CPT106 C++ Programming and Software Engineering II

# Group Project

**Team leader**

|  |  |  |  |
| --- | --- | --- | --- |
| **Student ID** | **1929013** | **Student Name** | **Yutong he** |

**Contribution table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Student ID** | **Student Name**  **(Signed manually)** | **Tasks** | **Contribution (%)** |
| 1929962 | Hanyuan Ma | the program code | 25% |
| 1929013 | Yutong He | The specification and the maintain of the code and document | 25% |
| 1927130 | Xinen Li | problem, analysis and test | 25% |
| 1822764 | Zhuofan Hu | Design and draw the UML.  Analysis each class. | 25% |
| Total | | | **100%** |

the document of the code

[Group Project 1](#_Toc74158904)

[Specifications: 3](#_Toc74158905)

[Analysis and design: 4](#_Toc74158906)

[management of source code and lifecycle by software tools: 15](#_Toc74158907)

[Test: 15](#_Toc74158908)

[Bug report: 18](#_Toc74158909)

[User manual: 18](#_Toc74158910)

# Specifications:

**Overall spec.:**

In this project, to meet the requirement of airways company, our group creates a computer system which can be responsible for a large part of the operation of the airways company. The operations contain: book tickets, change or add the flights information,

**System spec.:**

First, it matches the user id to the password, if they can be matched properly, then users can log in; otherwise, the users can not log in and achieve the steps after.

After logging in, users should choose the position of them. Therefore, here’s the specific functions that meets the requirement of airways company.

**For administrator:**

1. They can include new flight
2. They can change the price for the flight
3. They can change the dates for flights, which include leaving time and arriving time
4. They can change the source and destination for flights
5. Then can change the plane type for flights

**For agent:**

1. They can use flight ID to search the flight information
2. They can find the flights which still have seats
3. They can search flights by using leaving time
4. They can search flights by using arriving time
5. They can search flights by using source of the airport
6. They can search flights by using destination of the airport
7. They can book tickets for customer

**For manager:**

1. They can find the number of the certain type plane
2. They can get the total passenger number in certain flight
3. They can check the revenue of the airport

**For customer:**

1. They can book seat

# Analysis and design:

**Analysis:**

After reading the problem, there are the content we should design:

Enter the system with correct password and account number, then choice the role which you are. Base on different role, there are different operations can be selected.

**Login system:**

**Input:**

The account number and password to login, the selection of user’s operations.

**Output:**

The result of input, the tips which ask people to input.

**Variable:**

For the login system, the information of the account should be stored for login, thus, there are a file to store these data.

For the account number and password to login and the selection of user’s operations, aet them with variable type is “string”

The selection which user input should be a thin integral, set them with variable type is “int”

**Plane:**

There are two attributes about “plane”, they are the type of plane and the distributions of the seats in different type of plane. Thus, set the “plane” as a “class”, the type of plane and the distributions of the seats in different type of plane are set as variable with the type is “string”

**Flight:**

Set the “flight” as a class since it has plenty of attributes with it. They are ID, total number of the seat, remain number of the seat, leaving time, arriving time, leaving place, arriving place and price of one flight.

ID, total number of the seat, remain number of the seat, leaving time, arriving time, leaving place, arriving place of flight set as variables with type “string”

For the price of flight, set it as variable with type “double”

There are four types of roles.

1. **Administrator:**

**In this role, the input should be:**

the choice of operations;

for new flight: the information including dates, price, airline, airports and type of flight.

For existing flight: one or more change for the information which are included: dates, price, airports and type of flight, origin and destination.

**The output should:**

The selections of operations, the tips for the wrong input, the tips for the successful operations.

**The variable should be:**

The administrator its own, since there are plenty of different operators, set it as a “class”.

The selections should be a number, set it with type of “int”

The dates, seats, type of flight, the ID of flight are string of data, use type “string” as the type of data for them.

The price may not be an integer, thus, use “double” as the type of the data.

The information including dates, price, airline, airports and type of flight should be stored, thus set a file to store them.

1. **Agents**

**Input:**

For agents, this input should include: the selection which people choice, the ID, living time, arriving time, departure and the destination of the flight.

**Output:**

For agents, the output should include: The selections of operations which agents can do, the result of the operations, the tips which ask people to input, the ID, living time, arriving time, seats, departure and the destination of the flight.

**Variable:**

There are plenty of operations which agents can do, thus, set agents with the type “class”.

The ID, living time, arriving time, departure and the destination of the flight and the name and ID of passengers are the string, thus, use “string” as the variable type for them. In addition, set them as the local variable in different function blocks.

The selections which people choice is a thin integral, thus, set it as a local variable with the variable type “int”.

There are some of data which requires to store to preparing to use or change, thus, set a file to store these data.

1. **Manager:**

**Input:**

For manager, the input should include: the operation which user select, the ID and type of the flight.

**Output:**

For manager, the output should include: the selections of operations, the tips which ask for the input, the result of operation, the total passenger’s number per flight, the total revenue.

**Variable:**

The “Manager” can do variety of operations, thus set it as a “class”

The selections which user select should be a thin integral, set it as a local variable with the type is “int”

The ID and type of the flight should be used to get the input from keyboard, set them as local variable with variable type is “string”

The passengers’ number and total revenue are used to output the result of them to the screen, set them as local variable and set the variable type “double” for them.

1. **Customer:**

**Input:**

The input should include: the ID and name of passenger, the seat which passenger select.

**Output:**

The tips which ask user to input, the information of the flight which passenger books, the seat, name and ID of passengers, the result after the operations.

**Variable:**

Since there are some information about passenger which are require to store and change, set a file to store them, set a “class” to input and output these data.

