Class & Object

1. Create a simple Person class with properties name and age. Create an object of this class and print the values.
2. Write a class Car with properties brand and model. Add a method displayDetails() to print car details.
3. Create a Student class with a constructor that takes name and grade. Instantiate two student objects and print their details.
4. Define a class Rectangle with properties width and height. Add a method to calculate the area of the rectangle.
5. Write a JavaScript class Animal with a speak() method. Create an object and call the method.
6. Create a Car class with properties like brand, model, and year. Create 2 objects from this class and display their details.
7. Design a Book class with properties title, author, and pages. Add a method read() that logs "Reading {title} by {author}". Create a few book objects and call their methods.
8. Build a Student class with properties name, age, and grade. Add a method getDetails() that returns all the student's information.
9. Create a Pen class with properties brand, color, and type (gel/ballpoint). Add a method write() that logs a writing message.
10. Design a TV class with brand, size, and isSmart. Add methods to turnOn() and turnOff().
11. Create a Pet class with name, species, and age. Add a method makeSound() that logs different sounds based on species.
12. Make a Recipe class with name, ingredients, and cookTime. Add a method displayRecipe()

Encapsulation

1. Create a BankAccount class with accountNumber and a private property balance. Add methods deposit(), withdraw(), and checkBalance().
2. Implement a PasswordManager class where the password is private. Provide methods to setPassword() and verifyPassword(inputPassword) securely.
3. Create a User class with private properties like email and password. Only allow access through methods (getter and setter methods).
4. Create a Door class with a private field isLocked. Provide methods to lock() and unlock(). Prevent direct access to isLocked.
5. Design a SecretNote class with private content. Allow only a method read(secretCode) to read it if the correct code is entered.
6. Make a VotingMachine class where votes are private. Only a public method castVote(candidateName) and showResults(adminPassword) can interact with votes.

Inheritance

1. Create a Person class with properties name and age. Then create a Teacher class that inherits from Person and adds a subject property and a method teach().

2. Create a Shape class with a method calculateArea(). Then create Rectangle and Circle classes that inherit from Shape and override the calculateArea() method.
3. Design a Vehicle class with common properties like speed. Create Bike and Truck classes that extend Vehicle and add their specific properties/methods.
4. Create an Appliance class with properties like brand and power. Create WashingMachine and Microwave classes extending Appliance.
5. Design a SportsPlayer class with name and team. Create Cricketer and Footballer classes extending it, each with their own method (bat() or kick()).
6. Build a Notification base class with method send(). Create EmailNotification, SMSNotification, and PushNotification subclasses overriding send() differently.
7. Create a Gadget class with model and price. Inherit a Smartphone class that adds cameraQuality and a takePhoto() method.

Advance

1. Create a Library class that holds a collection of Book objects. Implement methods like addBook(book), removeBook(title), and findBook(title).
2. Create a Product class with name, price, and private stockQuantity. Add methods to purchase(quantity) and restock(quantity). Inherit a PerishableProduct class from Product and add expiryDate.
3. Design a GameCharacter class with name, health, and strength. Then create Warrior and Mage subclasses that have additional abilities like attack() and castSpell().
4. Employee Management System
    - Task: Create an Employee class with properties like name, id, and department.
    - Encapsulation: Make the salary property private.
    - Methods: Implement methods to setSalary() and getSalary().
    - Inheritance: Create a subclass Manager that adds a teamSize property and a method getTeamSize().
5. Online Course Platform
    - Task: Design a Course class with properties title, instructor, and duration.
    - Encapsulation: Keep the enrolledStudents list private.
    - Methods: Add methods to enrollStudent(studentName) and getEnrolledStudents().
    - Inheritance: Create a subclass PaidCourse that includes a price property and a method applyDiscount(discountPercentage).
6. Library Catalog System
    - Task: Implement a LibraryItem class with properties title and publicationYear.
    - Inheritance: Create subclasses Book and Magazine. Book should have an author property, while Magazine should have an issueNumber.
    - Methods: Each subclass should have a method getDetails() that returns all relevant information.
7. E-Commerce Shopping Cart
    - Task: Create a Product class with properties name, price, and quantity.
    - Encapsulation: Make the quantity property private.
    - Methods: Implement methods to addStock(amount) and purchase(amount).
    - Inheritance: Develop a subclass DigitalProduct that overrides the purchase() method to handle license keys instead of stock.
8. Banking Application
    - Task: Design a BankAccount class with properties accountNumber and accountHolder.

- Encapsulation: Keep the balance property private.
- Methods: Include methods deposit(amount), withdraw(amount), and getBalance().
- Inheritance: Create subclasses SavingsAccount and CheckingAccount, each with specific rules for withdrawals and deposits.

9. University Enrollment System
    - Task: Implement a Person class with properties name and email.
    - Inheritance: Create subclasses Student and Professor. Student should have a studentID and a list of courses, while Professor should have an employeeID and a list of subjects.
    - Encapsulation: Keep the lists of courses and subjects private.
    - Methods: Add methods to addCourse(courseName) for students and addSubject(subjectName) for professors.
10. Create a Zoo system where you have a Zoo class that manages a collection of Animal objects. Implement methods like addAnimal(), feedAnimals(), and viewAllAnimals().
11. Build a FlightBooking system where Flight objects and Passenger objects interact. Each Passenger can book a Flight.
12. Design a Hospital Management system: Doctor, Patient, and Appointment classes. Allow booking and cancelling appointments.
13. Create an OnlineStore simulation with Product and Cart classes. Users can add products to the cart, remove products, and checkout.
14. Make a ChatApplication system with User and Message classes. Each User can send and receive messages.
15. Implement an abstract Shape class (cannot be directly instantiated) with method area(). Create Square and Triangle that extend it.
16. Create a SubscriptionPlan system with plans like Free, Standard, and Premium using inheritance. Each plan has different limitations.
17. Design a SchoolManagementSystem where there are Person → Student, Teacher, and Staff classes, all inheriting from Person.
18. Create a TaskManager app where each Task has name, deadline, and priority. Include methods to markComplete() and reschedule().