Class Outline

Airport Terminal Simulation
Programming Assignment 2

4 April 2021

Prepared By
Nolan Anderson
npa0002@uah.edu

Prepared for
Dr. Jacob Hauenstein
CS 307, Object Oriented Programming
Computer Science Department
University of Alabama in Huntsville

1.0 System Overview

To solve the problem of parsing XML files comprised of a large number of different flights (consists of departure time and city, destination city, aircraft type etc.) requires the creation of multiple classes. This software will largely be ran by the class outlined in section 3.0, with the following classes (section 4.0-6.0) parsing the data. In other words, sections 4.0-6.0 will provide section 3.0 with the data it needs to do its calculations and output. Mostly, sections 4.0-6.0 call the parsing functions and return data and they are much simpler than section 3.0. The output of the software is simple. It outputs all of the data of each flight every 5 seconds. This data includes the flight number, aircraft name and type, departure time and city, destination city among many other different data points. In summary, the main functionality of the program is to send out flight data. If you were the flight controller for every flight in the United States, this would provide you with every flight that is going out or currently in transit.

2.0 **Relevant Terms and Acronyms**

Class – This is a user-defined data type that we can use to hold member variables and functions.
Member Variables – The pieces that make up a class.
Member Functions – The "doers" of the class, as in they perform the calculations, output etc.
InFile – A standard name used for input file names.
N/a – This thing or value does not exist and is not needed in this function.
Void – A function type that does not need to return a value. Usually a function that creates, or performs output.

3.0 Flight Simulator Executor

3.1 Class FlightSim

3.1.1 Member Variables

Multiplier – Variable to switch how quickly the simulation runs (1X 2X 3X speed).
InFile – .txt file name to get the data for the program to parse.
FlightNum – The number of the flight, used in other functions to get data.

3.1.2 Member Functions

3.1.2.1 - FlightSim::FlightSim();
Actions Performed – Creates a new instance of the fligh simulation.
Calls 3.1.2.3, 3.1.2.4, and 3.1.2.5.
Arguments – N/a.
Return Value – N/a.

3.1.2.2 - ~FlightSim::FlightSim();
Actions Performed – Deconstructs the current instance of the simulation.
Arguments – N/a.
Return Value – N/a.

3.1.2.3 - int SetMultiplier();
Actions Performed – Sets the time multiplier for the program output.
Arguments – N/a.
Return Value – Integer of the multiplier.

3.1.2.4 - string   SetInFile();
Actions Performed – Sets the name of the input file.
Arguments – N/a
Return Value – Returns a string of the name of the input file.

3.1.2.5 - void   Start(int Multiplier, string InFile);
Actions Performed – The main function that starts and runs the simulation.
Arguments – The multiplier speed and InputFile name.
Return Value – Void.

3.1.2.6 - int   CurrentLocation(int FlightNum, int CurrentHr, int CurrentMin);
Actions Performed – Calculated the current location of the flight.
Arguments – Flight number, and the simulations current time in hours and minutes.
Return Value – Returns an integer of the location.

3.1.2.7 - int   FlightTime(int FlightNum);
Actions Performed – Returns the projected flight time.
Arguments – The flight number.
Return Value – An integer of the projected flight time.

3.1.2.8 - void   OutNewFlight(int FlightNum);
Actions Performed – Outputs a new flight.
Arguments – The flight number.
Return Value – Void.

3.1.2.9 - void   OutInterval(int FlightNum);
Actions Performed – Outputs all of the current flights, intervals.
Arguments – The flight number.
Return Value – Void.

There will be get and set functions defined for each private member variable.

4.0 Populating City Data from XML Files.

4.1 Class CityData

4.1.1 Member Variables

CityName – The Name of the City
StateName – Name of the state the city is in.
CityLat – Latitude of the City.
CityLon – Longitude of the City.
Distance – Distance to the other City.