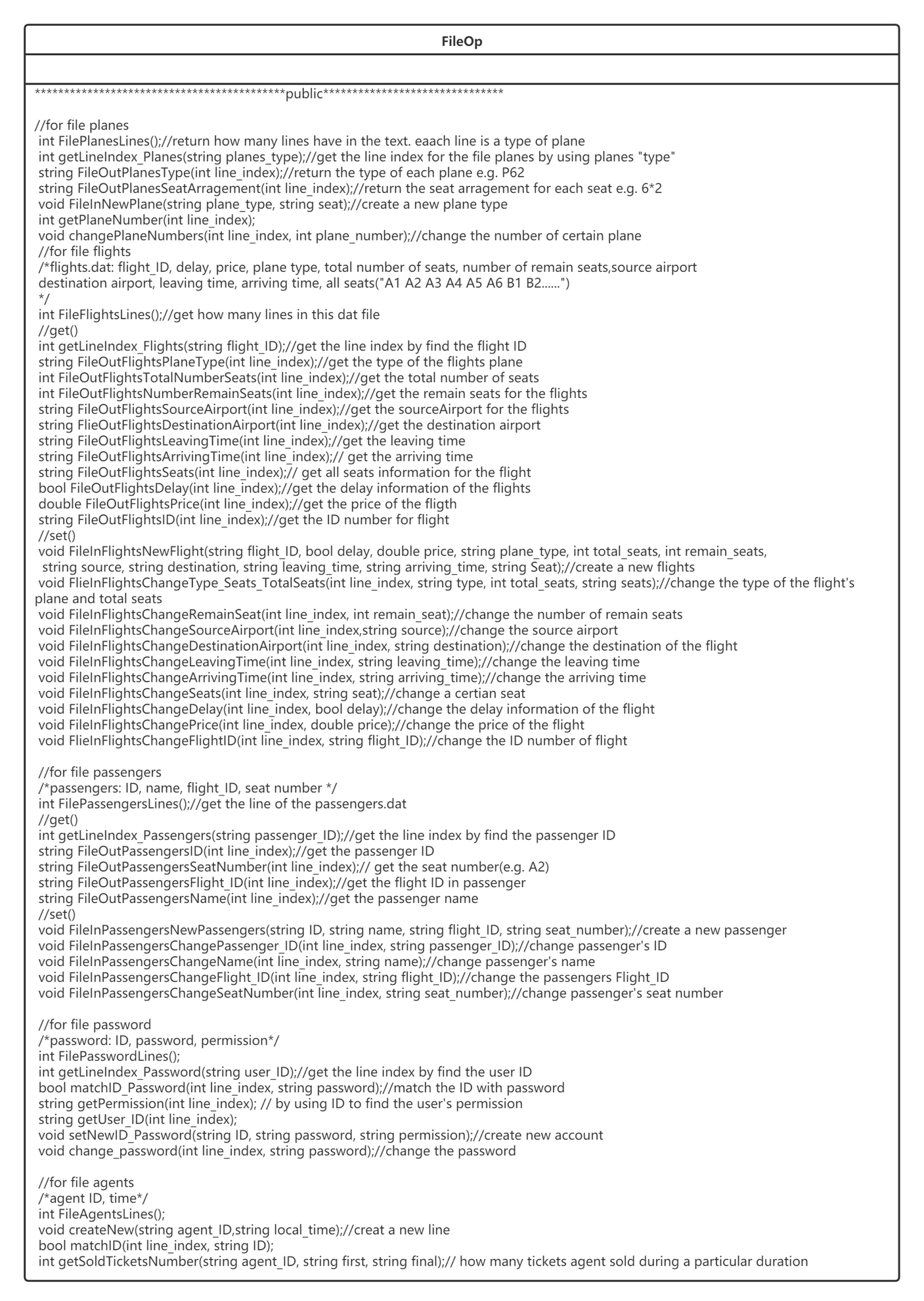
For customer, set a “class” for it to do the operation of booking ticket.

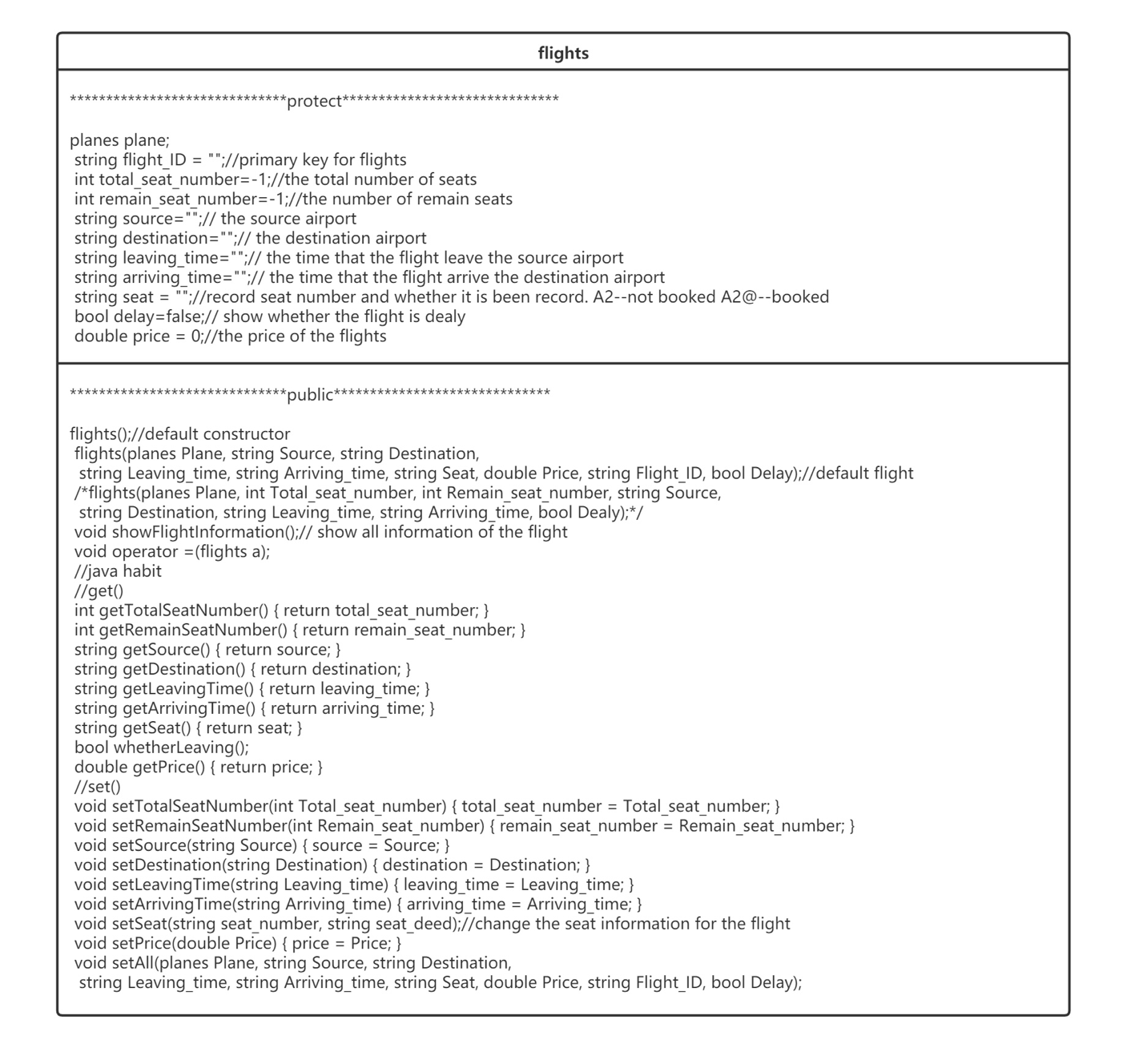
For ID and name of passenger, the seat which passenger select and the ID of the flight, in class “customer”, they are used to get the data from keyboard, set them as local variable with variable type is “string”

