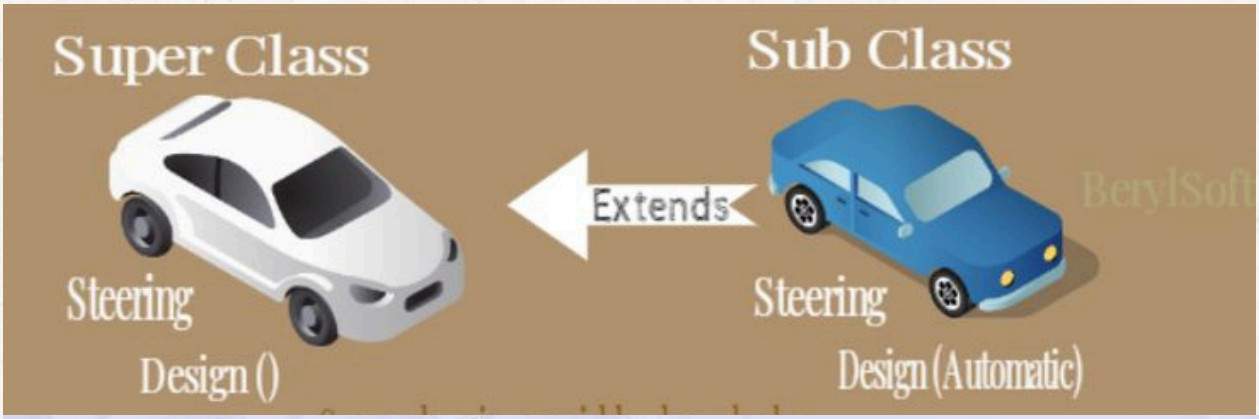
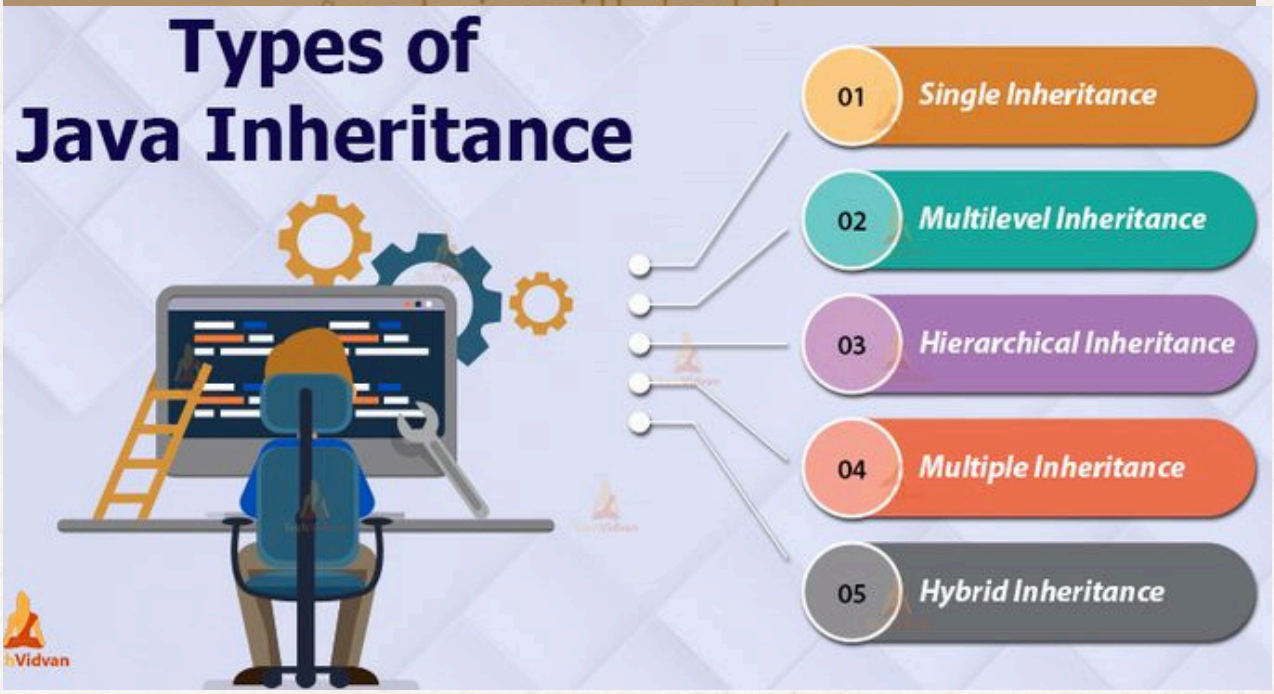
FAZAIA BILQUIS COLLEGE
NUR KHAN BASE,RWP





