**OOP PROJECT**

**Group Members:**

Umar Ahmad Cheema 70148628

Hafiz Muhammad Haris 70143269

Mehsum Khan 70149243

Musa Pervaiz 70142400

**Vehicle Parking Management System**

**Project Overview**

The **Vehicle Parking System** is a command-line based application developed using C++. This system allows users (mainly administrators) to perform various operations related to parking vehicles, including parking, viewing vehicle details, and removing vehicles. The project leverages object-oriented programming (OOP) principles such as inheritance, abstraction, and polymorphism to structure the system efficiently.

**Features**

* **Login Authentication**: Admin must enter a password to access the system.
* **Vehicle Parking**: Admin can park a car or a bike, entering details like driver name, registration number, company and model, and duration of stay.
* **View Vehicle Details**: Admin can search for a vehicle by registration number to view its details.
* **Remove Vehicle**: Admin can remove a vehicle from the system by providing the vehicle's registration number.
* **Expenses Calculation**: The system calculates parking fees based on time of stay.
* **File Persistence**: Vehicle data (car and bike details) are stored and retrieved from a text file for persistence.

Below is the detailed explanation of code, organized under various Object-Oriented Programming (OOP) principles.

**1. Abstraction**

Abstraction is demonstrated by hiding implementation details and exposing only essential functionality.

* **Class Login**:
  + An abstract class that defines the pure virtual function authenticate(). The actual implementation is provided in the AdminLogin class.
* **Class ParkingManagement**:
  + Another abstract class with pure virtual functions: parkVehicle(), viewVehicleDetails(), and removeVehicle(). These are implemented in the ParkingSystem class.

**2. Inheritance**

Inheritance allows one class to acquire properties and methods from another class.

* **AdminLogin inherits from Login**:
  + Implements the authenticate() method to handle admin authentication.
* **ParkingSystem inherits from ParkingManagement**:
  + Provides implementations for vehicle parking, viewing, and removal.

**3. Encapsulation**

Encapsulation restricts direct access to certain components of an object, ensuring better data security.

* **Private members**:
  + Sensitive information like the password in AdminLogin and the vehicle details in the Vehicle class are private or protected, accessed and modified through public methods.
* **Getter and Setter Methods**:
  + Vehicle::setVehicleDetails() and Vehicle::getRegistrationNumber() encapsulate data access and modifications.

**4. Polymorphism**

Polymorphism allows using the same interface for different underlying forms (data types or classes).

* **Runtime Polymorphism**:
  + Achieved through overriding the authenticate() method in AdminLogin and the virtual methods in ParkingManagement in the ParkingSystem class.

**5. File Handling**

The program interacts with external files to store and retrieve data persistently.

* **File Operations in ParkingSystem**:
  + loadVehiclesFromFile() reads vehicle data from a file and initializes Vehicle objects.
  + saveVehiclesToFile() writes the current state of vehicles to a file.
* **Serialization and Deserialization**:
  + Vehicle::toFileString() and Vehicle::fromFileString() handle the conversion between Vehicle objects and their file representations.

**6. Exception Handling**

Exception handling ensures that the program handles errors gracefully.

* **File-related Exceptions**:
  + Both loadVehiclesFromFile() and saveVehiclesToFile() use try-catch blocks to handle file errors, such as failure to open the file.

**7. Classes and Objects**

The program defines multiple classes to represent different entities.

* **Classes**:
  + Login, AdminLogin, Vehicle, ParkingManagement, ParkingSystem, and VehicleParkingSystem.
* **Objects**:
  + Instances of AdminLogin, ParkingSystem, and Vehicle are created and used to manage authentication, vehicle data, and parking operations.

**8. Input/Output Handling**

The program handles user input and provides console output.

* **User Authentication**:
  + The \_getch() function is used in AdminLogin::authenticate() to securely input a password without displaying it on the console.
* **Menu Navigation**:
  + Displays options and takes user input to navigate through different functionalities.

**9. Program Structure**

The VehicleParkingSystem class combines all components and manages the program's flow:

* **Authentication**:
  + Calls AdminLogin::authenticate() for user validation.
* **Parking Operations**:
  + Delegates tasks like parking, viewing, and removing vehicles to the ParkingSystem class.

