1. Inheritance in Java: It's a mechanism in which one class inherits the properties and behaviors (methods) of another class. It promotes code reusability and establishes a relationship between classes.

2. Superclass and Subclass: A superclass is the class being inherited from, also known as a parent class. A subclass is the class that inherits from the superclass, also known as a child class. The subclass inherits fields and methods from its superclass.

3. Implementation of Inheritance in Java: In Java, inheritance is achieved using the `extends` keyword. A subclass extends a superclass to inherit its properties and behaviours.

A class from where a subclass inherits features is called superclass. It is also called base class or parent class.

4. Polymorphism: it is a greek word poly means many and morphism means form, It's the ability of a method to take on multiple forms. In Java, it can be achieved through method overloading (compile-time polymorphism) and method overriding (runtime polymorphism).

5. Method Overloading vs. Method Overriding:

- Overloading: Same method name with different parameters within the same class.

- Overriding: A method in a subclass with the same name, return type, and parameters as a method in its superclass. The method in the subclass provides its own implementation.

6. Abstraction with Example: Abstraction is hiding the implementation details and showing only essential features of an object. For instance, a car's driver interacts with the gear lever without knowing its internal mechanisms.

7. Abstract vs. Final Methods:

- Abstract Method: Defined in an abstract class but doesn’t have an implementation. It must be overridden in a subclass.

- Final Method: It cannot be overridden by subclasses, ensuring the method's implementation remains constant.

Example:

java

abstract class Shape {

// Abstract method without implementation

public abstract void draw();

// Final method that cannot be overridden

public final void display() {

// Some code here

}

}

8. \*Final Class in Java:\* A final class is a class that cannot be subclassed. It prevents other classes from extending it.

9. \*Abstraction vs. Encapsulation:\*

- Abstraction: Focuses on hiding the implementation details and showing only necessary features.

- Encapsulation: Wraps data and methods that operate on the data into a single unit (class), preventing direct access to data from outside.

10. \*Runtime vs. Compile-time Polymorphism:\*

- Runtime Polymorphism: Occurs during runtime and is achieved through method overriding.

- Compile-time Polymorphism: Occurs at compile time and is achieved through method overloading.

Example:

java

class Animal {

void sound() {

System.out.println("Animal makes a sound");

}

}

class Dog extends Animal {

void sound() {

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

}

}

Animal a = new Dog(); // Runtime Polymorphism

a.sound(); // Output: "Dog barks"