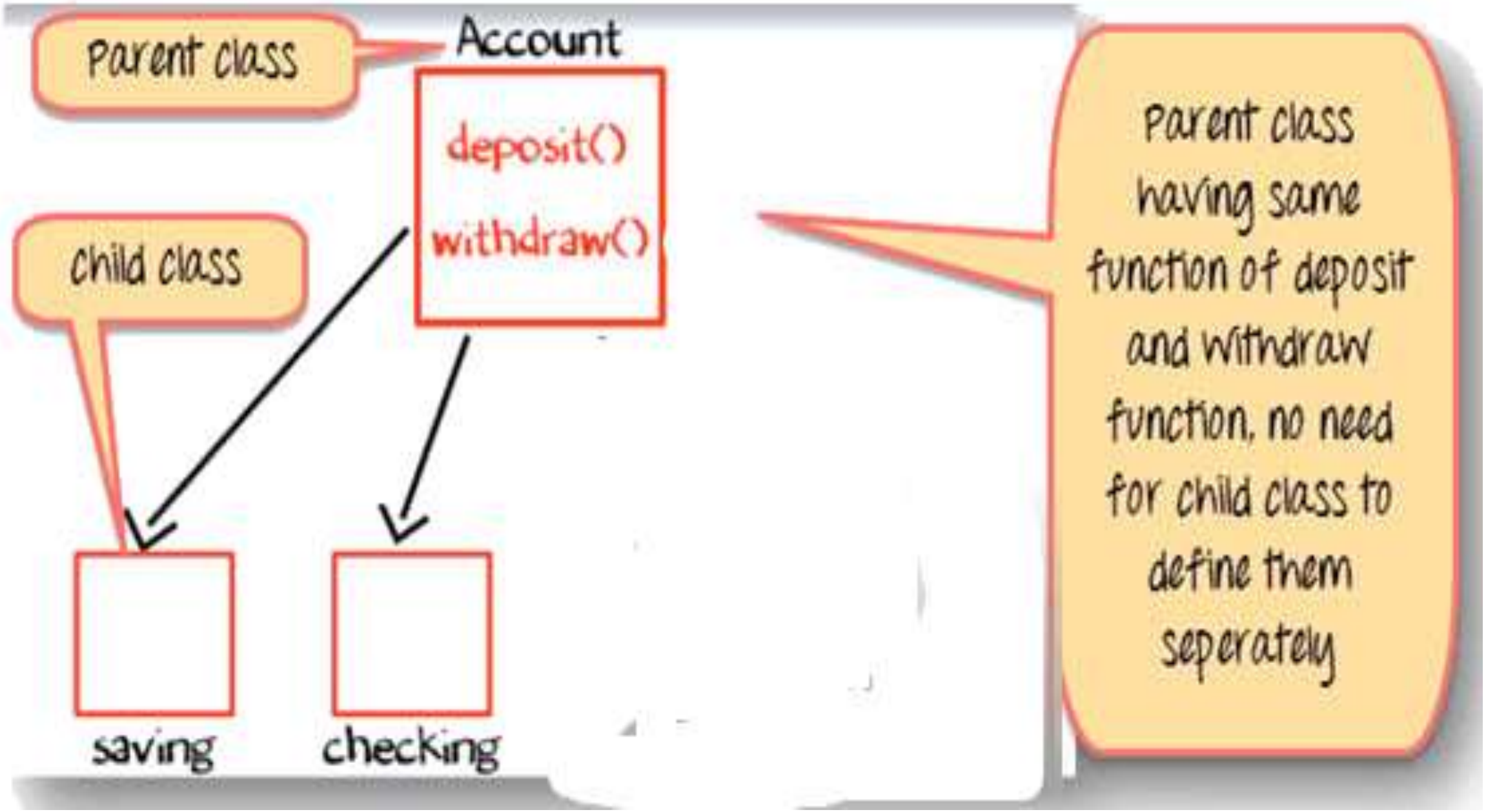
Day-8

- Agenda:-
  - Inheritance & Polymorphism
  - Abstract Class , Interfaces

Inheritance & Polymorphism

- Polymorphism in Java occurs when there are one or more classes or objects related to each other by inheritance.
- It is the ability of an object to take many forms. Inheritance lets users inherit attributes and methods, and polymorphism uses these methods to perform different tasks.
- Polymorphism means "many forms", and it occurs when we have many classes that are related to each other by inheritance.
- Inheritance lets us inherit attributes and methods from another class. Polymorphism uses those methods to perform different tasks. This allows us to perform a single action in different ways.
- Inheritance is one in which a new class is created that inherits the properties of the already exist class. It supports the concept of code reusability and reduces the length of the code in object-oriented programming.


Parent class
Account
deposit()
withdraw()
child class
saving
checking
Parent class having same function of deposit and withdraw function, no need for child class to define them seperately

- The idea behind inheritance in Java is that you can create new classes that are built upon existing classes. When you inherit from an existing class, you can reuse methods and fields of the parent class. Moreover, you can add new methods and fields in your current class also.

