Day 99/180 Object-Oriented Programming

3 Programming Challenges with Classes:

1. Bank Customer Class:

Challenge: Design a `Customer` class for a bank system that manages customer information and basic operations.

Attributes:

- `name`**: String containing the customer's full name.**
- `accountNumber`**: Unique integer identifying the customer's account.**
- `accountBalance`**: Double representing the current balance in the account.**
- `isActive`**: Boolean indicating whether the account is active.**

Methods:

- `deposit(amount)`**: Adds the specified amount to the account balance.**
- `withdraw(amount)`**: Deducts the specified amount from the account balance (check for sufficient funds).**
- `transfer(amount, targetAccount)`**: Transfers the specified amount to another customer's account (within the system).**
- `printDetails()`**: Prints the customer's name, account number, and current balance.**

Program :

```cpp
#include <iostream>
#include <string>

using namespace std;

class Customer {
private:
    string name;
    int accountNumber;
    double accountBalance;
    bool isActive;

public:
    // Constructor
    Customer(string customerName, int customerAccountNumber, double initialBalance)
    {
        name = customerName;
        accountNumber = customerAccountNumber;
        accountBalance = initialBalance;
        isActive = true;
    }


    // Deposit method
    void deposit(double amount) {
        if (amount > 0) {
            accountBalance += amount;
            cout << "Deposited $" << amount << " into account " << accountNumber << endl;
        } else {
            cout << "Invalid deposit amount" << endl;
        }
    }

    // Withdraw method
    void withdraw(double amount) {
        if (isActive && amount > 0 && accountBalance >= amount) {
            accountBalance -= amount;
            cout << "Withdrawn $" << amount << " from account " << accountNumber << endl;
        } else {
            cout << "Invalid withdrawal or insufficient funds" << endl;
        }
    }

    // Transfer method
    void transfer(double amount, Customer& targetAccount) {
        if (isActive && amount > 0 && accountBalance >= amount) {
            accountBalance -= amount;
            targetAccount.deposit(amount);
            cout << "Transferred $" << amount << " from account " << accountNumber
                 << " to account " << targetAccount.getAccountNumber() << endl;
        } else {
            cout << "Invalid transfer or insufficient funds" << endl;
        }
    }

    // Print customer details method
    void printDetails() const {
        cout << "Customer Name: " << name << endl;
        cout << "Account Number: " << accountNumber << endl;
        cout << "Account Balance: $" << accountBalance << endl;
    }

    // Getter for account number
    int getAccountNumber() const {
        return accountNumber;
    }
};

int main() {
    // Example usage of the Customer class
    Customer customer1("John Doe", 12345, 1000.0);
    Customer customer2("Jane Doe", 67890, 1500.0);

    customer1.printDetails();
    cout << endl;

    customer1.deposit(500.0);
    customer1.printDetails();
    cout << endl;

    customer1.withdraw(200.0);
    customer1.printDetails();
    cout << endl;

    customer1.transfer(300.0, customer2);
    customer1.printDetails();
    customer2.printDetails();

    return 0;
}
```

2. Car Class:

Challenge: Create a `Car` class that simulates the behavior of a vehicle.

Attributes:

- `model`: **String representing the car model name.**
- `year`: **Integer indicating the car's manufacturing year.**
- `fuelLevel`: **Double representing the remaining fuel quantity (percentage or liters).**
- `speed`: **Integer representing the current speed in kilometers per hour.**
- `isRunning`: **Boolean indicating whether the car is currently running.**

Methods:

- `startEngine()`: **Sets `isRunning` to true and prints a starting message.**
- `stopEngine()`: **Sets `isRunning` to false and prints a stopping message.**
- `accelerate(amount)`: **Increases the car's speed by the specified amount (check engine state and fuel level).**
- `brake(amount)`: **Decreases the car's speed by the specified amount (ensure speed doesn't become negative).**
- `refuel(amount)`: **Increases the fuel level by the specified amount (check for tank capacity).**
- `printStatus()`: **Displays the car's model, speed, fuel level, and running state.**

Program :

```
#include <bits/stdc++.h>

#include <iostream>
#include <string>
```

```cpp
using namespace std;



class Car {
private:
    string model;
    int year;
    double fuelLevel;
    int speed;
    bool isRunning;

public:
    Car(string carModel, int carYear, double initialFuelLevel)
    {
        model = move(carModel);
        year = carYear;
        fuelLevel = initialFuelLevel;
        speed = 0;
        isRunning = false;
    }

    // Method to start the engine
    void startEngine() {
        if (!isRunning && fuelLevel > 0) {
            isRunning = true;
            cout << "Engine started. Ready to go!" << endl;
        } else {
            cout << "Cannot start the engine. Check fuel level or engine state." <<
endl;
        }
    }

    // Method to stop the engine
    void stopEngine() {
        if (isRunning) {
            isRunning = false;
            cout << "Engine stopped. Have a great day!" << endl;
        } else {
            cout << "Engine is already stopped." << endl;
        }
    }

    // Method to accelerate
    void accelerate(int amount) {
        if (isRunning && fuelLevel > 0) {
            speed += amount;
            cout << "Accelerated to " << speed << " km/h." << endl;
        } else {
            cout << "Cannot accelerate. Check fuel level or engine state." << endl;
        }
    }

    // Method to brake
    void brake(int amount) {
        if (isRunning) {
            speed -= amount;
            if (speed < 0) {
                speed = 0;
            }
            cout << "Braked to " << speed << " km/h." << endl;
        } else {
            cout << "Cannot brake. Engine is not running." << endl;
        }
    }

    // Method to refuel
    void refuel(double amount) {
        // Assume the tank capacity is 100 liters for simplicity
        const double tankCapacity = 100.0;
        if (fuelLevel + amount <= tankCapacity) {
            fuelLevel += amount;
            cout << "Refueled. Current fuel level: " << fuelLevel << " liters." <<
endl;
        } else {
            cout << "Cannot refuel. Tank capacity exceeded." << endl;
        }
    }

    // Method to print car status
    void printStatus() const {
        cout << "Car Model: " << model << endl;
        cout << "Manufacturing Year: " << year << endl;
        cout << "Current Speed: " << speed << " km/h" << endl;
        cout << "Fuel Level: " << fuelLevel << " liters" << endl;
        cout << "Engine State: " << (isRunning ? "Running" : "Stopped") << endl;
    }
};

int main() {
    // Example usage of the Car class
    Car myCar("Toyota Camry", 2022, 50.0);
    myCar.printStatus();
    cout << endl;

    myCar.startEngine();
    myCar.accelerate(30);
    myCar.brake(10);
    myCar.printStatus();
    cout << endl;

    myCar.refuel(20.0);
    myCar.stopEngine();
    myCar.printStatus();

    return 0;
}
```

