Lab 09 Tasks

Task 01

Problem: A vehicle rental company is expanding its services to include cars, bikes, and future vehicle types. They need a flexible system to manage daily rates and display vehicle details without exposing internal calculations. The goal is to ensure new vehicle types (e.g., scooters, trucks) can be added seamlessly while maintaining a consistent interface for customers to view rental options.

Classes and Structure:

- **Abstract Class** Vehicle:
  - **Data Members**:
    - model (string): Stores the vehicle's model name.
    - rate (double): Stores the daily rental rate.
  - **Member Functions**:
    - getDailyRate(): Pure virtual function to return the daily rate.
    - displayDetails(): Pure virtual function to show model and rate.
- **Derived Class** Car:
  - Inherits model and rate from Vehicle.
  - Overrides getDailyRate() and displayDetails() to provide car-specific behavior.
- **Derived Class** Bike:
  - Inherits model and rate from Vehicle.
  - Overrides getDailyRate() and displayDetails() to provide bike-specific behavior.

Flow:

- Create Car and Bike objects.
- Use polymorphism to call displayDetails() and getDailyRate() for each vehicle.

Task 02

Problem: A homeowner wants to integrate smart lights and thermostats from different brands into a single app. The system must provide a unified way to turn devices on/off, adjust settings, and check statuses without requiring users to interact with brand-specific interfaces. Future devices (e.g., smart locks) should integrate without altering the core system.
Classes and Structure:

- **Abstract Class** SmartDevice:
  - **Member Functions**:

        - turnOn(): Pure virtual function to activate the device.
        - turnOff(): Pure virtual function to deactivate the device.
        - getStatus(): Pure virtual function to return the device's current state.
- **Derived Class** LightBulb:
    - **Data Members**:
        - isOn (bool): Tracks if the light is on/off.
        - brightness (int): Stores brightness level (0-100%).
    - Implements turnOn(), turnOff(), and getStatus() for light control.
- **Derived Class** Thermostat:
    - **Data Members**:
        - isOn (bool): Tracks if the thermostat is active.
        - temperature (double): Stores the current temperature setting.
    - Implements turnOn(), turnOff(), and getStatus() for temperature control.

Flow:

- Create LightBulb and Thermostat objects.
- Turn devices on/off and display their statuses.

Task 03

Problem: A public library is transitioning from manual record-keeping to a digital system. Staff need secure access to book details (title, author, ISBN) but must not modify records directly. The system should separate public interfaces (e.g., searching books) from internal data handling to prevent accidental data corruption.

Classes and Structure:

- **Class** Book (Header: Book.h, Implementation: Book.cpp):
    - **Data Members**:
        - title, author, ISBN (strings): Store book metadata.
    - **Member Functions**:
        - Constructor to initialize book details.
        - getTitle(), getAuthor(), getISBN(): Return book properties.
- **Class** Library (Optional for advanced students):
    - **Data Members**:
        - A collection (e.g., array/list) of Book objects.
    - **Member Functions**:
        - addBook(), removeBook(), searchBook(): Manage the collection.

Flow:

- Define Book in header and source files to separate interface and implementation.
- In main(), create a Book object and display its metadata.

Task 04

Problem: An e-commerce platform aims to support multiple payment methods (credit cards, digital wallets) to attract global customers. The payment system must validate transactions securely and allow new payment options (e.g., cryptocurrency) to be added without rewriting existing code.

Classes and Structure:

- **Abstract Class** PaymentMethod:
  - **Member Function**:
    - processPayment(double amount): Pure virtual function to handle transactions.
- **Derived Class** CreditCard:
  - **Data Members**:
    - cardNumber (string): Stores the card number.
  - Implements processPayment() to validate and process payments.
- **Derived Class** DigitalWallet:
  - **Data Members**:
    - balance (double): Tracks available funds.
  - Implements processPayment() to deduct balance.

Flow:

- Create CreditCard and DigitalWallet objects.
- Simulate payments and check for success/failure.

Task 05

Problem: A fitness app wants to help users track calories burned during activities like running and cycling. Each activity requires unique calculations (e.g., distance vs. speed), but the app must present results consistently. The system should allow adding new activities (e.g., swimming) in the future.

Classes and Structure:

- **Abstract Class** Activity:
  - **Member Function**:
    - calculateCaloriesBurned(): Pure virtual function to compute calories.
- **Derived Class** Running:
  - **Data Members**:
    - distance (double): Stores distance in kilometers.
    - time (double): Stores duration in minutes.
  - Implements calculateCaloriesBurned() for running.
- **Derived Class** Cycling:

- **Data Members**:
    - speed (double): Stores speed in km/h.
    - time (double): Stores duration in hours.
- Implements calculateCaloriesBurned() for cycling.

Flow:

- Create Running and Cycling objects.
- Display calories burned for each activity.