**10. Reusability and Modularity**

The code is modular and promotes reusability:

* **Reusability**:
  + Abstract classes (Login, ParkingManagement) provide reusable interfaces.
* **Modularity**:
  + Each functionality (e.g., authentication, vehicle management, file handling) is encapsulated in a separate class or method.

**11. User Interface Design**

Although basic, the program provides a structured console-based interface:

* **Separation of Concerns**:
  + displayLoginPage(), displayCarParkingMenu(), and printExitMessage() manage user interface elements.

This code implements a **Vehicle Parking Management System** using Object-Oriented Programming (OOP) principles in C++. Here's a detailed explanation of each component:

### ****Key Functionalities****

1. **Admin Login System**
   * Ensures only authorized users can access the system.
2. **Vehicle Parking System**
   * Allows users to park vehicles, view details, and remove vehicles.
3. **File Handling**
   * Saves and retrieves vehicle details from a file (vehicles.txt).

**Code Components**

*1. Global Helper Functions*

* printSeparator(): Prints a separator line for formatting.
* printHeader(): Displays the main header of the program.
* displayLoginPage(): Displays the login page UI.
* displayCarParkingMenu(): Displays options for the parking system.
* printExitMessage(): Displays a goodbye message.

*2. Login System*

* **Class Login (Abstract Base Class)**:
  + Contains a pure virtual function authenticate() that ensures any derived class must implement authentication logic.
* **Class AdminLogin (Derived from Login)**:
  + Implements authenticate() to verify the admin's password.
  + **Password Input**:
    - Uses \_getch() to read input without displaying characters directly on the console.
    - Replaces characters with asterisks (\*) for security.
  + Compares the input password to the predefined password (1234).

*3. Vehicle Class*

* Represents a vehicle with attributes:
  + driverName: Name of the vehicle driver.
  + registrationNumber: Unique identifier for the vehicle.
  + companyAndModel: Vehicle manufacturer and model.
  + timeStay: Hours the vehicle is parked.
  + parkingFee: Calculated based on timeStay.
* **Methods**:
  + setVehicleDetails(): Sets vehicle attributes.
  + displayVehicleDetails(): Prints vehicle information.
  + clearDetails(): Resets the attributes of the vehicle.
  + toFileString() & fromFileString():
    - Convert vehicle data to/from a file-friendly string format for saving/loading vehicle data.

*4. Parking Management*

* **Class ParkingManagement (Abstract Base Class)**:
  + Declares virtual methods:
    - parkVehicle()
    - viewVehicleDetails()
    - removeVehicle()
  + Enforces derived classes to implement these functionalities.
* **Class ParkingSystem (Derived from ParkingManagement)**:
  + Manages the parking of cars and bikes.
  + Maintains two arrays: cars[10] and bikes[10] for storing vehicles.
  + **File Handling**:
    - loadVehiclesFromFile(): Loads vehicle data from vehicles.txt.
    - saveVehiclesToFile(): Saves updated vehicle data to vehicles.txt.
  + **Vehicle Operations**:
    - parkVehicle(): Adds a new vehicle to the system and calculates the parking fee based on timeStay and vehicle type (Car: Rs. 100/hour, Bike: Rs. 50/hour).
    - viewVehicleDetails(): Searches for a vehicle by registration number and displays its details.
    - removeVehicle(): Removes a vehicle by registration number.

*5. VehicleParkingSystem Class*

* Encapsulates the entire system:
  + Combines AdminLogin and ParkingSystem classes.
  + **Workflow**:
    1. Authenticates the admin using AdminLogin.
    2. Displays the menu using displayCarParkingMenu().
    3. Executes menu options (e.g., parking, viewing, or removing vehicles).

*6. Main Function*

* Creates an instance of VehicleParkingSystem.
* Calls run() to start the application.

**Code Flow**

1. **Login Authentication**:
   * Admin enters a password to access the system.
2. **Menu Options**:
   * Displayed after successful login.
3. **Parking Operations**:
   * Park a car/bike, view vehicle details, remove a vehicle, or exit.
4. **File Handling**:
   * Data persists through file storage (vehicles.txt).