CS1342 – Spring 2023

Program 4 – Flight Planner

In this program, you will be building a program that manages flight information at various airport hubs around the country! Your program must be submitted to GitHub by 6:00 AM CST on **Monday, April 17, 2023.** You will receive a 30 point penalty for any program submitted within 48 hours after the due date. Any programs submitted after 6:00 AM CST on **Wednesday, April 19, 2023** will receive a 0.

Overview

Southwest Airlines recently faced challenges with their flight planning software and have called upon you to create a new one from scratch! All of their data has been exported to a file called ***flights.txt*** in which your program will begin reading from to populate flight and destination information, then perform the proper steps to output the status of each flight destination.

Input File

The input file will contain on each line a command in which your program will perform a particular task. The possible commands are:

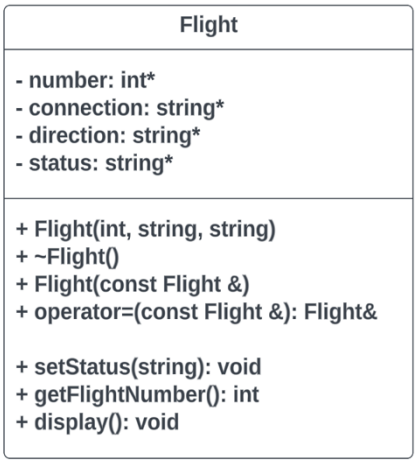
1. **CREATE DESTINATION [code] [name]**: this command will create a new destination object and add it to your flight planner. You will use the code (a single 3 character identifier) and the name as information for the destination.
2. **CREATE FLIGHT [from] [to] [flightNumber]:** this command will add a new flight to the "from" destination and "to" destination.
3. **UPDATE FLIGHT [flightNumber] [status]:** this command will update the status of a flight. A flight is either "ON TIME" or "DELAYED".
4. **DISPLAY STATUS [code]:** This command will display the status of all flights at a single destination.

To build your flight planner, you will be creating 3 classes: **Flight, Destination, and FlightPlanner.**

Program Specifications

Now let's take a look at how you will build the flight planner! Let's start from the ground up and look at the **Flight** class:

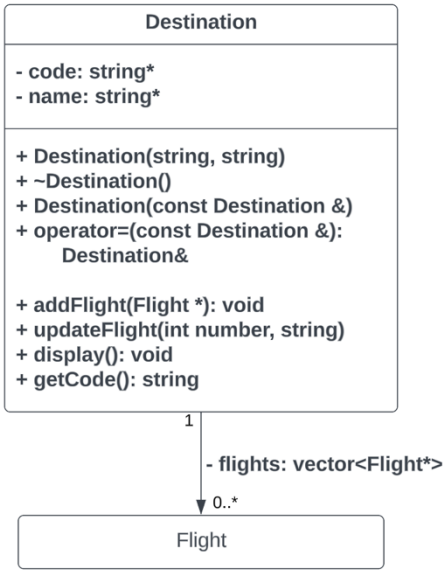
Flight (.h / .cpp)
Your flight class contains all information and logic for a single outbound or inbound flight. Your flight class will be customized using the following data members (**NOTE: ALL data members must be dynamically created on the heap!**):



Data Members
- **connection:** the 3 digit airport code where the flight is traveling to or from.
- **direction:** will be set to either "INBOUND" or "OUTBOUND".
- **status:** will be "ON TIME" or "DELAYED

Member Functions
- An overloaded constructor that initializes *number, connection, and direction.* It also sets the status to "ON TIME" by default.
- **setStatus(string)** – Updates the status of the flight with the value passed in.
- **getFlightNumber() –** returns the number for the flight as an integer.
- **display() –** prints the flight information to the screen (see formatting below in example output)
- **Implement the rule of 3!** (you will also implement the Rule of Three for this class. The destructor, copy constructor and copy assignment operator).

Destination (.h / .cpp)
Next you will create a destination class which will contain information about a particular airport that you service.

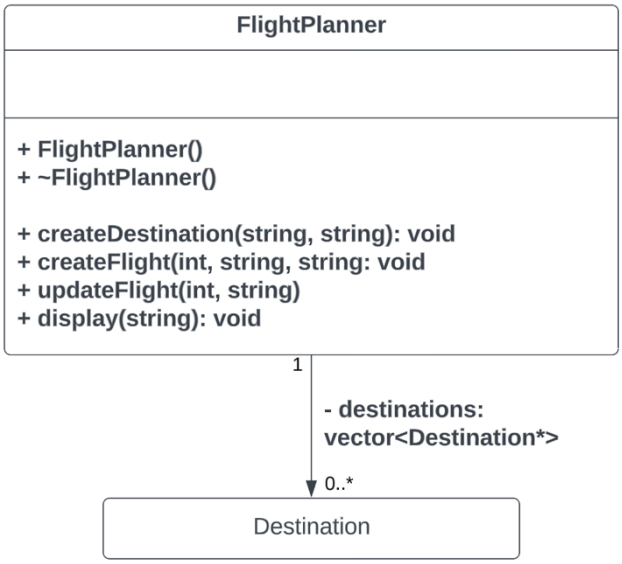


Data Members
- **code:** the 3 character airport code for this destination (e.g. "DFW")
- **name:** the full name of the airport
- **flights:** a vector containing Flight object pointers.

Member Functions
- An overloaded constructor that initializes the code and name of the destination.
- **addFlight(Flight * newFlight):** will add the newFlight object pointer to the vector of flights
- **updateFlight(int number, string status):** will linear search your vector of flights for the one matching the "number" passed in. If it finds a flight that matches, call the Flight object's **setStatus(string)** function.

- **display():** a void function that displays the destination code and name along with all inbound and outbound flights (see format below in sample output).
- **getCode():** a getter function for the airport code.
- **Implement the rule of 3!** (you will also implement the Rule of Three for this class. The destructor, copy constructor and copy assignment operator).

FlightPlanner (.h / .cpp)
Now it's time to put it all together in your FlightPlanner class!



Data Members

- **destinations:** A vector of pointers to Destination objects.

Member Functions
- A default constructor (can be blank)
- A destructor – will loop through all destination pointers and delete each one

- **createDestination(string code, string name):** will create a destination object on the heap, and add the pointer to the vector

- **createFlight(int flightNumber, string to, string from):** Will create and add the flight info to the "to" and "from" destinations. To do so perform the following steps:
    1. Create an inbound flight object (on the heap): use the *flightNumber*, *from,* and set the direction to "INBOUND"
    2. Create an outbound flight object (on the heap): use the *flightNumber, to,* and set the direction to "OUTBOUND"
    3. Linear search your list of destinations and find the one that matches the *to* code. Add the *inbound* flight to that destination using the **addFlight()** function.
    4. Linear search your list of destinations and find the one that matches the *from* code. Add the *outbound* flight to that destination using the **addFlight()** function.
- **updateFlight(int flightNumber, string status):** Iterate over your destinations and call the **updateFlight()** function on the destination.
- **display(string airportCode):** Linear search your destinations and call the **display()** method on the destination that matches the airportCode pass in as an argument.

Main.cpp
Create a main.cpp that reads in the input file and uses a flight planner object to manage your flights! You will create one flight planner object and for each command read in the file call the corresponding function using your flight planner class.

Example Input (file)

CREATE DESTINATION AUS Austin International Airport
CREATE DESTINATION DFW Dallas Fort-Worth International Airport
CREATE DESTINATION CHI Chicago
CREATE FLIGHT DFW AUS 6488
CREATE FLIGHT AUS CHI 5544
CREATE FLIGHT CHI DFW 2342
UPDATE FLIGHT 2342 DELAYED
UPDATE FLIGHT 5544 ON TIME
UPDATE FLIGHT 6488 ON TIME
DISPLAY STATUS AUS
DISPLAY STATUS DFW
DISPLAY STATUS CHI

Example Output (console)

```
AUS: Austin International Airport
        INBOUND Flight # 6488 traveling from DFW is ON TIME
        OUTBOUND Flight # 5544 traveling from CHI is ON TIME

DFW: Dallas Fort-Worth International Airport
        OUTBOUND Flight # 6488 traveling from AUS is ON TIME
        INBOUND Flight # 2342 traveling from CHI is DELAYED

CHI: Chicago
        INBOUND Flight # 5544 traveling from AUS is ON TIME
        OUTBOUND Flight # 2342 traveling from DFW is DELAYED


Process finished with exit code 0
```

Grading

Your program will be graded on the following criteria

- Flight class implementation **15 points**
- Destination class implementation **15 points**
- FlightPlanner class implementation **20 points**
- Correct implementation of Rule of 3 **10 points**
- Data Members and objects are dynamically allocated on the heap **20 points**
- A working main.cpp **20 points**