Lab #04 Types of Inheritance

SUBMITTED BY: *MARIA ATTA*

ROLL NO: *BSCS-13-F24-01*

SUBJECT: *OOP'S*

LAB 5: CONCEPT OF INHERITANCE

- **INHERITANCE:**

  A way for one class (child) to copy features like variables and methods from another class (parent). It saves time by reusing code.

- **TYPES OF INHERITANCE:**

  1. ***SINGLE INHERITANCE:***
     - One child class copies from one parent class (e.g., "Dog" from "Animal")

  2. MULTILEVEL INHERITANCE:
     - A chain of classes, like grandparent → parent → child (e.g., "Animal" → "Dog" → "Puppy").

  3. HIERARCHICAL INHERITANCE:
     - One parent class shares with many child classes (e.g., "Person" → "Student" and "Teacher").

- **PARENT CLASS AND CHILD CLASS:**
  - **Parent Class:** The main class that shares its features (also called "superclass").
  - **Child Class:** The class that takes features from the parent using extends (also called "subclass").

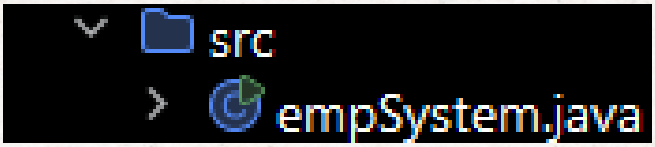
- **THE SUPER KEYWORD:**
  - Let the child class call the parent's constructor or methods to avoid repeating code.

- **CONSTRUCTOR:**
  - A special method that sets up a class's variables when you create an object (e.g., setting a name or salary).

- **OBJECTS:**
  - Real examples of a class, created using the class blueprint (e.g., an employee named "Ahmed").
- **METHODS:**
  - Actions a class can do, like calculating a salary or printing a message.
- **ATTRIBUTES (VARIABLES)**
  - Information a class stores, like a name or price.
- **CODE REUSABILITY**
  - Using the same code in many places without rewriting it, is made possible by inheritance.
- **MAINTAINABILITY**
  - How easy it is to update code. Inheritance helps by letting changes in the parent affect all children automatically.

PROGRAM 1: SINGLE INHERITANCE - EMPLOYEE SALARY CALCULATION

Problem: Calculate an employee's annual salary and total salary with a bonus using a parent Employee class and a child PermanentEmployee class with super.



- ## Program:

empSystem.java

```java
class Employee {  1 usage  1 inheritor
    String name;  1 usage
    int id;  1 usage
    double basicSalary;  3 usages
    Employee(String name, int id, double basicSalary) {  1 usage
        this.name = name;
        this.id = id;
        this.basicSalary = basicSalary;  }
    double calculateAnnualSalary() {  1 usage
        return basicSalary * 12;  }
}
class PermanentEmployee extends Employee {  2 usages
    double bonus;  2 usages
    PermanentEmployee(String name, int id, double basicSalary, double bonus) {  1 usage
        super(name, id, basicSalary);
        this.bonus = bonus;  }
    double calculateTotalSalary() {  1 usage
        return (basicSalary * 12) + bonus; }
}
public class empSystem {
    public static void main(String[] args) {
        PermanentEmployee emp1 = new PermanentEmployee( name: "Ahmed",  id: 101,  basicSalary: 50000,  bonus: 10000);
        System.out.println("Annual Salary: " + emp1.calculateAnnualSalary());
        System.out.println("Total Salary with Bonus: " + emp1.calculateTotalSalary());
    }
}
```