- Inheritance represents the **IS-A relationship** which is also known as a *parent-child* relationship.

- Why use inheritance in java
  - For Method Overriding (so runtime polymorphism can be achieved).
  - For Code Reusability.

➢Terms used in Inheritance

- **Class:** A class is a group of objects which have common properties. It is a template or blueprint from which objects are created.

- **Sub Class/Child Class:** Subclass is a class which inherits the other class. It is also called a derived class, extended class, or child class.

- **Super Class/Parent Class:** Superclass is the class from where a subclass inherits the features. It is also called a base class or a parent class.

- **Reusability:** As the name specifies, reusability is a mechanism which facilitates you to reuse the fields and methods of the existing class when you create a new class. You can use the same fields and methods already defined in the previous class.

The syntax of Java Inheritance

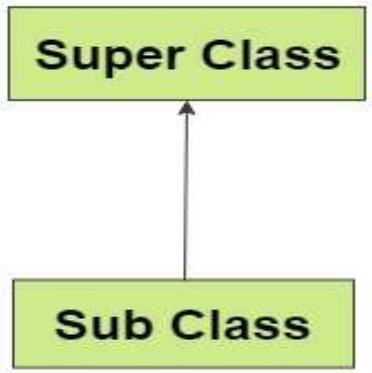
```
class Subclass-name extends Superclass-name
{
   //methods and fields
}
```

- The **extends keyword** indicates that you are making a new class that derives from an existing class. The meaning of "extends" is to increase the functionality.
- In the terminology of Java, a class which is inherited is called a parent or superclass, and the new class is called child or subclass.

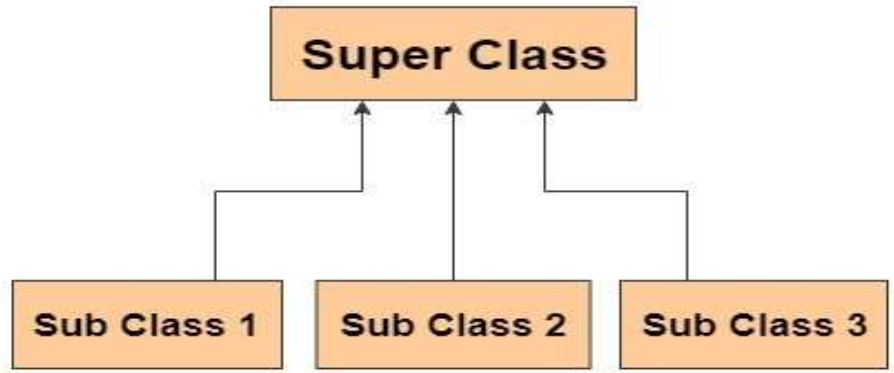
- Types of Inheritance are:

1. Single inheritance
2. Multi-level inheritance
3. Multiple inheritance
4. Hybrid inheritance
5. Hierarchical inheritance