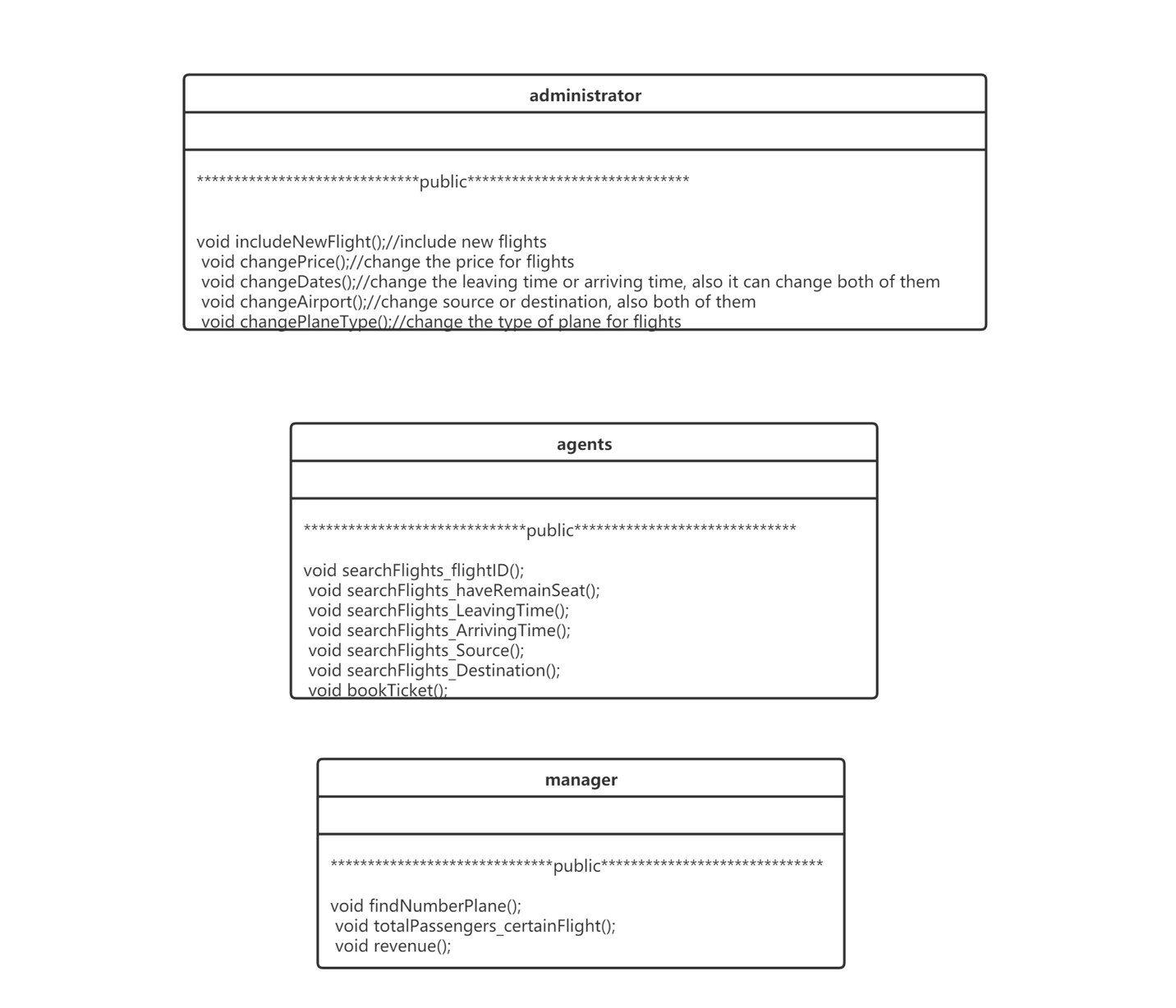
For ID and name of passenger, the seat which passenger select and the ID of the flight, in class “passenger”, they are use to input and output from the file “passenger”, set them as the global variable in class “passenger” with the type of variable is “string”