- ## Output:

```
src
  empSystem.java
"C:\Program Files\Java\jdk-23\bin\java.exe"
Annual Salary: 600000.0
Total Salary with Bonus: 610000.0
```

- ## Code Explanation:

1. **Task:** Calculate an employee's yearly salary and total salary with a bonus.
2. **How I Did It:** I made a parent-class Employee with name, ID, and salary, plus a method for yearly salary (***salary × 12***). The child class PermanentEmployee copies from it, add a bonus, and calculates the total (yearly salary + bonus). Used Super to set up the parent's stuff.

PROGRAM 2: MULTILEVEL INHERITANCE - GROCERY BILL CALCULATION CODE

Problem: Calculate the price of a grocery item with weight and discount using Item, WeighedItem, and DiscountedWeighedItem classes in a chain with super.

- **Program:**

GroceryBillingSystem.java

```java
class Item{ // Parent Class  1 usage  2 inheritors
    String name;  2 usages
    double pricePerKg;  3 usages
    //Constructor
    Item ( String name, double pricePerKg){  1 usage
        this.name = name;
        this.pricePerKg = pricePerKg;
    }
    double getPricePerKg(){  1 usage
        return pricePerKg;
    }
}
class WeighedItem extends Item{ //Child Class  1 usage  1 inheritor
    double weight;  2 usages
    // constructor
    WeighedItem(String name, double pricePerKg, double weight){  1 usage
        super(name,pricePerKg);
        this.weight = weight;
    }
    double getTotalPrice() {  2 usages
        return pricePerKg * weight;
    }
}
class DiscountedWeighedItem extends WeighedItem {  2 usages
    double discountPercentage;  2 usages
    DiscountedWeighedItem(String name, double pricePerKg, double weight, double discountPercentage) {  1 usage
        super(name, pricePerKg, weight);
        this.discountPercentage = discountPercentage;
    }
    double getFinalPrice() {  1 usage
        double totalPrice = getTotalPrice();
        double discount = (totalPrice * discountPercentage) / 100;
        return totalPrice - discount;
    }
}
public class GroceryBillingSystem {
    public static void main(String[] args) {
        DiscountedWeighedItem apples = new DiscountedWeighedItem( name: "Apples", pricePerKg: 100, weight: 2, discountPercentage: 10);
        System.out.println("Price per kg of " + apples.name + ": " + apples.getPricePerKg());
        System.out.println("Total Price before discount: " + apples.getTotalPrice());
        System.out.println("Final Payable Amount after discount: " + apples.getFinalPrice());
    }
}
```

- **Output:**

```
"C:\Program Files\Java\jdk-23\bin\java.exe" "-javaagent:C:
Price per kg of Apples: 100.0
Total Price before discount: 200.0
Final Payable Amount after discount: 180.0
```

- ## Code Explanation:

1. **Task:** Figure out the price of grocery items with weight and a discount.
2. **How I Did It:** Created Item with name and price per kg, then WeighedItem added weight and multiplied for total price. DiscountedWeighedItem adds a discount and subtracts it from the total. Used super to link everything up.

PROGRAM 3: HIERARCHICAL INHERITANCE - SCHOOL ATTENDANCE SYSTEM

Problem: Track attendance for students and teachers using a Person class with Student and Teacher child classes and super.

