1. Inheritance in Java:

Inheritance is a fundamental concept in Java that allows a class to inherit the properties and behaviors (fields and methods) of another class. It facilitates code reuse and the creation of a hierarchy of classes. The class that inherits is called a subclass or derived class, while the class being inherited from is called a superclass or base class.

2. Superclass and Subclass:

The superclass (or base class) is the class that is being inherited from. It contains the common attributes and methods shared by its subclasses. Subclasses inherit these attributes and methods from the superclass and can also have their own additional attributes and methods.

3. Implementation of Inheritance in Java:

Inheritance is achieved in Java by using the "extends" keyword. You can create a subclass that extends a superclass. For example:

java

class Superclass {

// Superclass members

}

class Subclass extends Superclass {

// Subclass members

}

4. Polymorphism:

Polymorphism is a concept in Java that allows objects of different classes to be treated as objects of a common superclass. It enables the ability to use a single interface to represent various types of objects. Polymorphism can be achieved through method overriding and method overloading.

5. Method Overloading vs. Method Overriding:

- Method overloading is when a class has multiple methods with the same name but different parameter lists. It is determined at compile-time.

- Method overriding is when a subclass provides a specific implementation of a method that is already defined in its superclass. It is determined at runtime and involves the "@Override" annotation.

6. Abstraction with Example:

Abstraction is the process of hiding complex implementation details and showing only the necessary features of an object. For example, in Java, an abstract class can be used to define a blueprint for other classes to inherit from, providing abstract methods that must be implemented by its subclasses.

java

abstract class Shape {

abstract void draw();

}

class Circle extends Shape {

void draw() {

// Implementation for drawing a circle

}

}

7. Abstract Method vs. Final Method:

- An abstract method is declared in an abstract class and does not have an implementation. Subclasses are required to provide an implementation for abstract methods.

- A final method is a method that cannot be overridden by subclasses.

Example of an abstract method:

java

abstract class Shape {

abstract void draw();

}

Example of a final method:

java

class Parent {

final void show() {

// Final method implementation

}

}

8. Final Class in Java:

A final class in Java is a class that cannot be subclassed. It means that you cannot create a class that extends a final class. Final classes are often used when you want to prevent any further extension or modification of a class.

Example

final class ParentClass

{

void showData()

{

System.out.println("This is a method of final Parent class");

}

}

//It will throw compilation error

class ChildClass extends ParentClass

{

void showData()

{

System.out.println("This is a method of Child class");

}

}

class MainClass

{

public static void main(String arg[])

{

ParentClass obj = new ChildClass();

obj.showData();

}

}

9. Abstraction vs. Encapsulation:

- Abstraction is about hiding the complexity of an object by providing a simplified view. It focuses on what an object does.

- Encapsulation is about hiding the internal details of an object and restricting access to its data. It focuses on how an object achieves its functionality.

10. Runtime vs. Compile-time Polymorphism:

- Runtime polymorphism is achieved through method overriding and is resolved at runtime. It allows a subclass to provide a specific implementation of a method defined in its superclass.

- Compile-time polymorphism is achieved through method overloading and is resolved at compile-time. It allows a class to have multiple methods with the same name but different parameter lists, and the appropriate method is determined at compile-time based on the method arguments.