**Assignment: (Core Java)**

**9 : Inheritance and Polymorphism**

**Que.1 Inheritance Types and Benefits**

**Ans.1**  Inheritance is one of the pillars of OOP It allows a class to acquire the properties and methods of another class using the extends keyword.

Java supports the following types of inheritance:

* Single Inheritance
* Multilevel Inheritance
* Hierarchical Inheritance
* Multiple Inheritance
* Hybrid Inheritance

**Single Inheritance:** In single inheritance, a sub-class is derived from only one super class. It inherits the properties and behavior of a single-parent class. Sometimes, it is also known as simple inheritance.

(Base class)

A

B

(Derived class)

**Multilevel:** In Multilevel Inheritance, a derived class will be inheriting a base class and as well as the derived class also acts as the base class for other classes.

A

(Base class 1)

(Base class 2)

B

(Derived class)

C

**Hierarchical:** In hierarchical inheritance, more than one subclass is inherited from a single base class. i.e. more than one derived class is created from a single base class.

(Base class)

Fruit

Banana

Apple

(Derived class)

Java does not support hybrid/multiple inheritance with classes

**Benefits of Inheritance**

1. **Code Reusability** – Common code written in the parent can be reused in child classes.
2. **Method Overriding (Runtime Polymorphism)** – Child can provide a specific implementation of a parent’s method.
3. **Extensibility** – Easy to extend existing classes without modifying them.
4. **Code Organization** – Helps structure related classes logically.
5. **Maintainability** – Centralized code in parent reduces duplication and makes updates easier.

**Que.2 Method Overriding**

**Ans.2** Method overriding occurs when a subclass provides a specific implementation for a method that is already defined in its superclass. The overridden method in the subclass must have the same name, parameters, and return type as the method in the superclass. This allows a subclass to specialize or change the behaviour of an inherited method, providing a more specific implementation relevant to its own context. The Override annotation can be used to explicitly indicate that a method is intended to override a superclass method, which helps in compile-time error checking.

**Que.3 Dynamic Binding (Run-Time Polymorphism)**

**Ans.3** Dynamic binding refers to the process in which linking between method call and method implementation is resolved at run time (or, a process of calling an overridden method at run time). Dynamic binding is also known as run-time polymorphism or late binding. Dynamic binding uses objects to resolve binding.

Characteristics of Java Dynamic Binding

* Linking − Linking between method call and method implementation is resolved at run time.
* Resolve mechanism − Dynamic binding uses object type to resolve binding.
* Example − [Method overriding](https://www.tutorialspoint.com/java/java_overriding.htm) is the example of Dynamic binding.
* Type of Methods − Virtual methods use dynamic binding.

**Que.4 Super Keyword and Method Hiding**

**Ans.4 The super keyword** provides a way to explicitly access members of the superclass from within a subclass. It is used in three primary scenarios:

* To call a superclass constructor: This is necessary for initializing the inherited parts of a new object. The super() call must be the first statement in the subclass's constructor.
* To access an overridden superclass method: If a subclass provides its own implementation of an inherited method (method overriding), super.methodName() can still be used to call the original superclass version.
* To access a hidden superclass field: When a subclass declares an instance variable with the same name as one in the superclass, super.fieldName resolves the ambiguity by explicitly referencing the superclass's field.

**Method hiding** applies exclusively to static methods. Unlike overriding, which resolves method calls at runtime based on the actual object type, method hiding is resolved by the compiler based on the reference variable's type.

If a subclass defines a static method with the same signature as a static method in its superclass, the subclass's method is said to "hide" the superclass's version.