- ## Package 1: university Student

SchoolSystem.java

```java
class Person{ // Parent Class  2 usages  2 inheritors
    String name;  4 usages
    int ID;  2 usages
    Person(String name, int ID){ //Constructor  2 usages
        this.name = name;
        this. ID = ID;
    }
    void displayInfo(){  2 usages
        System.out.println("Name : " + name);
        System.out.println("ID : " + ID);
    }
}
class Student extends Person{ // Child Class  2 usages
    int gradeLevel;  2 usages
    Student(String name, int ID,int gradeLevel){ //Constructor  1 usage
        super(name,ID);
        this.gradeLevel = gradeLevel;
```

```java
    }
    void markAttendance() {  1 usage
        System.out.println(name + " (Student - Grade " + gradeLevel + ") is present.");
    }
}
class Teacher extends Person{  2 usages
    String subject;  2 usages
    Teacher(String name, int ID, String subject){  1 usage
        super(name,ID);
        this.subject = subject;
    }
    void markAttendance() {  1 usage
        System.out.println(name + " (Teacher - " + subject + ") is present.");
    }
}
public class SchoolSystem {
    public static void main(String[] args) {
        Student student1 = new Student( name: "Ali",  ID: 101,  gradeLevel: 10);
        student1.displayInfo();
        student1.markAttendance();
        System.out.println();
        Teacher teacher1 = new Teacher( name: "Ms. Ayesha",  ID: 201,  subject: "Mathematics");
        teacher1.displayInfo();
        teacher1.markAttendance();
    }
}
```

- ***Output:***

```
"C:\Program Files\Java\jdk-23\bin\java.exe" "-javaagent:C:
Name : Ali
ID : 101
Ali (Student - Grade 10) is present.

Name : Ms. Ayesha
ID : 201
Ms. Ayesha (Teacher - Mathematics) is present.
```

- ***Code Explanation:***

1. **Task:** Track attendance for students and teachers.
2. **How I Did It:** Made Person with name and ID for both. Student adds grade and marks attendance, Teacher adds subject and does the same. Both copy from Person with extends, and super sets up the shared parts.

HOME TASKS:

SCENARIO 1: SINGLE INHERITANCE - LIBRARY MANAGEMENT

Problem: Make a Book class and a Member class that inherits from it to show borrowed book details using super.

- ***Program:***

LibrarySystem.java

```java
class Books {  1 usage  1 inheritor
    String title;  2 usages
    String author;  2 usages
    int bookID;  2 usages
    Books(String title, String author,int bookID){  // Constructor  1 usage
        this.title = title;
        this.author = author;
        this.bookID = bookID; }
}
class Member extends Books {  2 usages
    String name;  2 usages
    String date;  2 usages
    Member(String title, String author,int bookID,String name, String date){ // Constructor  1 usage
        super(title, author, bookID);
        this.name = name;
        this.date = date; }
    void displayDetail(){ //  function to display detail  1 usage
        System.out.println("Book : " + title + ", Author : " + author + ", ID:" + bookID + ".");
        System.out.println("Borrowed By: " + name + ", on : " + date + ".");
    }
}
public class LibrarySystem {
    public static void main(String[]args){
        Member m1 = new Member( title: "OOPs in Java", author: "John Doe", bookID: 102, name: "Maria Atta", date: "2025-03-12");
        m1.displayDetail();
    }
}
```

- ***Output:***

```
"C:\Program Files\Java\jdk-23\bin\java.exe" "-javaagent:C:
Book : OOPs in Java, Author : John Doe, ID:102.
Borrowed By: Maria Atta, on : 2025-03-12.
```

- ***Code Explanation:***

1. **Task:** Show who borrowed a book.
2. **How:** Book has a title, author, and ID. Member inherits it, adds name and date, and uses super to set book details. A method shows everything.

SCENARIO 2: MULTILEVEL INHERITANCE - VEHICLE REGISTRATION SYSTEM

Problem: Create Vehicle, ElectricVehicle, and FastChargingEV classes in a chain to show vehicle details using super.