4.1.2 Member Functions

4.1.2.1 - CityData::CityData(string InFile);
Actions Performed – Constructs a new instance of the city data.
Arguments – The name of the input file.
Return Value – N/a.

4.1.2.2 - ~CityData::CityData();
Actions Performed – Deconstructs the current instance of the City Data class.
Arguments – N/a.
Return Value – N/a.

4.1.2.3 - void SetData(string InFile);
Actions Performed – Calls the XML parser to set data for section 4.1.1.
Arguments – The name of the input file.
Return Value – Void

4.1.2.4 – int CalculateDistance(char *depCity, char *arrCity)
Actions Performed – Calculates the distances between two cities
Arguments – Characters for the city / state symbols
Return Value – Int for distance

4.1.2.5 – double TripTime(double distance, double CruiseSpeed)
Actions Performed – Calculates overall trip time
Arguments – Distance and cruise speed
Return Value – Returns the amount of time to get to destination

4.1.2.6 – double DistFromStart(double distance, double TripTime, double Elapsed)
Actions Performed – Calculates distance from start
Arguments – Distance, triptime, and elapsed time.
Return Value – Returns the distance from the start.

4.1.2.7 – double DistToDst(double DistFromStart, double calcDistance)
Actions Performed – Calculates distance to destination.
Arguments – Distance from start and calculated distance
Return Value – The distance to destination airport

There will be get and set functions defined for each private member variable.

5.0 Populating Flight Data from XML files.

5.1 Class FlightData

5.1.1 Member Variables

Airline – Name of the airline
AircraftType – The type of the aircraft.
DepartureCity – The city the FlightNum is departing from.
DestinationCity – The intended arrival city of FlightNum.
FlightNum – The number of the flight.

5.1.2 Member Functions

5.1.2.1 - FlightData::FlightData(string InFile);
Actions Performed – Constructor for the flight data, calls SetData.
Arguments – Name of the input file.
Return Value – N/a.

5.1.2.2 - ~FlightData::FlightData();
Actions Performed – Deconstructor for the current instance of the FlightData class.
Arguments – N/a.
Return Value – N/a.

5.1.2.3 - void SetData(string InFile);
Actions Performed – Calls the parsing functions to populate 5.1.1 Variables.
Arguments – Name of the input file.
Return Value – Void.

5.1.2.4 – double CurrentLat(double lat1, double lat2, double elapsedtime, double TripTime)
Actions Performed – Calculates current latitude.
Arguments – Destination lat, arrival lat, elapsed time, and total trip time.
Return Value – Current latitude

5.1.2.5 – double CurrentLon(double lon1, double lon2, double elapsedtime, double TripTime)
Actions Performed – Calculates current longitude.
Arguments – Destination lon, arrival lon, elapsed time and trip time.
Return Value – Current longitude.

5.1.2.6 – double CurrentAlt(double ElapsedMin, double ROC)
Actions Performed – Calculated current altitude
Arguments – Elapsed time and rate of climb
Return Value – Current altitude value.

There will be get and set functions defined for each private member variable.

6.0 Parsing Aircraft Data from XML files.

6.1 Class AircraftData

6.1.1 Member Variables

Make – The make of the aircraft.

Model – The model of the aircraft.
Speed – The flight speed of the aircraft.
ClimbSpeed – The speed at which the aircraft gets to cruising altitude.
WingSpan – The wingspan of the aircraft.
FuselageLength – The length of the fuselage.

6.1.2 Member Functions

6.1.2.1 - AircraftData::AircraftData(string InFile);
Actions Performed – Constructor for the Aircraft data.
Arguments – The name of the input file in terms of a string.
Return Value – No return value, constructor.

6.1.2.2 ~AircraftData::AircraftData();
Actions Performed – Deconstructor for the aircraft data, will clear the data parsed.
Arguments – N/a.
Return Value – No return value, deconstructor.