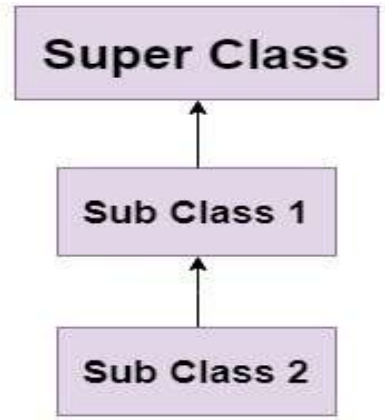
Single Inheritance



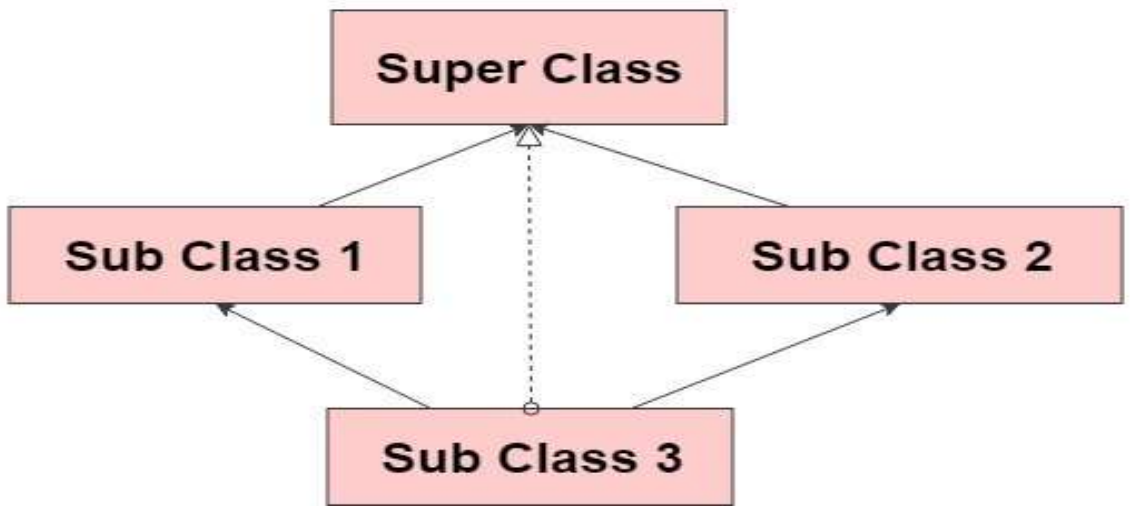
Hierarchial Inheritance



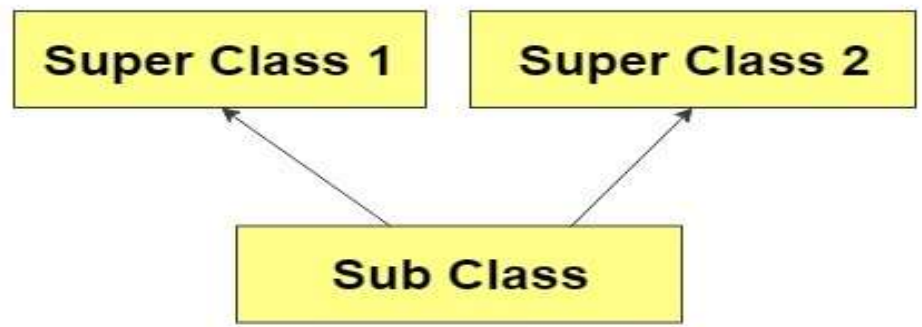
MultiLevel Inheritance



Hybrid Inheritance



Multiple Inhertance



Single inheritance

- In single inheritance, a single subclass extends from a single superclass.
- refers to a child and parent class relationship where a class extends the another class.

```
class Super {
   .....
}
class Sub extends Super {
   .....
}
```

Multi-level inheritance

- When there is a chain of inheritance, it is known as *multilevel inheritance*.

```
class Shape {
  public void display() {
    System.out.println("Inside display");
  }
}
class Rectangle extends Shape {
  public void area() {
    System.out.println("Inside area");
  }
}
class Cube extends Rectangle {
  public void volume() {
    System.out.println("Inside volume");
  }
}
```

Hierarchical Inheritance

- When two or more classes inherits a single class, it is known as hierarchical inheritance.

- In Hierarchical Inheritance, one class serves as a superclass (base class) for more than one subclass. In the below image, class A serves as a base class for the derived class B, C and D.

- Example:-

```
class A {
    public void print_A() { System.out.println("Class A"); }
}
class B extends A {
    public void print_B() { System.out.println("Class B"); }
}
class C extends A {
    public void print_C() { System.out.println("Class C"); }
}
class D extends A {
    public void print_D() { System.out.println("Class D"); }
}
```

Multiple Inheritance

- In Multiple inheritances, one class can have more than one superclass and inherit features from all parent classes.

- Please note that Java does not support multiple inheritances with classes.

- In java, we can achieve multiple inheritances only through Interfaces.

hybrid inheritance

- It is a mix of two or more of the above types of inheritance. Since java doesn’t support multiple inheritances with classes, hybrid inheritance is also not possible with classes.
- In java, we can achieve hybrid inheritance only through Interfaces.

Super Keyword

- The super keyword is similar to "this" keyword.
- The keyword super can be used to access any data member or methods of the parent class.
- Super keyword can be used at variable, method and constructor level.
- Syntax:

```
super.<method-name>();
```

Abstract Classes

- A class which is declared with the abstract keyword is known as an abstract class in Java. It can have abstract and non-abstract methods (method with the body).
- Abstraction is a process of hiding the implementation details and showing only functionality to the user.
- Another way, it shows only essential things to the user and hides the internal details, for example, sending SMS where you type the text and send the message. You don't know the internal processing about the message delivery.
- Abstraction lets you focus on what the object does instead of how it does it.
- Ways to achieve Abstraction

  There are two ways to achieve abstraction in java

  - Abstract class (0 to 100%)
  - Interface (100%)

- An abstract class must be declared with an abstract keyword.
- It can have abstract and non-abstract methods.
- It cannot be instantiated.
- It can have constructors and static methods also.
- It can have final methods which will force the subclass not to change the body of the method.

Example:-

```
abstract class Bike{
  abstract void run();
}

class Honda4 extends Bike{
void run(){System.out.println("running safely");
}
}

public static void main(String args[]){
 Bike obj = new Honda4();
 obj.run();
}
```

Interface

- An interface in Java is a blueprint of a class. It has static constants and abstract methods.

- The interface in Java is a mechanism to achieve abstraction. There can be only abstract methods in the Java interface, not method body. It is used to achieve abstraction and multiple inheritance in Java.

- In other words, you can say that interfaces can have abstract methods and variables. It cannot have a method body.

- Syntax :

```
interface <interface_name> {

    // declare constant fields
    // declare methods that abstract
    // by default.
}
```