**JAVA NOTES**

**Introduction**

Java technology is both a programming language and a platform.

**FEATURES OF JAVA**

1. Simple
2. Object-Oriented

Object-oriented because we organize the information as a combination of different types of objects that incorporates both data and behavior.

1. Portable

We can run java program on any machine which has JVM

1. Platform independent

The Java platform has two components:

* The Java Virtual Machine
* The Java Application Programming Interface (API) : the API contains libraries of related classes and interfaces; these libraries are known as packages*.*

1. Secured

It is secure because it has no explicit pointer and java runs inside a virtual machine

1. Robust

Its robust because strong memory management and exceptional handling

1. Architecture neutral

No implementation dependent features

1. Dynamic
2. High Performance
3. Multithreaded

Thread is like a separate program, executing concurrently. We can write Java programs that deal with many tasks at once by defining multiple threads. The main advantage of multi-threading is that it doesn't occupy memory for each thread. It shares a common memory area.

1. Distributed

Can create distributed applications. We may access files by calling the methods from any machine on the internet.

**BASIC PROGRAM**

**class** Simple{

**public** **static** **void** main(String args[]){

      System.out.println("Hello Java");

     }

}

* **class** keyword is used to declare a class in java.
* **public** keyword is an access modifier which represents visibility, it means it is visible to all.
* **static** is a keyword, if we declare any method as static, it is known as static method. The core advantage of static method is that there is no need to create object to invoke the static method. The main method is executed by the JVM, so it doesn't require to create object to invoke the main method. So it saves memory.
* **void** is the return type of the method, it means it doesn't return any value.
* **main** represents startup of the program.
* **String[] args** is used for command line argument. We will learn it later.
* **System.out.println()** is used print statement. We will learn about the internal working of System.out.println statement later.

**Object-Oriented Programming**

* **OBJECT: is** a software bundle of related state and behavior. An object stores its state in fields*,* and exposes its behavior through methods
* Hiding internal state and requiring all interaction to be performed through an object's methods is known as **data encapsulation** — a fundamental principle of object-oriented programming.

**CLASS**

* A class is a blueprint or prototype from which objects are created.

**class** <class\_name>{

    field;

    method;

}

# Inheritance

It is a mechanism in which one class acquires all the properties and behaviors of parent class. By using the keyword extends.

**class** childclass-name **extends** Parentclass-name

{

   //methods and fields

}

# Interface

It has static constants and abstract methods. There can be only abstract methods in the java interface not method body.

* a class extends another class, an interface extends another interface but a **class implements an interface**.
* By using the Keyword **Implements**

**Variables**

* **Instance Variables:** Non-static fields are also known as *instance variables* because their values are unique to each instance of a class.
* **Class Variables:** A class variable is any field declared with the static modifier; this tells the compiler that there is exactly one copy of this variable in existence, regardless of how many times the class has been instantiated.
* **Local Variables: These variables can be accessed only with the particular class where they are declared.**

**Primitive Data Types**

A primitive type is predefined by the language and is named by a reserved keyword. Java supports **8 Primitive data** **types**

* **byte**: The byte data type is an 8-bit, It has a minimum value of -128 and a maximum value of 127 . Its default value is 0
* **short**: The short data type is a 16-bit. Its default value is 0
* **int**: is a 32-bit the arithmetic operations for unsigned integers. Its default value is 0
* **long**: is a 64-bit. Its default value is 0L
* **float**: The float data type is a single-precision 32-bit. Its default value is 0.0f
* **double**: The double data type is a double-precision 64-bit. Its default value is 0.0d
* **boolean**: The boolean data type has only two possible values: true and false. Its default value is false
* **char**: The char data type is a single 16-bit Unicode character. Its default value is '\u0000'.

**Arrays**

An array is a container object that holds a fixed number of values of a single type. The length of an array is given when the array is created and it is fixed. In general it is 0 to n-1 (n=length).

