**9**

**. Inheritance and Polymorphism**

**1)**

**In**

**heritance Types and Benefits**

**Answer: Inheritance means** properties of parent class extends into child class

: Properties of super class extends into subclass

: Main purpose is : Reusability , extendibility

: To used "extends" keyword through create inheritance

: Always called last child class to create object with access the properties of parent class except private

: There are mainly 5 types

1) Single: only one parent having only one child

class Animal

{

void eat()

{

System.out.println("This animal eats food.");

}

}

class Dog extends Animal

{

void bark()

{

System.out.println("The dog barks.");

}

}

public class SingleInheritance

{

public static void main(String[] args)

{

Dog dog = new Dog(); dog.eat(); // Inherited method dog.bark(); // Dog-specific method

}

}

2) Multilevel: single inheritance having one another child class Animal

{

void eat()

{

System.out.println("This animal eats food.");

}

}

class Mammal extends Animal {

void walk()

{

System.out.println("This mammal walks.");

}

}

class Dog extends Mammal

{

void bark()

{

System.out.println("The dog barks.");

}

}

public class MultilevelInheritance

{

public static void main(String[] args)

{

Dog dog = new Dog(); dog.eat(); // From Animal class dog.walk(); // From Mammal class dog.bark(); // From Dog class

}

}

1. Hierarchical: one parent having 2 or more child
2. Multiple: java does not support directly
3. Hybrid: java does not support directly

# Method Overriding

**Answer:** The whole signature of the method should be same in super class as well as in subclass but its behaviors (body part of the method) are different.

Syntax:

class ParentClass

{ void show()

{

System.out.println("This is the parent class method.");

}

}

class ChildClass extends ParentClass

{ void show()

{

System.out.println("This is the child class method.");

}

}

# Dynamic Binding (Run-Time Polymorphism)

**Answer: Dynamic Binding**, also known as **Run-Time Polymorphism**, is a mechanism where the method to be called is determined at runtime based on the actual type of the object, not the reference type.

Example:

class Animal

{ void sound()

{

System.out.println("Animals make sound");

}

}

class Dog extends Animal

{ void sound()

{

System.out.println("Dog barks");

}

}

class Cat extends Animal

{ void sound()

{

System.out.println("Cat meows");

}

}

public class DynamicBindingExample

{ public static void main(String[] args)

{

Animal animal; // Reference of type Animal

animal = new Dog(); // Object of type Dog animal.sound(); // Calls Dog's sound method (runtime decision)

animal = new Cat(); // Object of type Cat animal.sound(); // Calls Cat's sound method (runtime decision)

}

}

# Super Keyword and Method Hiding

**Answer:** The super keyword in Java refers to the **immediate parent class** of the current object. It is commonly used in inheritance to:

1. Access parent class **methods** or **variables** when they are hidden by subclass implementations.
2. Invoke the **parent class constructor** explicitly.

**Method Hiding** occurs when a **static method** in a subclass has the **same signature** as a static method in its parent class. Instead of overriding, the method in the subclass hides the method in the parent class. The method called is determined at **compile-time** based on the reference type

=============================================================

========

=====================================================================

=====================================================================

==========================

===========================================