03 (OOP Principles)

Date
Page No.

(1) Inheritance

- It is a mechanism in Java by which one class acquires (inherits) the properties and behaviours (~~[illegible]~~ fields and methods) of another class. It helps in code reusability and method overriding.

- Superclass (Parent / Base class): The class whose properties are inherited
- Subclass (Child / Derived class): The class that inherits the superclass.

- Syntax :-

```
class Superclass {
    //fields & methods
}
class Subclass extends Superclass {
    //additional fields and methods.
}
```

(i) Private $\overset{\text{keyword}}{\wedge}$ in Inheritance

⇒ In Java, the private access modifier makes a class member accessible only within the class it is declared. When a subclass inherits ~~any~~ from a superclass, any private fields or methods in the class are?

(a) Not accessible in the subclass,
(b) Not visible in the subclass,
(c) And technically not Inherited

Access is only possible through public or protected

- or from getter & setters.

1.
2.
3

Date
Page No.

(ii) Super keyword

⇒ Super is a reference keyword used within a subclass to refer to its immediate parent class. (Calls above one constructor while using many constructors at a time).

It used to :-

~~(a) Call the~~

Use case	purpose	Syntax
1. Call ~~par~~ parent constructor	To initialize parent class fields	super();
2. Call parent method	To invoke parent version of overriden method	super.methodName();
3. Access parent variable	If subclass has a variable with same name.	super.variableName;

- Rules of super
  (a) super() must be the first statement in a constructor.
  (b) It only refers to the immediate parent.
  (c) Cannot be used in static methods.
  (d) If parent constructor needs parameter, you must call super(param1, param2 . ~).

(iii) Types of Inheritance

(a) Single Inheritance
→ In single inheritance, a subclass interits from one superclass only. This is the most basic form of inheritance. It forms a one-level hieracy - used to extend or customize functionality of a single parent class.

Date
Page No.

Syntax :-

```
Class parent {
    // parent class members
}
class Child extends Parent {
    //child class members
    }
```

b) Multilevel Inheritance

→ In multilevel inheritance, a class inherits from a class which itself inherits from another class. It creates a chain of Inheritance.

Syntax :-

```
Class GrandParent {
    //level 1
    }
Class Parent extends Grandparent {
    // level 2
    }
Class Child extends Parent {
    //level 3
    }
```

Date
Page No.

c) Hierarchical Inheritance

⇒ In hierarchical inheritance, multiple subclasses inherit from a single superclass. Each subclass has an independent relationship with the parent, but not with each other.

Syntax:

```
Class Parent {
    // shared base class.
}
class ChildA extends Parent {
    //subclass A
}
Class ChildB extends Parent {
    //subclass B
}
```

(d) Multiple Inheritance (X X)

⇒ When a class tries to inherit from more than one class.

⇒ It is Not Supported in Java (via classes)
↓

Because java avoids ambiguity caused by the Diamond problem - when two super classes have the same method and subclass inherits from both.

Solution

⇒ Use interface to acheive multiple Inheritance.

(e) Hybrid Inheritance (X X)

⇒ It is a combination of more than one type of Inheritance (e.g. → multiple + multi level).

⇒ It is not Supported in Java (via classes).

↓

Because it depends on multiple Inheritance, which is not allowed through classes.

<u>Solution</u>

• Hybrid inheritance Can be acheived using a mix of classes + interfaces.