- ***Program:***

VehicleSystem.java

```java
class Vehicle {  1 usage  2 inheritors
    String make;  2 usages
    String model;  2 usages
    int registrationNo;  2 usages
    Vehicle(String make, String model,int registrationNo){  1 usage
        this.make = make;
        this.model = model;
        this.registrationNo = registrationNo;
    }
    void display() {  1 usage
        System.out.println("Make: " + make + ", Model: " + model + ", Reg: " + registrationNo);
    }
}
class ElectricVehicle extends Vehicle{  1 usage  1 inheritor
    int batteryCapacity;  2 usages
    ElectricVehicle(String make,String model, int registrationNo,int batteryCapacity){  1 usage
        super(make, model, registrationNo);
        this.batteryCapacity = batteryCapacity;
    }
    void displayEV() {  1 usage
        display();
        System.out.println("Battery: " + batteryCapacity + " kWh");
    }
}
class FastChargingEV extends ElectricVehicle{  2 usages
    int chargingSpeed;  2 usages
    FastChargingEV(String make, String model,int registrationNo,int batteryCapacity,int chargingSpeed){  1 usage
        super(make,model,registrationNo,batteryCapacity);
        this.chargingSpeed = chargingSpeed;
    }
    void displayAll() {  1 usage
        displayEV();
        System.out.println("Charging Speed: " + chargingSpeed + " kW");
    }
}
public class VehicleSystem {
    public static void main(String[]args){
        FastChargingEV EV1 = new FastChargingEV( make: "Tesla", model: "123", registrationNo: 456, batteryCapacity: 45, chargingSpeed: 60);
        EV1.displayAll();
    }
}
```

- ***Output:***

```
"C:\Program Files\Java\jdk-23\bin\java.exe"
Make: Tesla, Model: 123, Reg: 456
Battery: 45 kWh
Charging Speed: 60 kW
```

- ***Code Explanation:***

1. **Task:** Show details of a fast-charging electric vehicle.
2. **How:** Vehicle has basic info, ElectricVehicle adds battery, and FastChargingEV adds charging speed. Each uses super to pass data up the chain. Methods show all details step-by-step.

SCENARIO 2: HIERARCHICAL INHERITANCE - BANK ACCOUNT SYSTEM

Problem: Use Account with SavingsAccount and CurrentAccount children to manage accounts with super.

- ***Program:***

BankSystem.java

```
class Account {  2 usages  1 inheritor
    int AccountNo;  3 usages
    int balance;  4 usages
    Account(int AccountNo, int balance) { // Constructor  2 usages
        this.AccountNo = AccountNo;
        this.balance = balance;
    }
}
class SavingsAccount extends Account {  2 usages
    double interestRate;  2 usages
    SavingsAccount(int AccountNo, int balance, double interestRate) {  1 usage
        super(AccountNo, balance);
        this.interestRate = interestRate;
    }
    double calculateInterest() {  1 usage
        return balance * (interestRate / 100);
    }
}
class CurrentAccount extends Account {  2 usages
```

```java
    double overDraftLimit;  2 usages
    CurrentAccount(int AccountNo, int balance, double overDraftLimit) {  1 usage
        super(AccountNo, balance);
        this.overDraftLimit = overDraftLimit;
    }
    void displayLimit() {  1 usage
        System.out.println("Overdraft Limit: " + overDraftLimit);
    }
}
public class BankSystem {
    public static void main(String[] args) {
        SavingsAccount s1 = new SavingsAccount( AccountNo: 1001,  balance: 5000,  interestRate: 5);
        System.out.println("Account " + s1.AccountNo + ": Balance = " + s1.balance);
        System.out.println("Interest: " + s1.calculateInterest());
        CurrentAccount c1 = new CurrentAccount( AccountNo: 1002,  balance: 3000,  overDraftLimit: 1000);
        System.out.println("Account " + c1.AccountNo + ": Balance = " + c1.balance);
        c1.displayLimit();
    }
}
```

- ## Program:

```
"C:\Program Files\Java\jdk-23\bin\java.exe"
Account 1001: Balance = 5000
Interest: 250.0
Account 1002: Balance = 3000
Overdraft Limit: 1000.0
```

- ## Code Explanation:

1. **Task:** Manage savings and current accounts.
2. **How:** Account has number and balance. SavingsAccount adds interest and calculates it, CurrentAccount adds overdraft limit and shows it. Both use super to set parent data.