**PRACTICAL 1.1**

* Introduction to Object Oriented Concepts, comparison of Java with other object oriented programming languages. Introduction to JDK, JRE, JVM, javadoc, command line argument.

🡪Object means a real word entity such as pen, chair, table etc. Object- Oriented Programming is a methodology or paradigm to design a program using classes and objects. It simplifies the software development and maintenance by providing some concepts:

● Object

● Class

● Inheritance

● Polymorphism

● Abstraction

● Encapsulation

🡪**Object**: Any entity that has state and behaviour is known as an object. For example: chair, pen, table, keyboard, bike etc. It can be physical and logical.

🡪**Class**: Collection of objects is called class. It is a logical entity.

🡪**Inheritance**: When one object acquires all the properties and behaviours of parent object i.e. known as inheritance. It provides code reusability. It is used to achieve runtime polymorphism.

🡪**Polymorphism**: When one task is performed by different ways i.e. known as polymorphism. For example: to convince the customer differently, to draw something e.g. shape or rectangle etc. In java, we use method overloading and method overriding to achieve polymorphism. Another example can be to speak something e.g. cat speaks meaw, dog barks woof etc.

🡪**Abstraction**: Hiding internal details and showing functionality is known as abstraction. For example:

phone call, we don't know the internal processing.

In java, we use abstract class and interface to achieve abstraction.

🡪**Encapsulation**: Binding (or wrapping) code and data together into a single unit is known as encapsulation. For example: capsule, it is wrapped with different medicines.

A java class is the example of encapsulation. Java bean is the fully encapsulated class because all the data members are private here.

🡪**Difference between JDK, JRE and JVM**

Understanding the difference between JDK, JRE and JVM is important in Java. We are having brief overview of JVM here.

If you want to get the detailed knowledge of Java Virtual Machine, move to the next page. Firstly, let's see the basic differences between the JDK, JRE and JVM.

🡪**JVM**

JVM (Java Virtual Machine) is an abstract machine. It is a specification that provides runtime environment in which java bytecode can be executed. JVMs are available for many hardware and software platforms. JVM, JRE and JDK are platform dependent because configuration of each OS differs. But, Java is platform independent. The JVM performs following main tasks:

● Loads code

● Verifies code

● Executes code

● Provides runtime environment

**🡪JRE**

JRE is an acronym for Java Runtime Environment. It is used to provide runtime environment. It is the implementation of JVM. It physically exists. It contains set of libraries + other files that JVM uses at runtime.

Implementations of JVMs are also actively released by other companies besides Sun Micro Systems

🡪**JDK**

JDK is an acronym for Java Development Kit. It physically exists. It contains JRE + development tools.