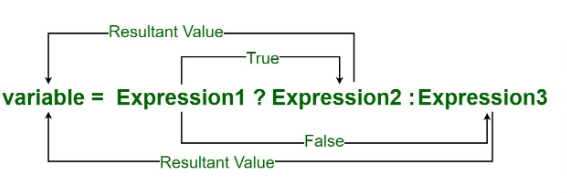
}

}

**Output:-**



**Conditional** **Operator:-** Java ternary operator is the only conditional operator that takes three operands. It’s a one-liner replacement for the if-then-else statement and is used a lot in Java programming. We can use the ternary operator in place of if-else conditions or even switch conditions using nested ternary operators. Although it follows the same algorithm as of if-else statement, the conditional operator takes less space and helps to write the if-else statements in the shortest way possible.



**Program:-**

//Ternary operator

import java.util.\*;

class ternary

{

public static void main(String args[])

{

int a, b, result;

Scanner obj=new Scanner(System.in);

a=obj.nextInt();

b=obj.nextInt();

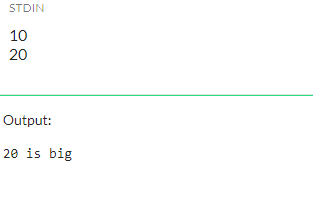
result=a>b?a:b;

System.out.println(result+” is big”);

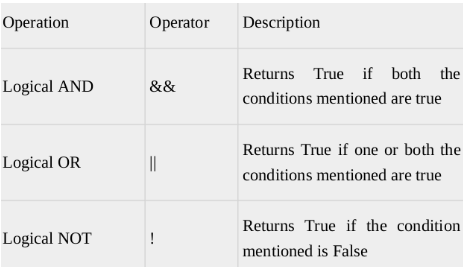
}

}

**Output:-**



**Logical Operators:-** Logical operators are used to perform logical “AND”, “OR” and “NOT” operations, i.e. the function similar to AND gate and OR gate in digital electronics. They are used to combine two or more conditions.



**Program:-**

//Logical And

import java.util.\*;

class logicaland

{

public static void main(String args[])

{

int d;

Scanner obj=new Scanner(System.in);

int a=obj.nextInt();

int b=obj.nextInt();

int c=obj.nextInt();

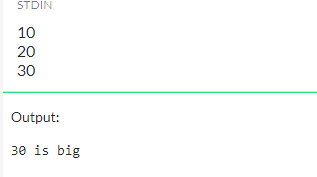
d=((a>b)&&(a>c))?a:(b>c?b:c);

System.out.println(d+” is big”);

}

}

**Output:-**



**Program:-**

//Logical Or

import java.util.\*;

class or

{

public static void main(String args[])

{

Scanner obj=new Scanner(System.in);

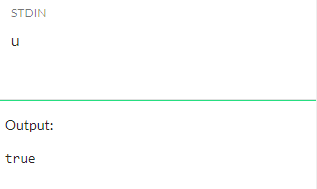
char ch=obj.next().charAt(0);

System.out.println((ch==’a’)||(ch==’e’)||(ch==’i’)||(ch==’o’)||(ch==’u’));

}

}

**Output:-**



**Unary Operators:-** Java unary operators are the types that need only one operand to perform any operation like increment, decrement, negation, etc. It consists of various arithmetic, logical and other operators that operate on a single operand.

|  |  |  |
| --- | --- | --- |
| **Types** | **Description** | **Example** |
| Pre Increment | First it performs increment and then it assigns value | ++i |
| Post Increment | First it assigns value and then it performs increment | i++ |
| Pre Decrement | First it performs decrement and then it assign value | --i |
| Post Decrement | First it assigns value and then it performs decrement | i-- |