3. Laptop Class:

Challenge: Design a `Laptop` class that represents a portable computer system.

Attributes:

- `brand`**: String representing the laptop brand and model.**
- `screenSize`**: Double indicating the screen size in inches.**
- `processor`**: String specifying the processor type and speed.**
- `ram`**: Integer representing the available RAM capacity in gigabytes.**
- `storage`**: Integer representing the storage capacity in gigabytes.**
- `batteryLevel`**: Double showing the remaining battery percentage.**
- `isOn`**: Boolean indicating whether the laptop is currently powered on.**

Methods:

- `powerOn()`**: Sets** `isOn` **to true and prints a startup message.**

- `powerOff()`: Sets `isOn` to false and prints a shutdown message.
- `openApps(numApps)`: Simulates opening a specified number of applications, potentially impacting battery life.
- `closeApps(numApps)`: Simulates closing applications, restoring battery life.
- `charge(amount)`: Increases the battery level by the specified amount (check for maximum capacity).
- `printSpecs()`: Displays the laptop's brand, screen size, processor, RAM, storage, and battery level.

Program:

```cpp
#include <iostream>
#include <string>

using namespace std;

class Laptop {
private:
    string brand;
    double screenSize;
    string processor;
    int ram;
    int storage;
    double batteryLevel;
    bool isOn;

public:
    // Parametrized constructor
   Laptop(string laptopBrand, double laptopScreenSize, string laptopProcessor,
           int laptopRAM, int laptopStorage)
    {
        brand = laptopBrand;
        screenSize = laptopScreenSize;
        processor = laptopProcessor;
        ram = laptopRAM;
        storage = laptopStorage;
        batteryLevel = 100.0;
        isOn = false;
    }

    // Method to power on the laptop
    void powerOn() {
        if (!isOn) {
            isOn = true;
            cout << "Laptop powered on. Welcome!" << endl;
        } else {
            cout << "Laptop is already powered on." << endl;
        }
    }

    // Method to power off the laptop
    void powerOff() {
        if (isOn) {
            isOn = false;
            cout << "Laptop powered off. Goodbye!" << endl;
        } else {
            cout << "Laptop is already powered off." << endl;
        }
    }

    // Method to simulate opening applications
    void openApps(int numApps) {
        if (isOn && batteryLevel > 0) {
            cout << "Opened " << numApps << " applications. Battery life may
be impacted." << endl;
        } else {
            cout << "Cannot open applications. Laptop is off or out of
battery." << endl;
        }
    }

    // Method to simulate closing applications
    void closeApps(int numApps) {
        if (isOn) {
            cout << "Closed " << numApps << " applications. Battery life
restored." << endl;
        } else {
            cout << "Cannot close applications. Laptop is off." << endl;
        }
    }
```

```cpp
    // Method to charge the laptop battery
    void charge(double amount) {
        const double maxBatteryCapacity = 100.0;
        if (isOn && batteryLevel + amount <= maxBatteryCapacity) {
            batteryLevel += amount;
            cout << "Charged laptop battery. Current battery level: " <<
batteryLevel << "%" << endl;
        } else {
            cout << "Cannot charge. Laptop is off or maximum battery capacity
reached." << endl;
        }
    }

    // Method to print laptop specifications
    void printSpecs() const {
        cout << "Laptop Brand: " << brand << endl;
        cout << "Screen Size: " << screenSize << " inches" << endl;
        cout << "Processor: " << processor << endl;
        cout << "RAM: " << ram << " GB" << endl;
        cout << "Storage: " << storage << " GB" << endl;
        cout << "Battery Level: " << batteryLevel << "%" << endl;
    }
};

int main() {
    // Example usage of the Laptop class
    Laptop myLaptop("Dell XPS", 13.3, "Intel Core i7", 16, 512);

    myLaptop.printSpecs();
    cout << endl;

    myLaptop.powerOn();
    myLaptop.openApps(5);
    myLaptop.charge(20.0);
    myLaptop.printSpecs();
    cout << endl;

    myLaptop.powerOff();
    myLaptop.printSpecs();

    return 0;
}
```