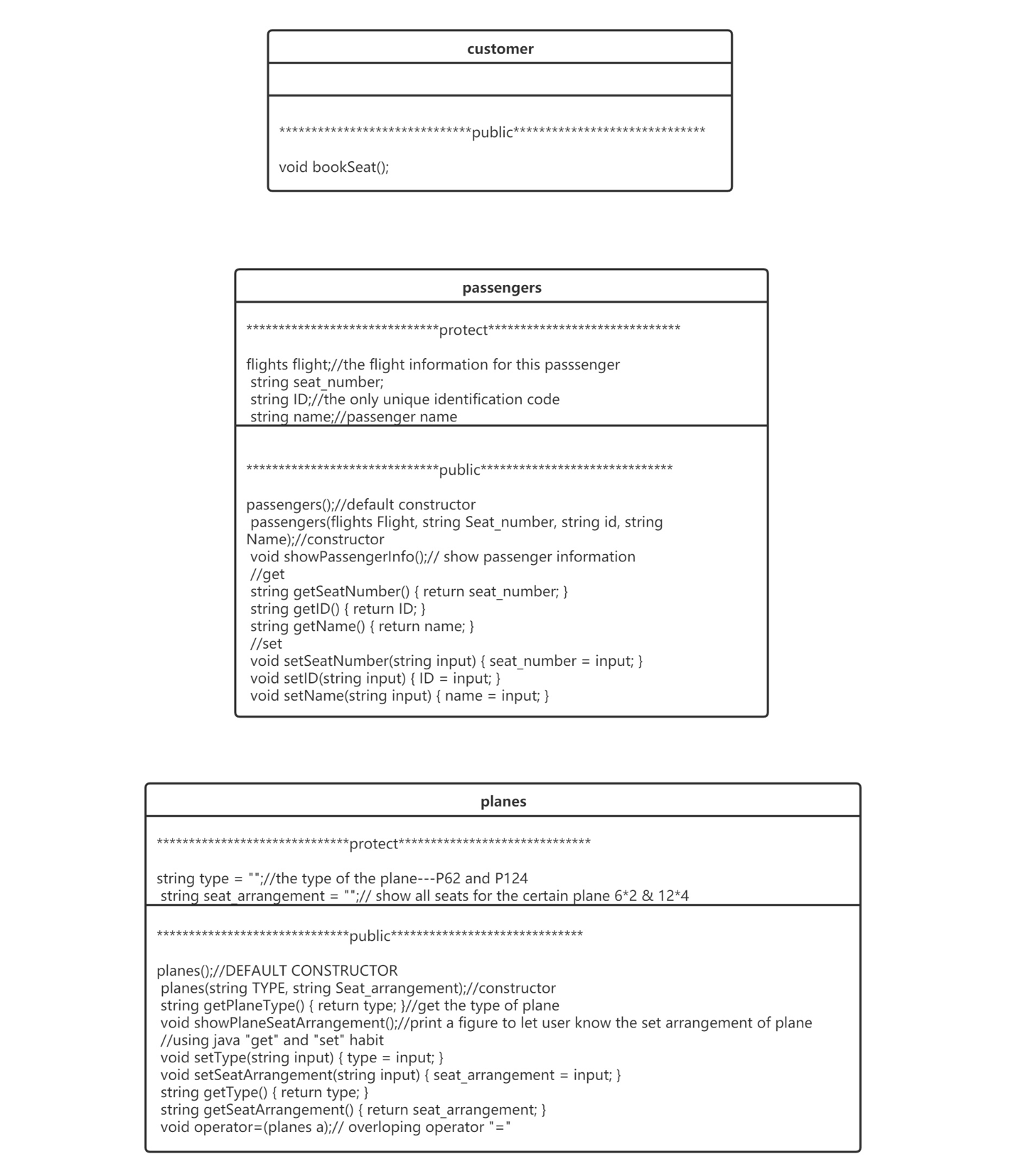
**Design:**

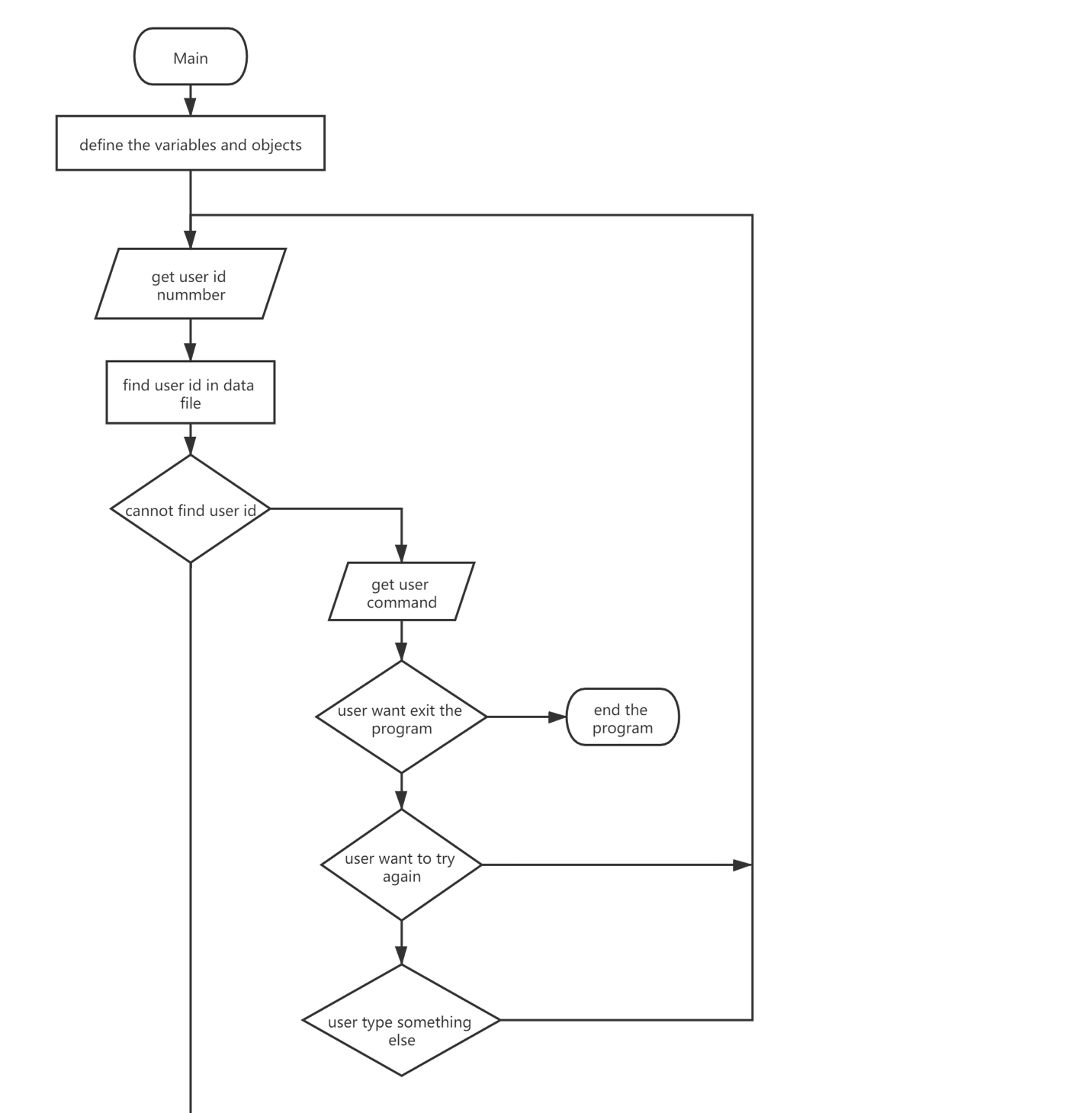
1. **Class design:**

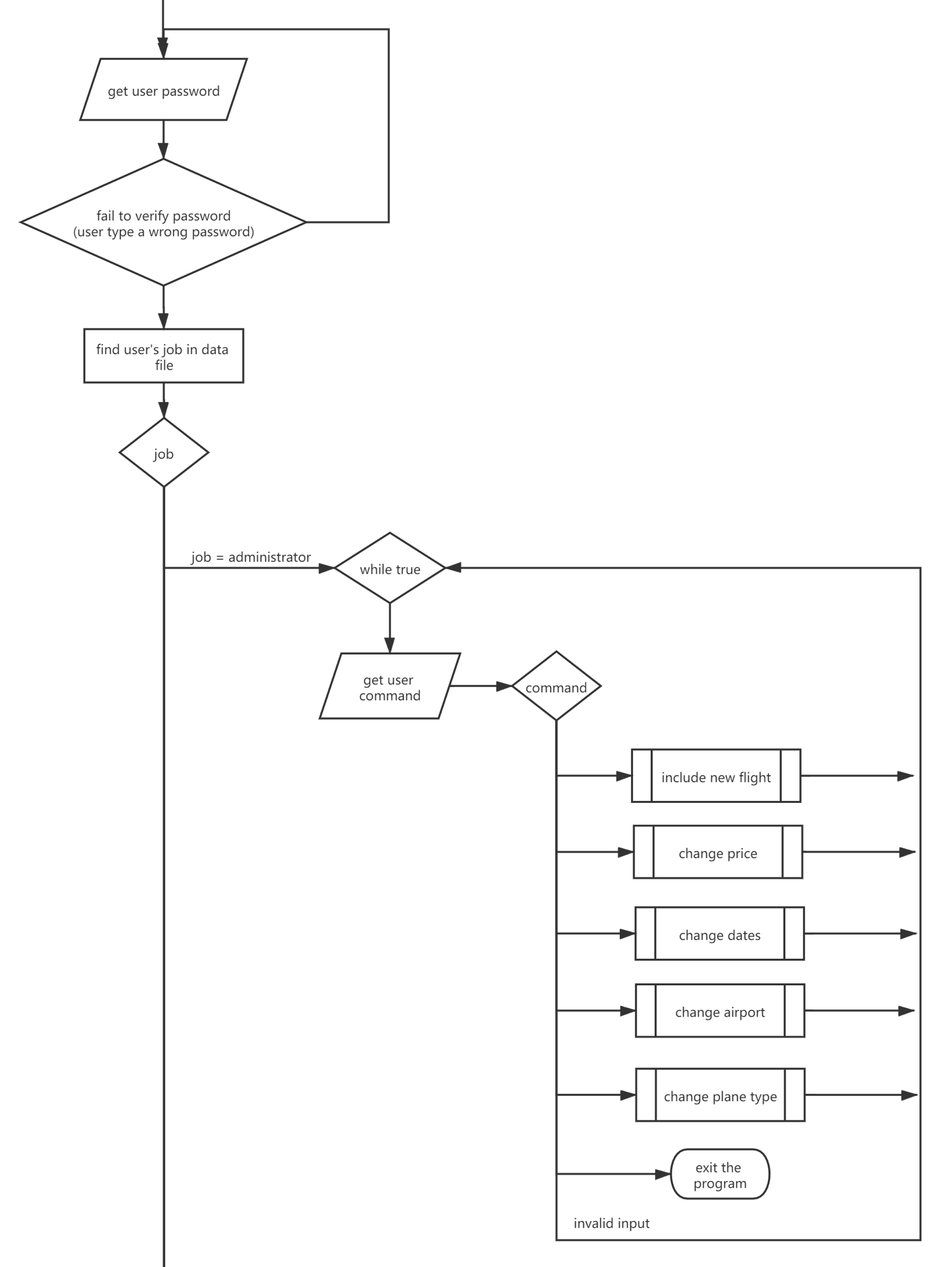


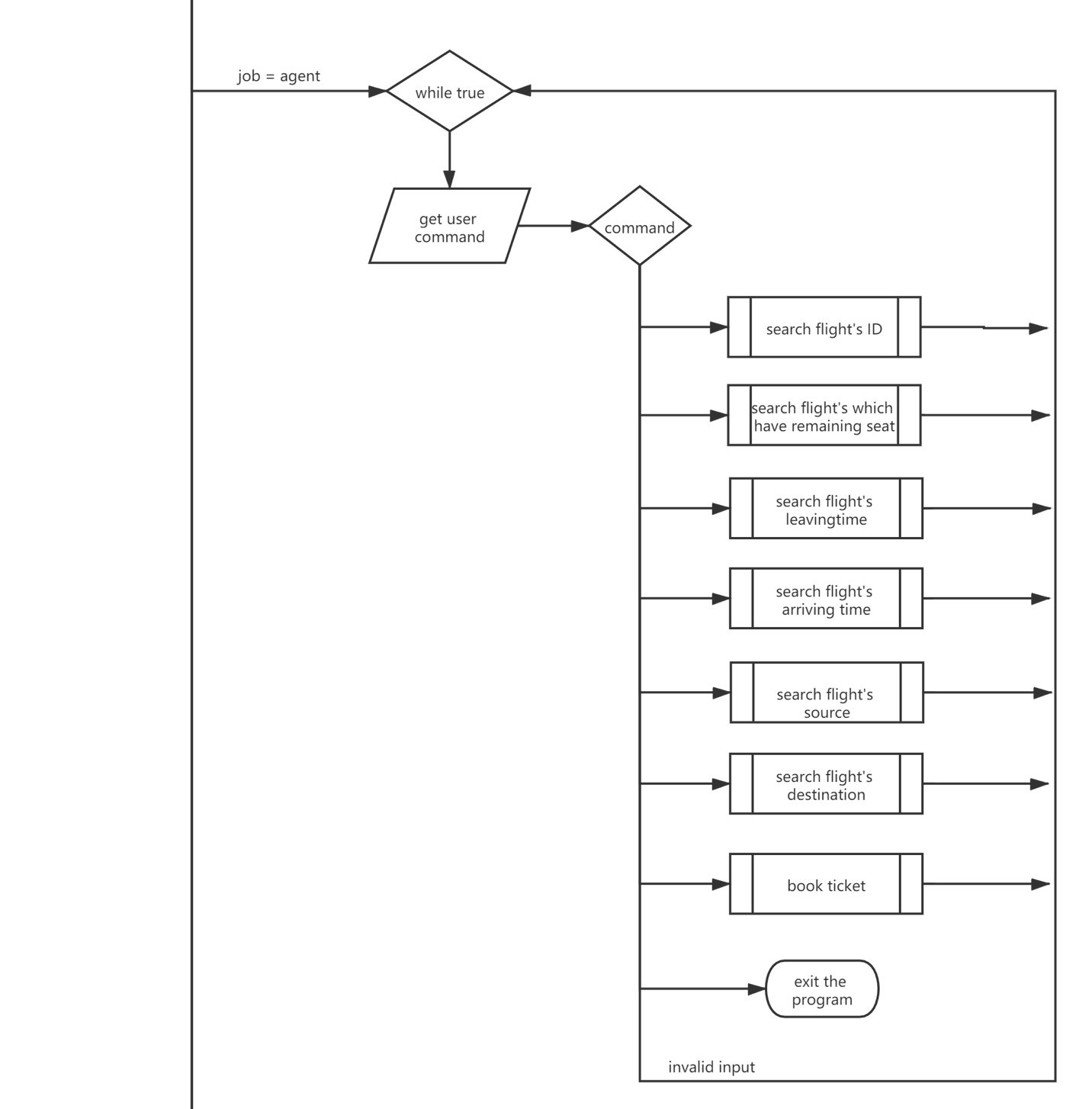


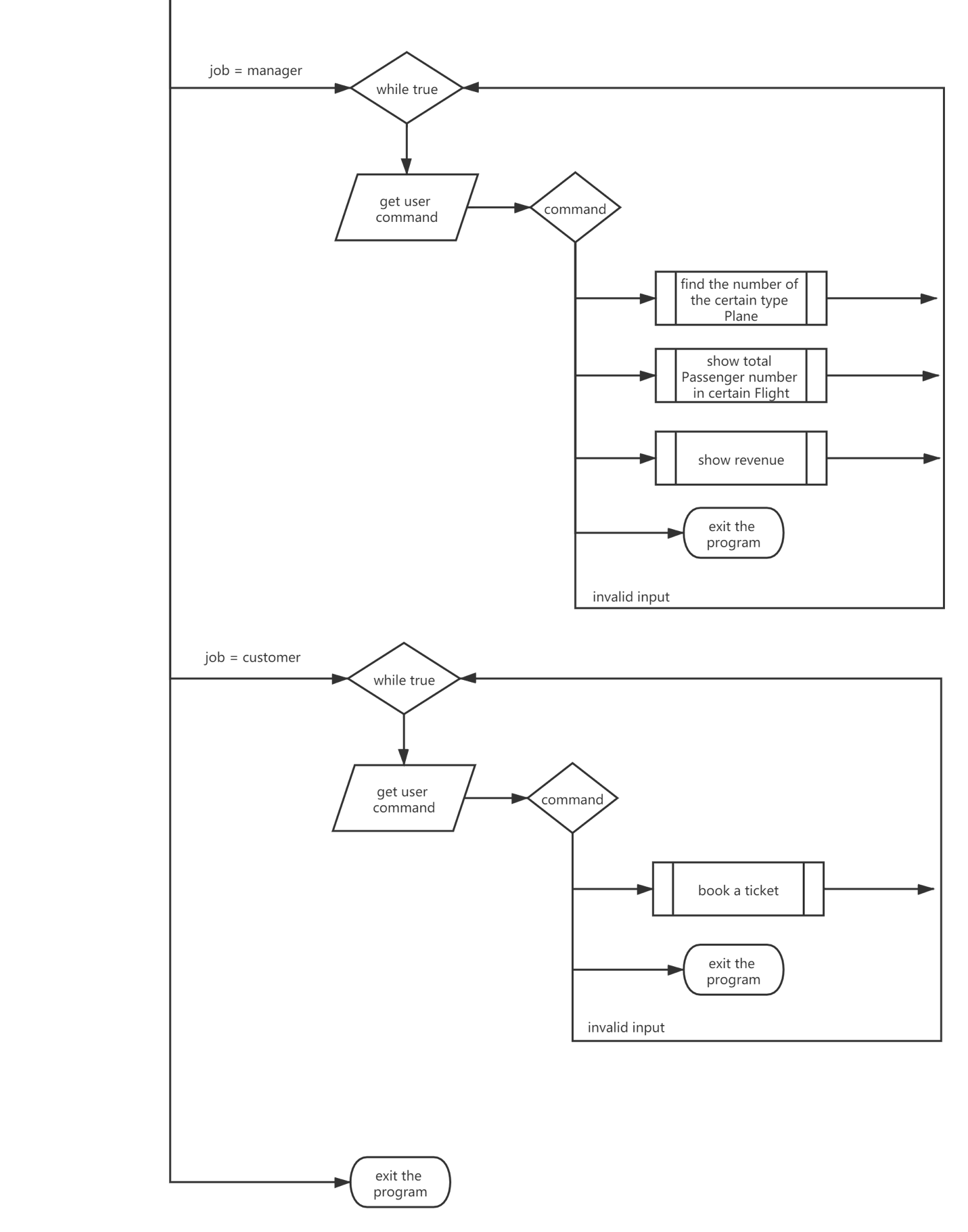




1. **Main loop design:**







# management of source code and lifecycle by software tools:

**management of source:**

our group use GitHub to maintain our code.

The website addresses which can read the code and document are shown as below:

The address of the code:

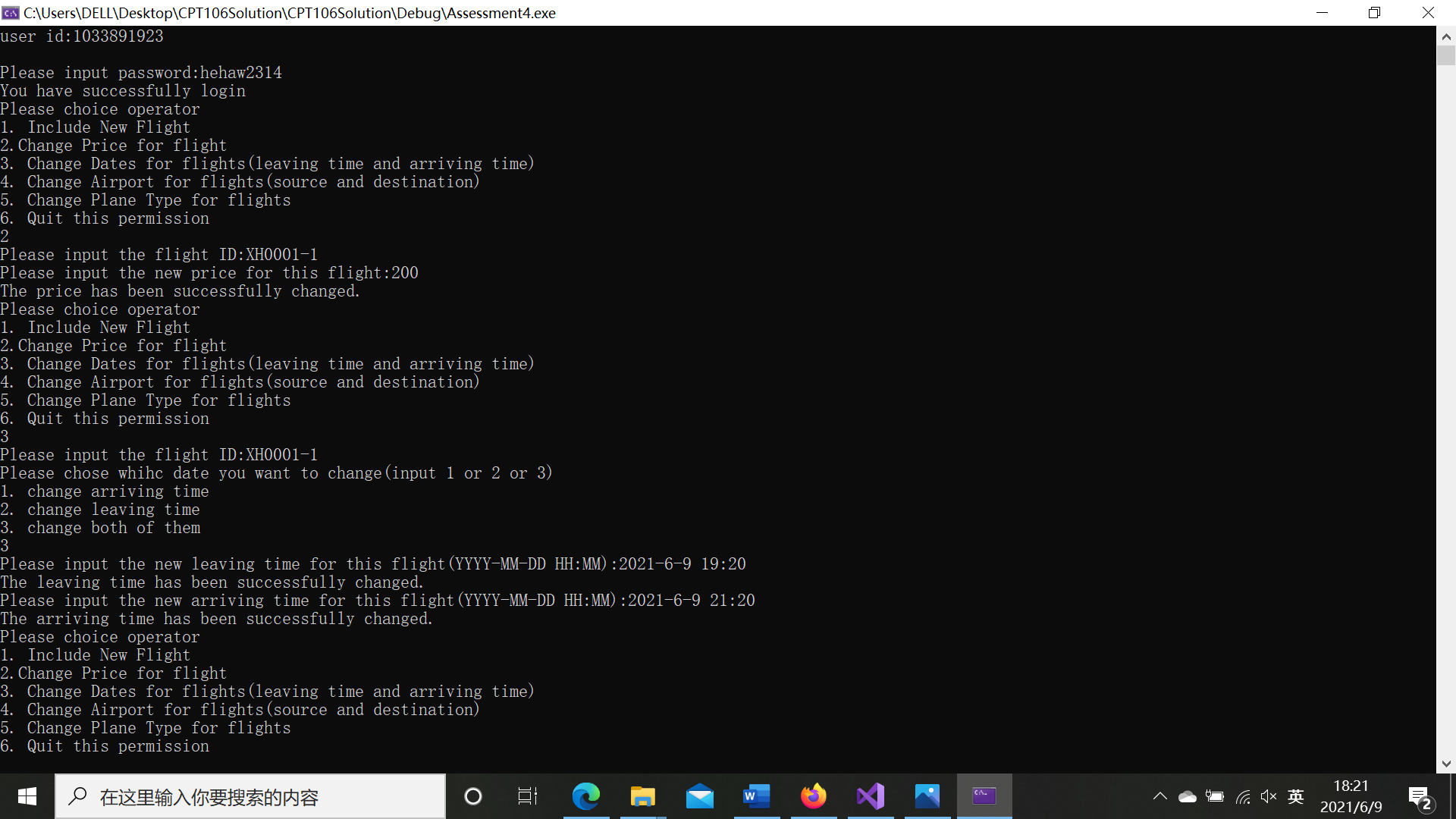
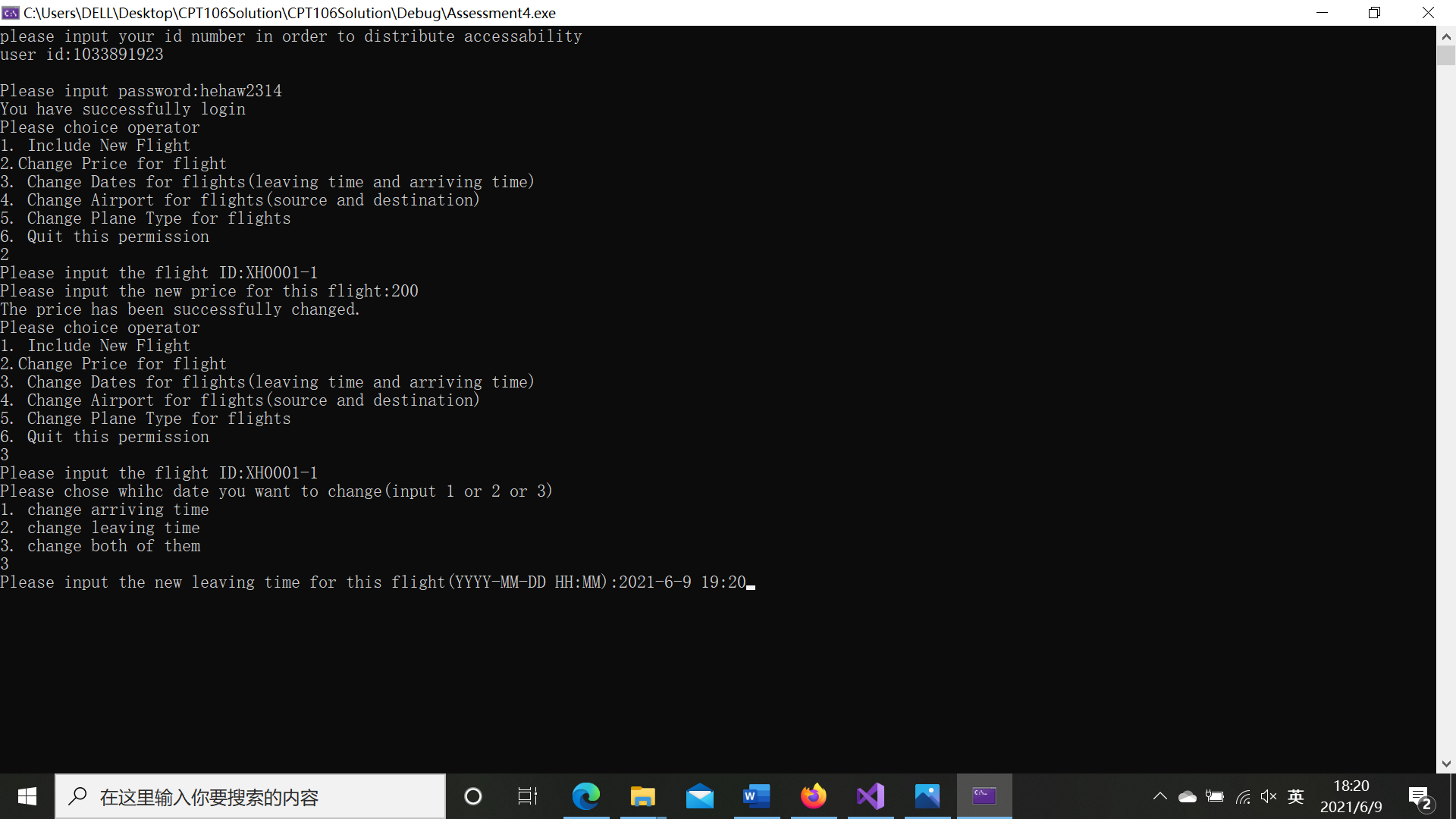
The address of the document:

**lifecycle of this program:**

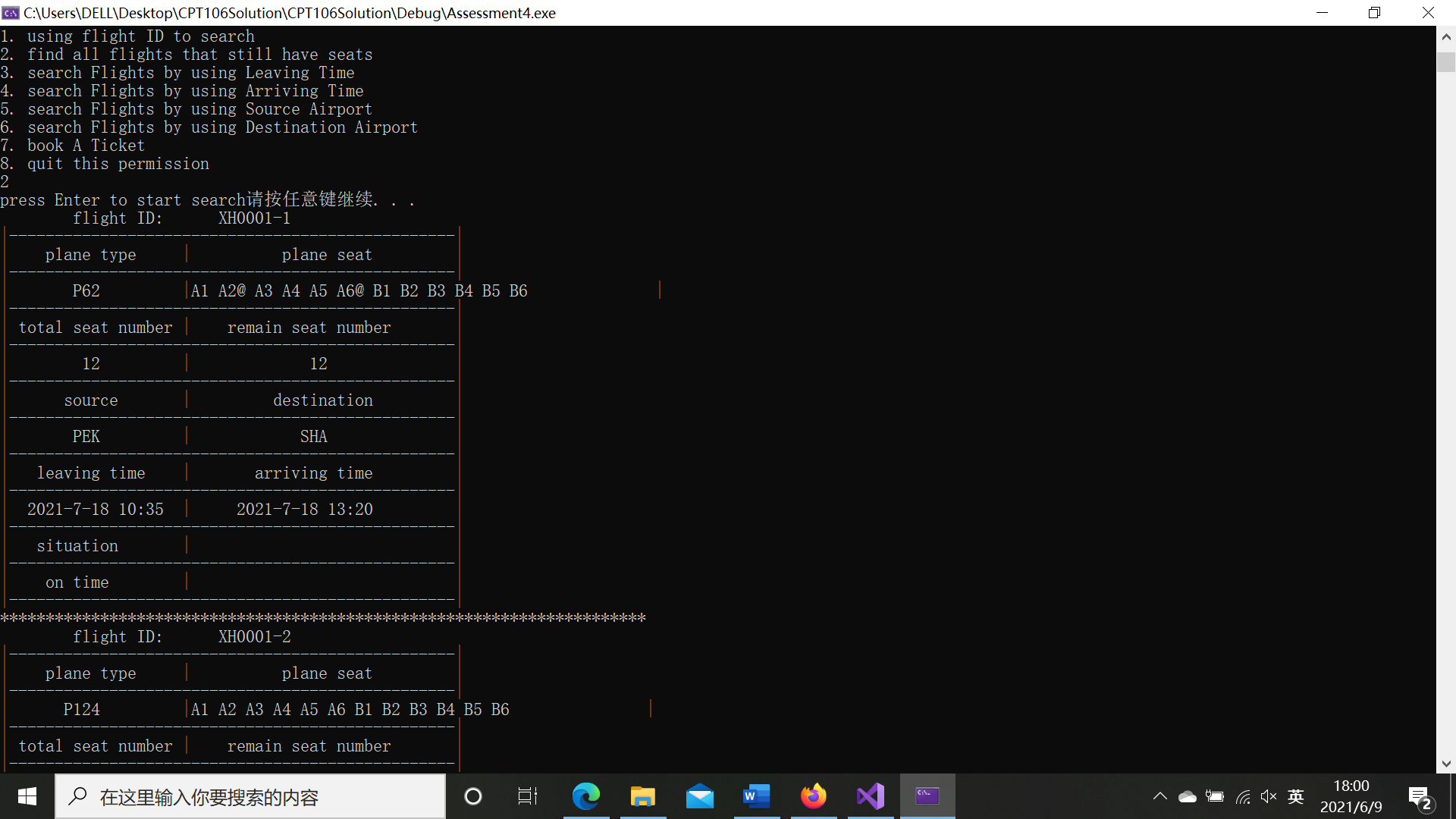
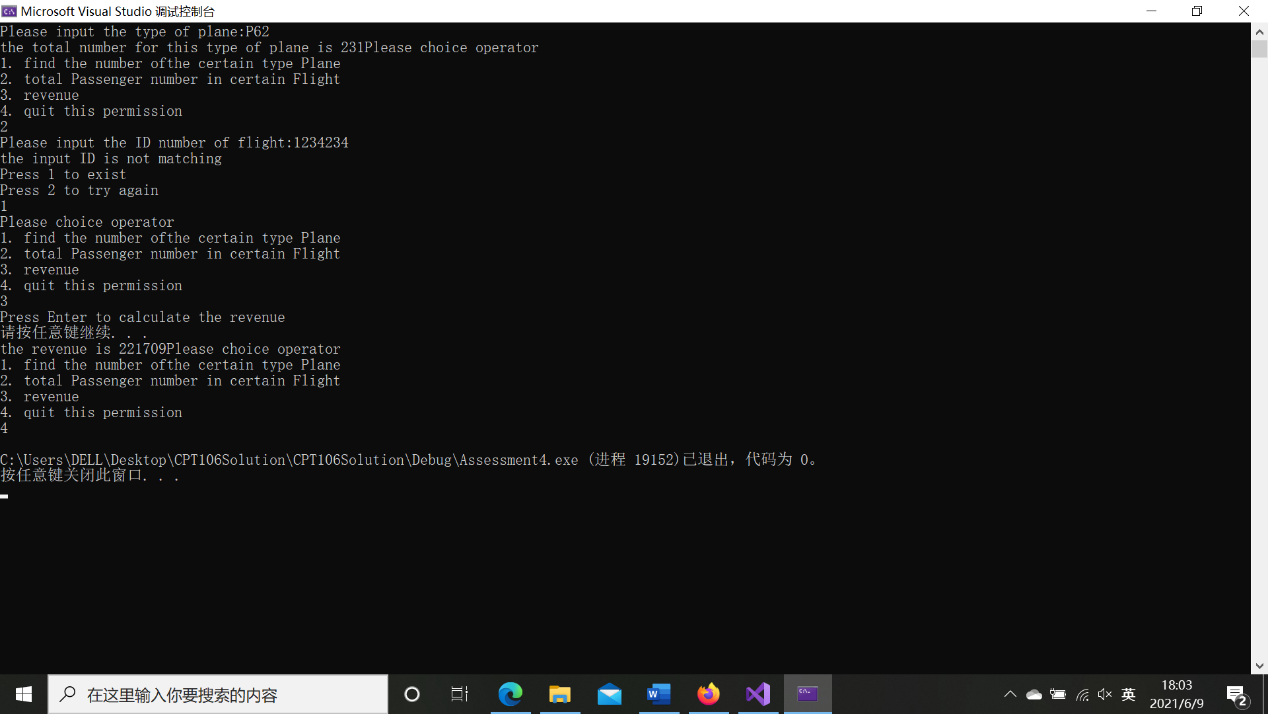
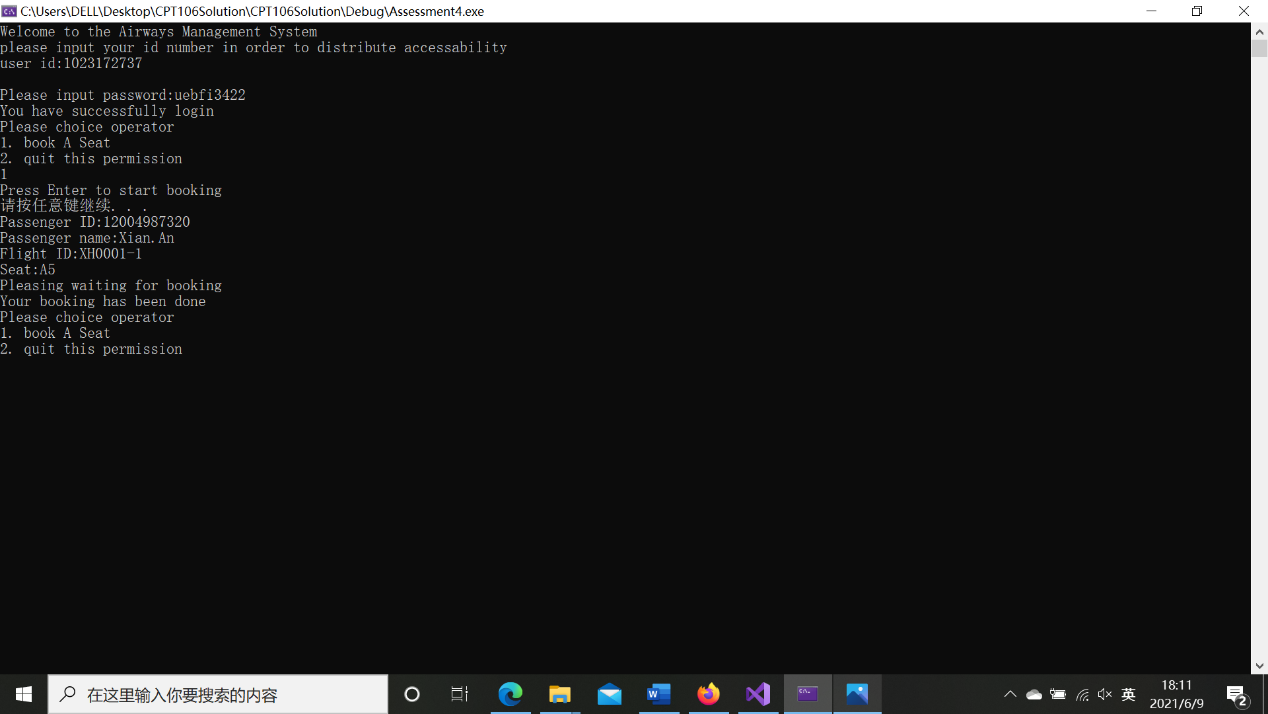
this program is designed due to the requirement of the airport company, which is convenient for the company to maintain all the information of users and flights. Moreover, after the code is maintained in GitHub, the programmer can see all the changes after it was uploaded. Since it’s an open-source project, everyone can see the codes. Therefore, if there are some bugs and insufficient in the program and others users report them, the programmer can change it instantly. Maybe with the develop of machine learning and AI technology, this code don’t need to be improved by people, which means the information can be inputted automatically by computer and then we need to use another updated code. So, after this, the lifecycle of this product is finished.

# Test:

The testing is showed in below:

**For administration:**

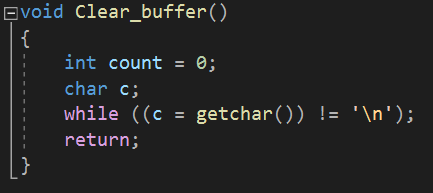
**For Agent:**

**For manager:****For customer:**

# Bug report:

There exist several bugs when I write this program. This part will discuss what these bugs are and how I fixed them.

I use type “int” to get the input in the interactive system. But there exists a problem when user input a string type value, such as “a”. The program will get into drop-dead halt. In order to find why this problem happened. I use the “step in” debugging method to find the reason. By looking at the memory and the value of the input. I find that the problem is the “cin”. The “cin” stream has some problem. Comparing “cin” stream with string stream, I decided that the problem is I didn’t clear the “cin” stream and clear the buffer. This cause the problem of the drop-dead halt. So we use function “clear()” and “Clear\_buffer()” to deal. However the function “Clear\_buffer()” is not exist in the standard library. The code for this function is shown below.



(This solved solution will not be discussed in this report because it seems a bit complex)

# User manual:

1. Administrator:

|  |  |
| --- | --- |
| Step | Describe |
| 1 | Input ID and passwords |
| 2 | Input correspond number to choose command   |  |  | | --- | --- | | 1 | Include new flight | | 2 | Change price for flight | | 3 | Change dates for flights | | 4 | Change airport for flights | | 5 | Change plane type for flights | | 6 | Quit | |

1. Agent

|  |  |
| --- | --- |
| Step | Describe |
| 1 | Input ID and students |
| 2 | Input correspond number to choose command   |  |  | | --- | --- | | 1 | Search by flight ID | | 2 | Search flights which still have seats | | 3 | Search flight by leaving time | | 4 | Search flight by arriving time | | 5 | Search flight by source airport | | 6 | Search flight by destination airport | | 7 | Book a ticket | | 8 | Quit | |

1. Manager

|  |  |
| --- | --- |
| Step | Describe |
| 1 | Input ID and students |
| 2 | Input correspond number to choose command   |  |  | | --- | --- | | 1 | Find the number of certain type plane | | 2 | Total passenger number in certain flight | | 3 | Show revenue | | 4 | Quit | |

1. Customer

|  |  |
| --- | --- |
| Step | Describe |
| 1 | Input ID and students |
| 2 | Input correspond number to choose command   |  |  | | --- | --- | | 1 | Book a ticket | | 2 | quit | |