Lab 07 Tasks

Task 01

Imagine developing a comprehensive banking system simulator for a prominent financial institution. In this system, a base class called **Account** encapsulates essential data members such as *accountNumber*, *balance*, *accountHolderName*, and optionally *accountType*. It offers member functions like **deposit(amount)** to add funds, **withdraw(amount)** to remove funds with proper error checking, **calculateInterest()** to compute interest based on varying rules, **printStatement()** to output detailed transaction histories, and **getAccountInfo()** to retrieve general account details.

Derived classes like **SavingsAccount**, **CheckingAccount**, and **FixedDepositAccount** extend this structure by incorporating specialized data members—such as *interestRate* and *minimumBalance* for **SavingsAccount** or *maturityDate* and *fixedInterestRate* for **FixedDepositAccount**. These derived classes override key functions like **calculateInterest()** and **printStatement()** to provide account-type-specific functionalities. Additionally, the **withdraw()** function is overridden in specific accounts to apply different transaction rules, ensuring that each account type follows realistic banking policies while maintaining polymorphic behavior.

Task 02

Picture an innovative design tool aimed at architects and graphic designers that allows for creating, manipulating, and analyzing various geometric shapes. The system is structured around a **Shape** class, which includes data members such as *position*, *color*, and an optional *borderThickness*. It provides functions like **draw()** for rendering, **calculateArea()** for area measurement, and **calculatePerimeter()** for perimeter computation.

Derived classes such as **Circle, Rectangle, Triangle, and Polygon** introduce their own properties—for example, a **Circle** includes *radius* and *center position*, while a **Rectangle** includes *width*, *height*, and *top-left corner position*. Each derived class overrides **draw()**, **calculateArea()**, and **calculatePerimeter()** to handle their respective geometries.

Task 03

Envision creating a robust application for a global finance firm that needs to handle and compute multiple currencies with real-time conversion capabilities. This system is built on a base class called **Currency**, which contains core data members such as *amount*, *currencyCode*, *currencySymbol*, and an optional *exchangeRate*. It provides functions like **convertToBase()** for converting the amount into a common base currency, **convertTo(targetCurrency)** for converting between currencies, and **displayCurrency()** for showcasing currency details.

Derived classes like **Dollar, Euro, and Rupee** extend this foundation by introducing currency-specific details and overriding **convertToBase()** and **displayCurrency()** to reflect the exchange rates and regional currency formats dynamically.

Task 04

Imagine designing a digital management system for a large university that seamlessly integrates academic and administrative operations. At the heart of this system is a **Person** class that stores universal data members such as *name*, *id*, *address*, *phoneNumber*, and *email*. It provides functions like **displayInfo()** to show personal details and **updateInfo()** to modify them.

Derived classes—**Student, Professor, and Staff**—extend this system by introducing specific data members:

- **Student** includes *coursesEnrolled*, *GPA*, and *enrollmentYear*, modifying **displayInfo()** to show academic records.
- **Professor** has *department*, *coursesTaught*, *and salary*, updating **displayInfo()** to display faculty-specific details.
- **Staff** maintains *department*, *position*, *and salary*, tailoring **displayInfo()** to reflect administrative roles.

An additional **Course** class, with attributes like *courseId*, *courseName*, *credits*, *instructor*, and *schedule*, provides functions such as **registerStudent(student)** and **calculateGrades()**. This ensures an interactive system where function overrides enable dynamic role-based management.

Task 05

Visualize a digital multimedia library management system designed to catalog and circulate a diverse range of media items, including books, DVDs, CDs, and magazines. The **Media** class serves as the base, encapsulating core data members such as *title*, *publicationDate*, *uniqueID*, and *publisher*. It provides functions like **displayInfo()** to output media details, **checkOut()** to process lending, and **returnItem()** to handle returns.

Derived classes—**Book, DVD, CD, and Magazine**—introduce additional attributes:

- **Book** includes *author*, *ISBN*, *and numberOfPages*, modifying **displayInfo()** for books.
- **DVD** incorporates *director*, *duration*, *and rating*, updating **displayInfo()** for movie details.
- **CD** includes *artist*, *numberOfTracks*, *and genre*, adapting **displayInfo()** for music albums.

The system supports **function overloading** for searching media by different attributes (e.g., title, author, or publication year).