**Operators**

**Simple Assignment Operator**

= Simple assignment operator

## **Arithmetic Operators**

+ Additive operator (also used

for String concatenation)

- Subtraction operator

\* Multiplication operator

/ Division operator

% Remainder operator

## **Unary Operators**

+ Unary plus operator; indicates

positive value (numbers are

positive without this, however)

- Unary minus operator; negates

an expression

++ Increment operator; increments

a value by 1

-- Decrement operator; decrements

a value by 1

! Logical complement operator;

inverts the value of a boolean

## **Equality and Relational Operators**

== Equal to

!= Not equal to

> Greater than

>= Greater than or equal to

< Less than

<= Less than or equal to

## **Conditional Operators**

&& Conditional-AND

|| Conditional-OR

?: Ternary (shorthand for

if-then-else statement)

## **Type Comparison Operator**

instanceof Compares an object to

a specified type

## **Bitwise and Bit Shift Operators**

~ Unary bitwise complement

<< Signed left shift

>> Signed right shift

>>> Unsigned right shift

& Bitwise AND

^ Bitwise exclusive OR

| Bitwise inclusive OR

# Control Flow Statements

# if-then :it is the most basic of all the control flow statements. It tells your program to execute a certain section of code only if a particular test evaluates to true.

# if-then-else statement :provides a secondary path of execution when an "if" clause evaluates to false.

# Switch statement: Unlike if-then and if-then-else, the switch statement allows for any number of possible execution paths.

# while and do-while statements continually execute a block of statements while a particular condition is true.

# THIS keyword

# this is a **reference variable** that refers to the current object. This keyword can be used for following purposes

1. this can be used to refer current class instance variable.
2. this can be used to invoke current class method (implicitly)
3. this() can be used to invoke current class constructor.
4. this can be passed as an argument in the method call.
5. this can be passed as argument in the constructor call.
6. this can be used to return the current class instance from the method.

* If a class has multiple methods having same name but different in parameters, it is known as **Method Overloading**. Method overloading is done within the same class
* If subclass (child class) has the same method as declared in the parent class, it is known as **method overriding in java**. Method overriding is done among different classes.

**ABSTRACT METHODS AND CLASSES**

An abstract class is a class that is declared abstract—it may or may not include abstract methods. Abstract classes cannot be instantiated, but they can be subclassed.

An abstract method is a method that is declared without an implementation (without braces, and followed by a semicolon)

Syntax: **abstract** **class** A {}

**FINAL CLASS**

The **final keyword** in java is used to restrict the user. If we make the class final we cannot extend it.

Syntax: final class classname{ }

**NUMBER CLASSES**

the Java platform provides **wrapper classes** for each of the primitive data types. These classes "wrap" the primitive in an object. It helps to convert primitive into object and object into primitive*.*

* if you use a primitive where an object is expected, the compiler *boxes* the primitive in its wrapper class
* if you use a number object when a primitive is expected, the compiler *unboxes* the object for you.

Numeric wrapper classes are subclasses of the abstract class Number they are:

Byte, double, float, integer, long, short

Converts the value of this Number object to the primitive data type returned.

byte byteValue()  
short shortValue()  
int intValue()  
long longValue()  
float floatValue()  
double doubleValue()

int compareTo() : Compares this Number object to the argument.

boolean equals(Object obj): Determines whether this number object is equal to the argument.

The MIN\_VALUE and MAX\_VALUE constants contain the smallest and largest values that can be contained by an object of that type.

The Math class contains a variety of class methods for performing mathematical functions

**EXCEPTIONS**

An exception is an event, which occurs during the execution of a program, that disrupts the normal flow of the program's instructions. It is an object which is thrown at runtime.

* Exception Handling is a mechanism to handle runtime errors
* Try-Catch block

Java try block is used to enclose the code that might throw an exception. It must be used within the method.

Java try block must be followed by either catch or finally block.

Syntax: **try**{

//code that may throw exception

}**catch**(Exception\_class\_Name ref){}

* Java supports multiple catch blocks
* **FINALLY block**

**Java finally block** is a block that is used *to execute important code*.

Java finally block is always executed whether exception is handled or not.

Java finally block follows try or catch block.

* **THROW KEYWORD**

 used to explicitly throw an exception mainly customized exceptions

SYNTAX**: throw** exception;