23CS24C – OBJECT ORIENTED PROGRAMMING IN C++

Experiential Learning

AIRLINE PASSENGER AND LUGGAGE MANAGEMENT SYSTEM

Model Lab – 1 Report

Submitted By: M.Shanmugapriya

Team No:10

Team Members: R.Jerina Gnana Pushpa

P.M.Mathujaa

Scenario:Design and implement a Airline Passenger and Luggage Management System application with the following features:

a. Create a base class Passenger that represents a passenger in the system. It should have

attributes like passengerID, name, age, gender, nationality, and ticketNumber. Implement constructors, accessors, and mutators as needed.

b. Create derived classes EconomyPassenger and BusinessPassenger that inherit from the Passenger class. Each derived class should have its own unique attributes and methods. For example, an EconomyPassenger may have additional attributes like seat Number, mealPreference, and baggage Allowance, while a BusinessPassenger may include attributes like priorityBoarding, lounge Access, and additional Services.

c. Implement dynamic binding and polymorphism by defining virtual functions in the base class and overriding them in the derived classes. For example, create a virtual function displayDetails() that displays the passenger details based on their type (economy or business class).

d. Create a class Luggage to represent luggage information. It should have attributes like luggagelD, passengerlD (to associate luggage with passengers), weight, size, and contents. Implement appropriate constructors, accessors, and mutators

e. Implement exception handling to manage runtime errors. For example, handle exceptions when a passenger attempts to exceed luggage weight limit, add duplicate luggage ID, or if the passenger details are invalid.

f. Develop a file handling system to store and retrieve passenger and luggage information securely. Implement encryption and decryption mechanisms for data storage. Create functions for saving the passenger and luggage details to a file and leading data from a file, ensuring data remains encrypted.

Ensure that your program demonstrates the effective use of C++ features, adheres to object-oriented programming principles, and meets the specified requirements. You can add your own additional class, methods, Templates wherever needed. Your code should be well-structured, maintainable.capable of handling real-world scenarios and error- tolerant.

Implementation:

```
#include <iostream>

#include <string>

using namespace std;


const int MAX_SEATS = 100;


// Base class Passenger

class Passenger {

protected:

    string passengerID;

    string name;

    int age;

    char gender;

    string nationality;

    string ticketNumber;
    string mealPreference;

public:
    Passenger() {}

    void setDetails() {
        cout << "Enter Passenger ID: ";
        cin >> passengerID;
        cout << "Enter Name: ";
        cin.ignore();
        getline(cin, name);
        cout << "Enter Age: ";
        cin >> age;
        cout << "Enter Gender (M/F): ";
        cin >> gender;
        cout << "Enter Nationality: ";
        cin.ignore();
        getline(cin, nationality);
        cout << "Enter Ticket Number: ";
        cin >> ticketNumber;
    }

    // Check availability of a specific seat
    bool checkAvailability(const bool seats[], int seatNumber) {
        if (seatNumber >= 0 && seatNumber < MAX_SEATS) {
            return !seats[seatNumber]; // Seat is available if it's not booked
    }
    return false; // Invalid seat number
}

// Book a specific seat
bool bookTicket(bool seats[], int seatNumber) {
    if (checkAvailability(seats, seatNumber)) {
        seats[seatNumber] = true; // Book the seat
        cout << "Seat " << seatNumber << " booked successfully." << endl;
        return true;
    } else {
        cout << "Seat " << seatNumber << " is not available." << endl;
        return false;
    }
}

bool selectAndBookSeat(bool seats[]) {
    int seatNumber;
    cout << "Enter seat number to book: ";
    cin >> seatNumber;

    return bookTicket(seats, seatNumber);
}

void selectMealPreference() {
    int choice;
```

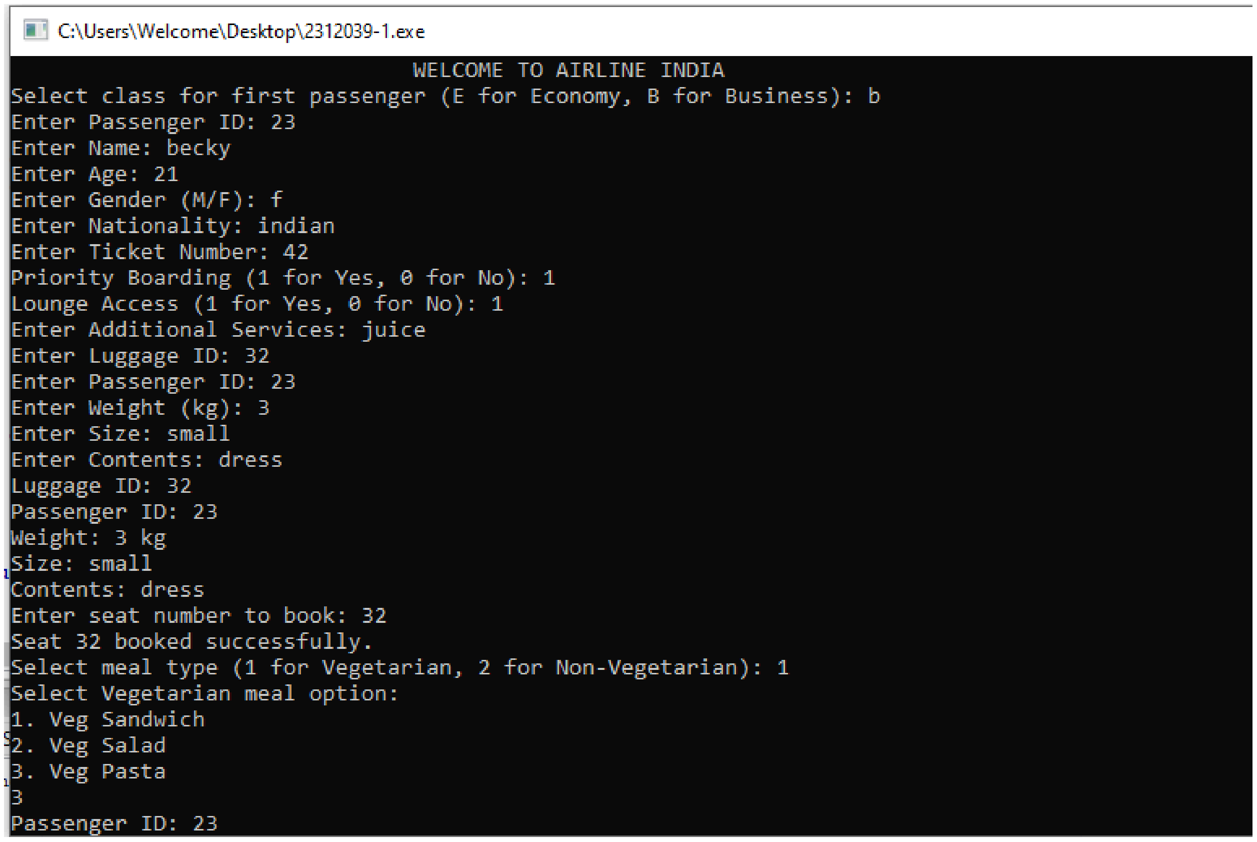
```
cout << "Select meal type (1 for Vegetarian, 2 for Non-Vegetarian): ";
cin >> choice;

if (choice == 1) {
    cout << "Select Vegetarian meal option:" << endl;
    cout << "1. Veg Sandwich" << endl;
    cout << "2. Veg Salad" << endl;
    cout << "3. Veg Pasta" << endl;
    cin >> choice;
    switch (choice) {
        case 1:
            mealPreference = "Veg Sandwich";
            break;
        case 2:
            mealPreference = "Veg Salad";
            break;
        case 3:
            mealPreference = "Veg Pasta";
            break;
        default:
            cout << "Invalid choice. Defaulting to Veg Sandwich." << endl;
            mealPreference = "Veg Sandwich";
            break;
    }
} else if (choice == 2) {
    cout << "Select Non-Vegetarian meal option:" << endl;
```

```
        cout << "1. Chicken Sandwich" << endl;

        cout << "2. Chicken Salad" << endl;

        cout << "3. Chicken Pasta" << endl;

        cin >> choice;

        switch (choice) {

            case 1:

                mealPreference = "Chicken Sandwich";

                break;

            case 2:

                mealPreference = "Chicken Salad";

                break;

            case 3:

                mealPreference = "Chicken Pasta";

                break;

            default:

                cout << "Invalid choice. Defaulting to Chicken Sandwich." << endl;

                mealPreference = "Chicken Sandwich";

                break;

        }

    } else {

        cout << "Invalid choice. Defaulting to Vegetarian meal." << endl;

        mealPreference = "Veg Sandwich";

    }

}


virtual void displayDetails() {
        cout << "Passenger ID: " << passengerID << endl;
        cout << "Name: " << name << endl;
        cout << "Age: " << age << endl;
        cout << "Gender: " << gender << endl;
        cout << "Nationality: " << nationality << endl;
        cout << "Ticket Number: " << ticketNumber << endl;
        cout << "Meal Preference: " << mealPreference << endl;
    }
};

// Derived class EconomyPassenger
class EconomyPassenger : public Passenger {
protected:
    int seatNumber;
    int baggageAllowance;

public:
    EconomyPassenger() : Passenger() {}

    void setEconomyDetails() {
        setDetails();
        cout << "Enter Baggage Allowance (kg): ";
        cin >> baggageAllowance;
    }

    void displayDetails() override {
        Passenger::displayDetails();

        cout << "Baggage Allowance: " << baggageAllowance << " kg" << endl;

    }

};


// Derived class BusinessPassenger

class BusinessPassenger : public Passenger {

protected:

    bool priorityBoarding;

    bool loungeAccess;

    string additionalServices;


public:

    BusinessPassenger() : Passenger() {}


    void setBusinessDetails() {

        setDetails();

        cout << "Priority Boarding (1 for Yes, 0 for No): ";

        cin >> priorityBoarding;

        cout << "Lounge Access (1 for Yes, 0 for No): ";

        cin >> loungeAccess;

        cout << "Enter Additional Services: ";

        cin.ignore();

        getline(cin, additionalServices);

    }
    void displayDetails() {
        Passenger::displayDetails();
        cout << "Priority Boarding: " << (priorityBoarding ? "Yes" : "No") << endl;
        cout << "Lounge Access: " << (loungeAccess ? "Yes" : "No") << endl;
        cout << "Additional Services: " << additionalServices << endl;
    }
};

// Class Luggage
class Luggage {
protected:
    string luggageID;
    string passengerID;
    double weight;
    string size;
    string contents;

public:
    Luggage() {}

    void setLuggageDetails() {
        cout << "Enter Luggage ID: ";
        cin >> luggageID;
        cout << "Enter Passenger ID: ";
        cin >> passengerID;
        cout << "Enter Weight (kg): ";
        cin >> weight;
        cout << "Enter Size: ";
        cin >> size;
        cout << "Enter Contents: ";
        cin.ignore();
        getline(cin, contents);
    }

    void displayDetails() {
        cout << "Luggage ID: " << luggageID << endl;
        cout << "Passenger ID: " << passengerID << endl;
        cout << "Weight: " << weight << " kg" << endl;
        cout << "Size: " << size << endl;
        cout << "Contents: " << contents << endl;
    }
};

void processPassenger(Passenger& passenger, bool seats[]) {
    if (passenger.selectAndBookSeat(seats)) { // Select and book a seat
        passenger.selectMealPreference(); // Select meal preference
        passenger.displayDetails(); // Display passenger details
    }
}

int main() {
    cout << "                              WELCOME TO AIRLINE INDIA" << endl;
```

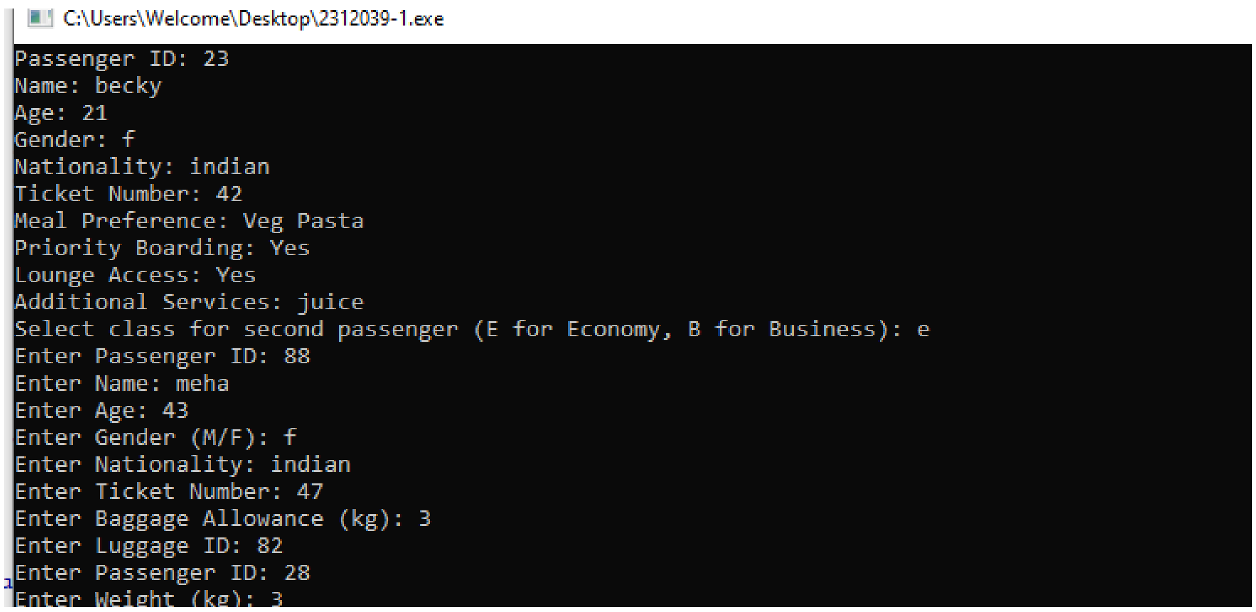
```
char classSelection1, classSelection2;

cout << "Select class for first passenger (E for Economy, B for Business): ";

cin >> classSelection1;


bool seats[MAX_SEATS] = {false}; // Initialize all seats as available


switch (classSelection1) {

    case 'E':

    case 'e': {

        EconomyPassenger econPass1;

        econPass1.setEconomyDetails();

        Luggage l1;

        l1.setLuggageDetails();

        l1.displayDetails();

        processPassenger(econPass1, seats);

        break;

    }

    case 'B':

    case 'b': {

        BusinessPassenger busiPass1;

        busiPass1.setBusinessDetails();

         Luggage l2;

        l2.setLuggageDetails();

        l2.displayDetails();

        processPassenger(busiPass1, seats);

        break;
    }
    default:
        cout << "Invalid class selection for first passenger." << endl;
        break;
}

cout << "Select class for second passenger (E for Economy, B for Business): ";
cin >> classSelection2;

switch (classSelection2) {
    case 'E':
    case 'e': {
        EconomyPassenger econPass2;
        econPass2.setEconomyDetails();
         Luggage l3;
        l3.setLuggageDetails();
        l3.displayDetails();
        processPassenger(econPass2, seats);
        break;
    }
    case 'B':
    case 'b': {
        BusinessPassenger busiPass2;
        busiPass2.setBusinessDetails();
         Luggage l4;
        l4.setLuggageDetails();
```

```
            l4.displayDetails();

            processPassenger(busiPass2, seats);

            break;

        }

        default:

            cout << "Invalid class selection for second passenger." << endl;

            break;

    }


    return 0;

}
```

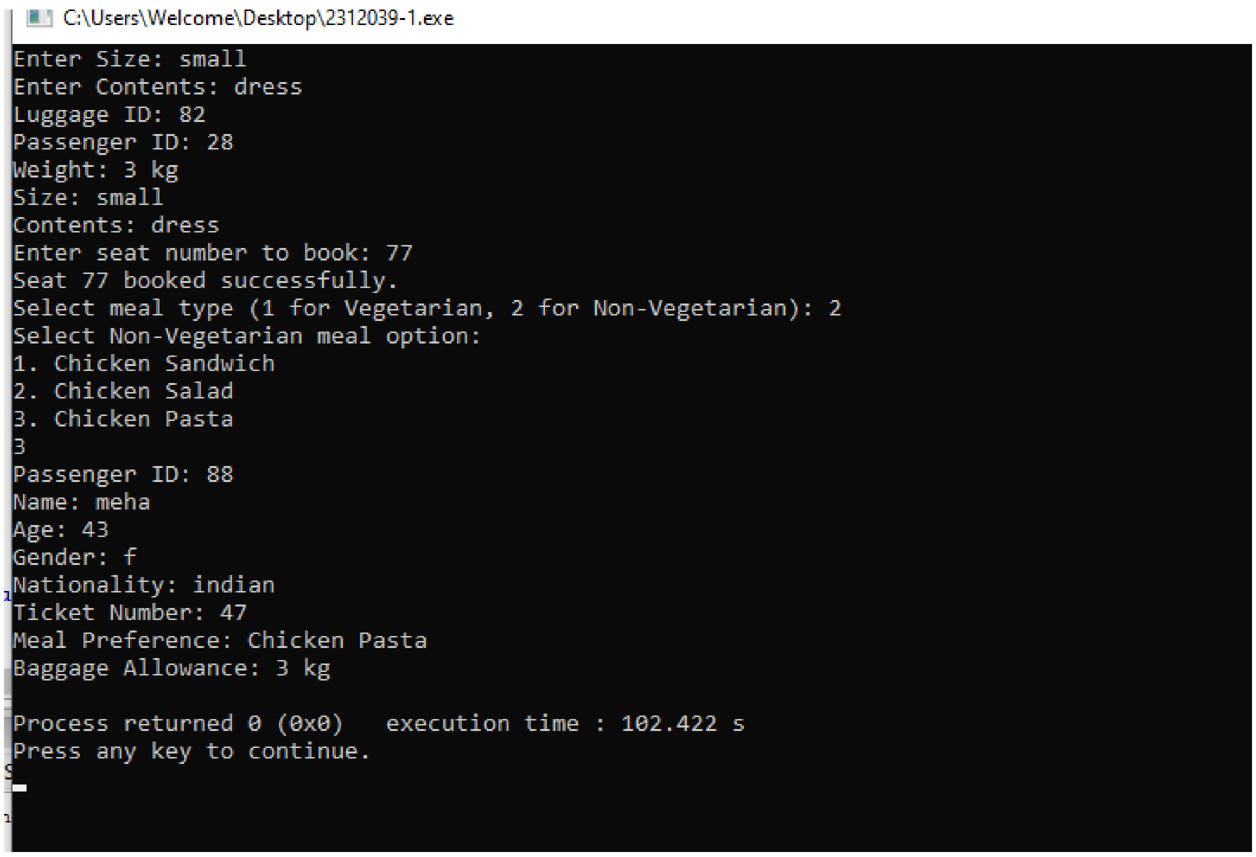
Output:



```
C:\Users\Welcome\Desktop\2312039-1.exe
                               WELCOME TO AIRLINE INDIA
Select class for first passenger (E for Economy, B for Business): b
Enter Passenger ID: 23
Enter Name: becky
Enter Age: 21
Enter Gender (M/F): f
Enter Nationality: indian
Enter Ticket Number: 42
Priority Boarding (1 for Yes, 0 for No): 1
Lounge Access (1 for Yes, 0 for No): 1
Enter Additional Services: juice
Enter Luggage ID: 32
Enter Passenger ID: 23
Enter Weight (kg): 3
Enter Size: small
Enter Contents: dress
Luggage ID: 32
Passenger ID: 23
Weight: 3 kg
Size: small
Contents: dress
Enter seat number to book: 32
Seat 32 booked successfully.
Select meal type (1 for Vegetarian, 2 for Non-Vegetarian): 1
Select Vegetarian meal option:
1. Veg Sandwich
2. Veg Salad
3. Veg Pasta
3
Passenger ID: 23
```


C:\Users\Welcome\Desktop\2312039-1.exe

```
Passenger ID: 23
Name: becky
Age: 21
Gender: f
Nationality: indian
Ticket Number: 42
Meal Preference: Veg Pasta
Priority Boarding: Yes
Lounge Access: Yes
Additional Services: juice
Select class for second passenger (E for Economy, B for Business): e
Enter Passenger ID: 88
Enter Name: meha
Enter Age: 43
Enter Gender (M/F): f
Enter Nationality: indian
Enter Ticket Number: 47
Enter Baggage Allowance (kg): 3
Enter Luggage ID: 82
Enter Passenger ID: 28
Enter Weight (kg): 3
```


C:\Users\Welcome\Desktop\2312039-1.exe

```
Enter Size: small
Enter Contents: dress
Luggage ID: 82
Passenger ID: 28
Weight: 3 kg
Size: small
Contents: dress
Enter seat number to book: 77
Seat 77 booked successfully.
Select meal type (1 for Vegetarian, 2 for Non-Vegetarian): 2
Select Non-Vegetarian meal option:
1. Chicken Sandwich
2. Chicken Salad
3. Chicken Pasta
3
Passenger ID: 88
Name: meha
Age: 43
Gender: f
Nationality: indian
Ticket Number: 47
Meal Preference: Chicken Pasta
Baggage Allowance: 3 kg

Process returned 0 (0x0)   execution time : 102.422 s
Press any key to continue.
```