OOPs Assignments

What I, Abhishek Ingle,understood about the pillars of OOP through these assignments :

Q1). Bank Management.

I have three class and their respective methods::

Bank Class:

- Manages a list of accounts.
- Provides methods to add accounts, get account balance, and transfer money between accounts.

Account Class:

- Has attributes like accountId, accountHolderName, and balance.
- Provides methods to deposit money, withdraw money , and get account information , getAccountHolderName(), getBalance().

BankManagement Class:

- Contains the main() method.
- Creates a Bank object and adds accounts to it.
- Deposits and withdraws money from accounts.
- Transfers money between accounts.
- Checks account balances.

Here we do Encapsulation where our data Data (like account details) and methods (like deposit and withdraw) are encapsulated within appropriate classes (Bank and Account).

Q2). Instrument Management.

Instrument Class:

- Acts as a base class for Piano, Flute, and Guitar.
- Provides a method play() which will be overridden by subclasses.

Piano Class:

- Subclass of Instrument.
- Overrides the play() method to play a specific sound for a piano.

Flute Class:

- Subclass of Instrument.
- Overrides the play() method to play a specific sound for a flute.

Guitar Class:

- Subclass of Instrument.
- Overrides the play() method to play a specific sound for a guitar.

Main Class:

- Contains the main() method to test the instrument array.
- Creates an array of 10 Instrument objects.
- Assigns different types of instruments to Instrument references.
- Demonstrates polymorphic behavior by calling the play() method for each instrument, resulting in different sounds based on the instrument type.).Instrument.

We utilize inheritance and polymorphism to establish a **'has-a'** relationship between the base class Instrument and its subclasses (Piano, Flute, Guitar). The base class Instrument encapsulates common behavior shared by all musical instruments, allowing for code reusability and organization. Subclasses override the play() method to provide specific implementations, demonstrating polymorphic behavior. Additionally, the Main class showcases different methods being overridden, illustrating the flexibility and extensibility provided by inheritance and polymorphism.

Q4). Employee.

Full Time Employees:
They work full-time and have a fixed salary along with benefits like health insurance.
We can find out their salary.
We can also see their details, including their name, ID, salary, and benefits.

Part Time Employees:
They work part-time and are paid based on the hours they work and their hourly rate.
We can find out their salary too.
We can also see their details, including their name, ID, hourly rate, and hours worked.

Test Employee:
In the main part of the program, we create examples of both full-time and part-time employees.
We then show their details and calculate their salaries, demonstrating how each type of employee works differently.

Here I used

- Polymorphism is demonstrated through method overriding.
- I have used inheritance to create subclasses (FullTimeEmployee and PartTimeEmployee) that inherit attributes and methods from a superclass (Employee).
- Encapsulation is evident in how data like attributes like name, id, salary, benefits, hourlyRate, hoursWorked and methods like displayDetails() and calculateSalary() are encapsulated within their respective classes Employee, FullTimeEmployee and PartTimeEmployee.