1. **SINGLE INHERITANCE:**

CODE:

class College{

Course:string

constructor(course:string){

this.Course = course;

}

}

class Francis extends College{

Year:number

constructor(course:string, year:number){

super(course);

this.Year=year;

}

displayDetails():void{

console.log("Course: " + this.Course);

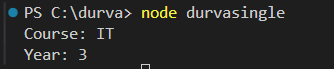
console.log("Year: " + this.Year);

}

}

let obj = new Francis("IT",3);

obj.displayDetails();

OUTPUT:  ****

1. **MULTIPLE INHERITANCE**CODE:

**interface Person {**

**name: string;**

**dob: string;**

**}**

**interface AcademicDetails {**

**grade: string;**

**}**

**interface ContactDetails {**

**subject: string;**

**}**

**interface Student extends Person, AcademicDetails, ContactDetails {**

**studentId: number;**

**}**

**let studentObject = <Student>{};**

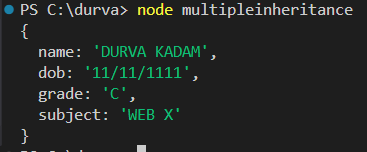
**studentObject.name = "DURVA KADAM";**

**studentObject.dob = "11/11/1111";**

**studentObject.grade = "C";**

**studentObject.subject = "WEB X";**

**console.log(studentObject);**

**OUTPUT: **

1. **MULTILEVEL INHERITANCE**

CODE:hehehe

class BankAccount {

balance: number;

constructor(initialBalance: number) {

this.balance = initialBalance;

}

deposit(amount: number): void {

this.balance += amount;

console.log(`Deposited ₹${amount}. New Balance: ₹${this.balance}`);

}

}

class SavingsAccount extends BankAccount {

interestRate: number;

constructor(initialBalance: number, interestRate: number) {

super(initialBalance);

this.interestRate = interestRate;

}

addInterest(): void {

let interest = (this.balance \* this.interestRate) / 100;

this.balance += interest;

console.log(`Interest Added: ₹${interest}. Updated Balance: ₹${this.balance}`);

}

}

class PremiumSavingsAccount extends SavingsAccount {

rewardPoints: number;

constructor(initialBalance: number, interestRate: number) {

super(initialBalance, interestRate);

this.rewardPoints = 0;

}

earnRewards(transactionAmount: number): void {

let points = Math.floor(transactionAmount / 100);

this.rewardPoints += points;

console.log(`Earned ${points} reward points. Total: ${this.rewardPoints}`);

}

}

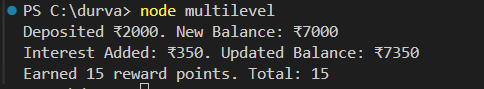
let myAccount = new PremiumSavingsAccount(5000, 5);

myAccount.deposit(2000);

myAccount.addInterest();

myAccount.earnRewards(1500);

OUTPUT:



1. **ACCESS MODIFIER**

CODE:

class ShadyGovDatabase {

public citizenRecords: string[];

protected classifiedFiles: string[]; // Only for government officials

private blackOpsSecrets: string[]; // Even most officials don’t know!

constructor() {

this.citizenRecords = ["John Doe: Clean Record", "Jane Smith: Tax Evader"];

this.classifiedFiles = ["UFO Sightings", "Aliens Exist!"];

this.blackOpsSecrets = ["Area 51 Plans", "Moon Landing Script 🤫"];

}

public accessPublicRecords(): void {

console.log("📂 Citizen Records (Public):", this.citizenRecords.join(", "));

}

protected accessClassifiedFiles(): void {

console.log("🔒 Accessing Classified Files:", this.classifiedFiles.join(", "));

}

private accessBlackOpsSecrets(): void {

console.log("🚫TOP SECRET! Only a few know:", this.blackOpsSecrets.join(", "));

}

}

// Subclass: Government Official

class CorruptOfficial extends ShadyGovDatabase {

public leakClassifiedData(): void {

console.log("🕵️‍♂️ I'm an official, I have access to classified files!");

this.accessClassifiedFiles(); // Allowed (protected)

}

}

// Subclass: Hacker

class Hacker {

public hackDatabase(target: ShadyGovDatabase): void {

console.log("👨‍💻 Hacking Attempt...");

target.accessPublicRecords(); // ✅ Allowed (public)

// target.accessClassifiedFiles(); // ❌ Error! Protected

// target.accessBlackOpsSecrets(); // ❌ Error! Private

}

}

// Testing the system

let citizen = new ShadyGovDatabase();

citizen.accessPublicRecords(); // ✅ Allowed

// citizen.accessClassifiedFiles(); // ❌ Error! Protected

// citizen.accessBlackOpsSecrets(); // ❌ Error! Private

let corruptOfficial = new CorruptOfficial();

corruptOfficial.leakClassifiedData(); // ✅ Allowed (protected access inside subclass)

let hacker = new Hacker();

hacker.hackDatabase(citizen); // Can only see public data

OUTPUT:

