Report Template: Object-Oriented Programming and Design

Anna Yevtushenko ОПД2

TASK:

We need to create a console system to book plane tickets. Using the command line, the user should be able to:

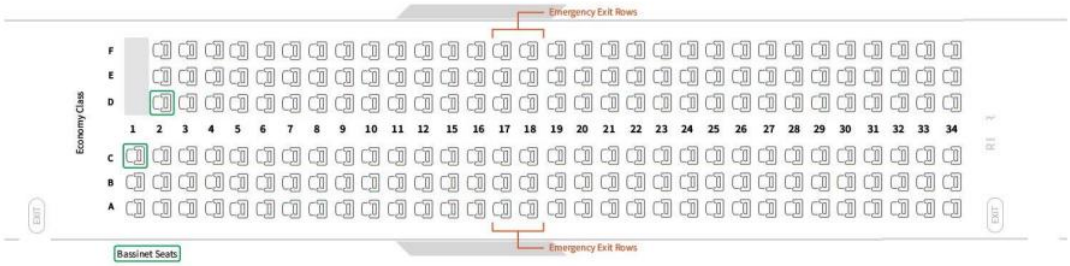
- •Check (booked/free places)
- • Book/cancel a ticket
- • Check customer information

To implement the program must first read the **config file**:

```
2 // number of records
11.12.2022 FQ12 6 1-20 100$ 21-40 50$ // date, flight number, number of seats per row (6 means A B C D E F), price for rows 1-20 is 100$, price for rows 21-40 is 50$
11.12.2022 HJ114 9 1-10 10$ 11-50 20$ // date, flight number, number of seats per row (6 means A B C D E F G H I), price for rows 1-10 is 10$, price for rows 11-50 is 20$
```

When I submit, this file must be different!!!!!!!!!!

An example of an airplane:



There must be a different number and the prices (seats per row)

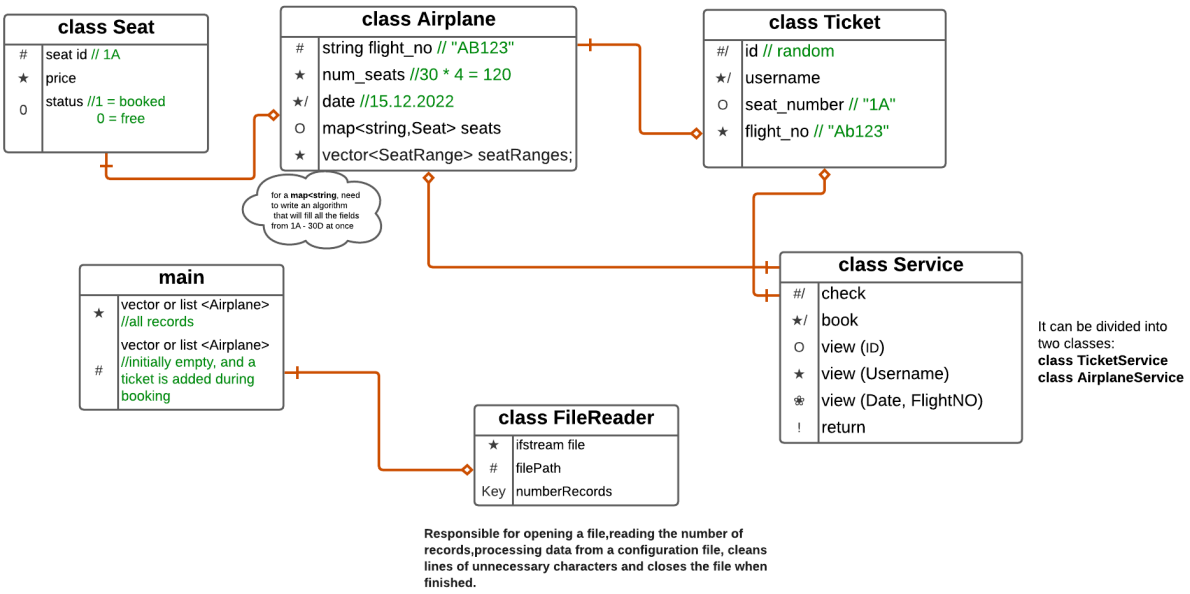
For example :

1-20 = **100**$

21-40 = **50**$

Pay attention!!!!!!!!!!!!! If the number of places per row is 6 (ABCDEF), if 9, as in the example of the second record, there should already be 9 letters.

15.12.2022 ABIR 4 1-15 120$ 16-30 80$



TESTING:

```
4 // number of records
01.01.2023 JK321 8 1-10 100$ 11-20 90$ 21-30 50$
03.01.2023 LM654 9 1-25 95$ 26-50 65$
05.01.2023 NO987 6 1-20 125$ 21-30 100$
07.01.2023 PQ123 9 1-10 200$ 11-20 150$ 21-30 100$
```

Book command

```
string commands[] = {

    "book 01.01.2023 JK321 1A CristianoRonaldo",
    "book 03.01.2023 LM654 1B OleksandrZinchenko",
    "book 05.01.2023 NO987 1B MykhailoMudryk",
    "book 11.12.2022 NO987 1C PavloZibrov", //incorrect date
    "book 11.12.2022 PQ126 1C LionelMessi", //incorrect flight_no
    "book 01.01.2023 JK321 1A KylianMbappe", //seat is already booked
```

```
Successfully read 4 flights from the file.
Confirmed with ticket ID 1
Confirmed with ticket ID 2
Confirmed with ticket ID 3
Date is incorrect.
Flight not found or incorrect date.
Seat is already booked.

D:\OOPD\OOPD-assigment1\ConsoleApplication1\x64\Debug\ConsoleApplication1.exe
th code 0.
Press any key to close this window . . .
```

Chek command

```
string commands[] = {

    "book 01.01.2023 JK321 1A CristianoRonaldo",
    "book 01.01.2023 JK321 1B OleksandrZinchenko",
    "book 01.01.2023 JK321 1C MykhailoMudryk",
    //"book 11.12.2022 NO987 1C PavloZibrov", //incorrect date
    //"book 11.12.2022 PQ126 1C LionelMessi", //incorrect flight_no
    //"book 01.01.2023 JK321 1A KylianMbappe", //seat is already booked


    "check 01.01.2023 JK321",
    "check 01.01.2023 JK322", //incorrect flight_no
    "check 01.03.2023 JK321", //incorrect date
```

```
19H - 90$
1D - 100$
1E - 100$
1F - 100$
1G - 100$
1H - 100$
```

```
9E - 100$
9F - 100$
9G - 100$
9H - 100$
Record not found or incorrect date.
Record not found or incorrect date.
```

Return command

```
"book 01.01.2023 JK321 21A CristianoRonaldo",
"book 01.01.2023 JK321 21A LionelMessi",//seat is already booked
"return 1",
"book 01.01.2023 JK321 21A LionelMessi",
```

```
Successfully read 4 flights from the file.
Confirmed with ticket ID 1
Seat is already booked.
Confirmed 50$ refund for CristianoRonaldo
Confirmed with ticket ID 2
```

View(ID) command

```
"book 01.01.2023 JK321 21A PavloZibrov",
"view 1",
"view 2", // doesn't exist
```

```
Successfully read 4 flights from the file.
Confirmed with ticket ID 1
Flight JK321, 01.01.2023, seat 21A, price 50$, PavloZibrov
Ticket not found.
```

View(Username) command

```
"book 01.01.2023 JK321 21A PavloZibrov",
"view PavloZibrov",
"view MykolaZyryanov", // did not book
```

```
Successfully read 4 flights from the file.
Confirmed with ticket ID 1
Flight JK321, 01.01.2023, seat 21A, price 50$
No tickets found for user: MykolaZyryanov
```

View(Date, FlightNo) command

```
"book 01.01.2023 JK321 21A CristianoRonaldo",
"book 01.01.2023 JK321 1F Andriy Shevchenko ",

"view 01.01.2023 JK321 "
```

```
Confirmed with ticket ID 1
Confirmed with ticket ID 2
Flight JK321, 01.01.2023, seat 21A, price 50$, User: CristianoRonaldo
Flight JK321, 01.01.2023, seat 1F, price 100$, User: AndriyShevchenko
```

```
>
> book 1.12.23 FR12 1A AdamSmith
Flight not found or incorrect date.
> book 01.01.2023 JK321 1A AdamSmith
Confirmed with ticket ID 1
> book 01.01.2023 JK321 1A AdamSmitah
Seat is already booked.
> return 1
Confirmed 100$ refund for AdamSmith
> book 01.01.2023 JK321 1A AdamSmitah
Confirmed with ticket ID 2
> book 01.01.2023 JK321 2A AdamSmitah
Confirmed with ticket ID 3
> view AdamSmitah
Flight JK321, 01.01.2023, seat 2A, price 100$
> return 1
Ticket not found.
> return 2
Confirmed 100$ refund for AdamSmitah
> book 01.01.2023 JK321 1A AdamSmitah
Confirmed with ticket ID 4
> view AdamSmitah
Flight JK321, 01.01.2023, seat 2A, price 100$
Flight JK321, 01.01.2023, seat 1A, price 100$
> view 4
Flight JK321, 01.01.2023, seat 1A, price 100$, AdamSmitah
> view 3
Flight JK321, 01.01.2023, seat 2A, price 100$, AdamSmitah
> view 2
Ticket not found.
> view 01.01.2023 JK321
Flight JK321, 01.01.2023, seat 2A, price 100$, User: AdamSmitah
Flight JK321, 01.01.2023, seat 1A, price 100$, User: AdamSmitah
>
```

```
10 \\number of records
10.01.2023 AB123 7 1-15 120$ 16-30 80$
12.01.2023 CD456 8 1-20 110$ 21-40 90$
14.01.2023 EF789 9 1-10 130$ 11-25 105$ 26-40 75$
16.01.2023 GH101 6 1-15 95$ 16-30 70$
18.01.2023 IJ112 8 1-20 140$ 21-35 115$
20.01.2023 KL221 7 1-10 150$ 11-20 110$
22.01.2023 MN334 9 1-15 125$ 16-25 90$ 26-35 60$
24.01.2023 OP556 6 1-12 135$ 13-25 100$
26.01.2023 QR789 8 1-20 120$ 21-30 85$
28.01.2023 ST910 7 1-10 175$ 11-20 130$
```

New configuration file:

Control Questions:

1)Explain data abstraction and encapsulation programming technique.

Абстракція даних — це концепція програмування, де приховуються складні деталі реалізації об'єкта, а користувачу надаються тільки важливі та необхідні для взаємодії з цим об'єктом характеристики. Таким чином, абстракція дозволяє працювати з об'єктом, не заглиблюючись у його внутрішню роботу.

Інкапсуляція — це принцип програмування, який полягає в приховуванні внутрішнього стану об'єкта та обмеженні доступу до нього ззовні. Інкапсуляція дозволяє захистити дані від некоректних змін, забезпечуючи доступ до них лише через спеціально визначені методи.

Аспект	Абстракція	Інкапсуляція
Визначення	Приховує непотрібні деталі та представляє лише основні функції.	Об'єднує дані та методи в єдиний блок, обмежуючи доступ до внутрішнього стану.
Фокус	Концепція рівня дизайну	Концепція рівня реалізації
призначення	Спрощуйте складні системи, зосереджуючись на важливому.	Забезпечте цілісність даних і надайте зрозумілий інтерфейс для взаємодії.
Що воно приховує	Приховує деталі впровадження	Приховує внутрішній стан і деталі реалізації
Рівень деталізації	Відображає лише основні функції	Об'єднує дані та методи разом, абстрагуючи реалізацію
Випадок використання	Надання спрощеного уявлення про складні системи.	Захист даних і забезпечення зрозумілого інтерфейсу для взаємодії.
Реалізація	Досягається через інтерфейси, абстрактні класи та успадкування.	Досягається за допомогою модифікаторів доступу (публічний, приватний, захищений) і класів.
Залежність	Можуть існувати без інкапсуляції, але часто використовуються разом.	Часто використовується в поєднанні з абстракцією для створення модульного коду, який зручно підтримувати.

2. What is an interface?

Інтерфейс — це контракт, який визначає набір методів, що клас повинен реалізувати, але не містить їх реалізації. Це дозволяє різним класам реалізувати ці методи по-своєму. У C++, інтерфейсом є клас, що складається тільки з чистих віртуальних методів.

Різниця між інтерфейсом і абстрактним класом полягає в тому, що в абстрактному класі можуть бути як реалізовані методи, так і чисто віртуальні, тоді як інтерфейс не містить жодної реалізації — всі методи мають бути чисто віртуальними.

> Одного разу навчившись кататись на велосипеді, людина вміє це робити навіть через декілька років.
> Велосипед- клас (публічні методи : крутити педалі, крутити руль та їздити).
> Тобто будь-який об'єкт, який реалізує інтерфейс велосипеда (у якого ми можемо покрутити руль та педалі) може використовувати наша людина.
>
> Абстрактний клас – клас у якого є хоча б один чистий віртуальний метод. Інтерфейс – це і є абстрактний клас, але у якого всі методи чисто віртуальні.
> Тобто не може бути ніякої реалізації.

3. Which access specifiers do you know?

- **Public-** члени класу доступні ззовні класу, їх можна викликати та змінюватии з будь-якої частини програми
- **Private** – члени класу доступні тільки всередині самого класу
- **Protected –** члени класу доступні всередині цього ж класу, або у його похідних класах

4. What is the difference between struct and class in C++?

Структура за дефолтом public та не бажано створювати функції всередині
Клас за дефолтом private а також можна створювати методи

5. What is this pointer?

Покажчик «this» у C++ — це неявний вказівник, доступний у нестатичних функціях-членах класу чи структури. Він вказує на поточний екземпляр об’єкта, дозволяючи об’єкту отримати доступ до власних змінних-членів і функцій.

Основні моменти про this:

1. **Вказує на поточний об'єкт:**
   Якщо у вас є клас, кожен його метод автоматично має доступ до вказівника this, який вказує на той екземпляр об'єкта, з яким працює метод.
2. **Застосування в конструкторах та методах:**
   Коли ви хочете чітко вказати, що звертаєтесь до поля або методу саме цього об'єкта, можна використовувати this. Особливо це корисно, коли параметри методу мають такі ж імена, як і поля класу:

```
class MyClass {
private:
    int value;

public:
    MyClass(int value) {
        this->value = value;  // "this->value" означає поле класу,
        //а просто "value" - параметр конструктора
    }

    void setValue(int value) {
        this->value = value;  // використовується для уникнення неоднозначності
    }
```

```
#include <iostream>

class Person {

private:
std::string name;
int age;

public:
Person(const std::string& name, int age) {
this->name = name;
this->age = age;
}

void displayInfo() {
std::cout << "Name: " << this->name << std::endl;
std::cout << "Age: " << this->age << std::endl;
}

void updateAge(int newAge) {
this->age = newAge;
}
};

int main() {

Person person("Armaan", 25);

person.displayInfo();
person.updateAge(30);
std::cout << "After updating age:" << std::endl;
person.displayInfo();

return 0;
}
```

6. How can you initialize fields in the object?

1) За допомогою конструкторів
2)Використання setters методів

7. What is constructor/ What is destructor?

Конструктор — це спеціальний метод у класі, який автоматично викликається при створенні об'єкта цього класу. Він ініціалізує поля об'єкта та може приймати параметри для задання початкових значень.

Деструктор — це спеціальний метод у класі, який автоматично викликається при знищенні об'єкта. Він використовується для звільнення ресурсів (пам'яті, файлів тощо), які були виділені об'єктом.

```
class MyClass {
public:
    int value;

    MyClass(int v) : value(v) {}

    ~MyClass() {

    }
};

MyClass obj(10);
```

9. What is a friend class?

Дружній клас — це клас, якому дозволено доступ до **приватних** або **захищених** даних іншого класу. Зазвичай, інші класи не можуть бачити ці дані, але якщо один клас оголошений як "друг" іншого, він отримує такий доступ.

```
#include <iostream>

class A {
private:
    int secret = 42;  // Приватне поле, до якого інші класи не мають доступу

    // Оголошуємо клас B другом, щоб він міг бачити secret
    friend class B;
};

class B {
public:
    void showSecret(A& obj) {
        std::cout << "Secret value: " << obj.secret << std::endl;
         // Доступ до приватного поля A
    }
};

int main() {
    A objA;
    B objB;
    objB.showSecret(objA);
    // Клас B може "бачити" секрет класу A
    return 0;
}
```

10. What is const member function?

const-функція у C++ — це функція, яка гарантує, що не буде змінювати стан об'єкта, для якого вона викликається. Це означає, що всередині цієї функції не можна змінювати значення членів класу (окрім тих, які позначені як mutable)

```
class MyClass {
private:
    int value;

public:
    MyClass(int v) : value(v) {}

    // const-функція-член
    int getValue() const {
        return value;  // Повертає значення, не змінюючи його
    }

    // Неконстантна функція
    void setValue(int v) {
        value = v;  // Може змінювати значення
    }
};

int main() {
    MyClass obj(10);

    // getValue() - це const-функція, вона не змінює об'єкт
    std::cout << obj.getValue() << std::endl;

    // setValue() - звичайна функція, може змінювати об'єкт
    obj.setValue(20);
    std::cout << obj.getValue() << std::endl;

    return 0;
}
```

11. What does mutable data member?

mutable — це ключове слово в C++, яке дозволяє змінювати значення змінної-члена навіть у const-функціях.

```
class MyClass {
private:
    mutable int counter = 0;
    // mutable дозволяє змінювати цю змінну навіть у const-функціях

public:
    void increment() const {
        counter++;  // Можна змінити, хоча функція const
    }
};
```

12. What is objects composition?

Об'єктна композиція — це концепція в ооп, коли один клас включає в себе інші класи як свої члени. Це означає, що один об'єкт містить інші об'єкти як частини своєї структури. Композиція дозволяє створювати складніші об'єкти, об'єднуючи простіші об'єкти.

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

class Room {
private:
    std::string roomName;

public:
    Room(std::string name) : roomName(name) {}

    void describe() const {
        std::cout << "This is the " << roomName << std::endl;
    }
};

class House {
private:
    Room livingRoom;
    Room bedroom;

public:
    House() : livingRoom("Living Room"), bedroom("Bedroom") {}
    // Ініціалізація кімнат всередині будинку

    void describeHouse() const {
        livingRoom.describe();
        bedroom.describe();
    }
};

int main() {
    House myHouse;
    myHouse.describeHouse();  // Описуємо кімнати будинку
    return 0;
}
```