6.1.2.3 - void SetData(string InFile);
Actions Performed – Sets the data of variables in section 6.1.1 by parsing the data in InFile.
Arguments – The name of the input file in terms of a string.
Return Value - Void

There will be get and set functions defined for each private member variable.

7.0 Parsing Flight Data XML File

7.1 Class Flight Data Parser

7.1.1 Member Variables

```
char m_sFlightDataFile[32]; // Name of the flight data file
int  m_iStartHour;          // Hour the scenario starts (24 hr clock)
int  m_iStartMin;           // Minute the scenario starts
int  m_iAircraftCount;      // Number of aircraft types
int  m_iFlightCount;        // Number of flights
bool m_bDataFileOK;         // Flag we have a good data file
fstream       *inFile;
```

7.1.2 Member Functions

7.1.2.1 FlightDataParser();

Actions Performed – Constructor for the Flight Data Parser
Arguments – None
Return Value – No return value, constructor.

7.1.2.2 - void InitFlightData(const char *dataFile);

Actions Performed – Initializes the flight data input file.
Arguments – The name of the input file in terms of a string.
Return Value – No return value, void.

7.1.2.3 - void getStartTime(int *hr, int *min);

Actions Performed – Returns the start time from the xml file.
Arguments – Pointer to the hour and minute private variables.
Return Value – No return value, void.

7.1.2.4 - int getAircraftCount();
Actions Performed – Returns the number of aircraft.
Arguments – None
Return Value – The number of aircraft.

7.1.2.5 - int getFlightCount();
Actions Performed – Returns number of flights
Arguments – None
Return Value – Returns number of flights

7.1.2.6 - bool getAircraftData(AircraftType T, char *make, char *desc,
double *roc, double *wngs, double *len,
double *cs, double *ca);

Actions Performed – Initializes the data by searching through stored data in xml file.
Arguments – Pointers to private variables that describe the aircraft.
Return Value – Boolean on whether or not it was successful.

7.1.2.7 - bool getFlightData(char *airline, char *plane,
int *flNum, char *departCity, int *depHr,
int *depMin, char *destCity);

Actions Performed – Initializes the data by searching through stored data in xml file.
Arguments – Pointers to private variables that describe the flights.
Return Value – Boolean on whether or not it was successful.

7.1.2.8 - bool getNextLine(char *buffer, int n);
Actions Performed – Finds the next line in the xml file.
Arguments – Buffer character and n for line count.
Return Value – Boolean for correctly executed.

7.1.9 - ~FlightDataParser();
Actions Performed – Deconstructor
Arguments – None
Return Value – None

8.0 Parsing City Data XML Files

8.1 Class City Data Parser

8.1.1 Member Variables

int m_iCityCount;
char m_sDataFile[32];
bool m_bDataFileOK;
fstream *inFile;

8.1.2 Member Functions

8.1.2.1 CityDataParser();

Actions Performed – Constructor
Arguments – None
Return Value – None, constructor.

8.1.2.2 - void InitCityData(const char *dataFile);

Actions Performed – Initializes city data.
Arguments – Input file
Return Value – Void

8.1.2.3 - void getCityTime()

Actions Performed – Finds the time of the cities
Arguments – None
Return Value – Void

8.1.2.4 – bool getCityData(char *name, char *state, char *symbol, double *lat, double *lon);

Actions Performed – Sets pointers to city data, usually private variables in a class.
Arguments – Pointers to variables that describe each city.
Return Value – Boolean for status of completion.

8.1.2.5 – void getCitySymbolsArray(char ***array);

Actions Performed – Initializes a set of characters for city symbols.
Arguments – Char array
Return Value – None, sets a pointer.

8.1.2.6 - bool getNextLine(char *buffer, int n);

Actions Performed – Goes to next line of the xml file
Arguments – Buffer character and line number.
Return Value – Boolean on successful / fail completion.

8.1.2.7 - ~CityDataParser();

Actions Performed – Deconstructor
Arguments – None
Return Value – None, deconstructor

9.0 Aircraft Factory

9.1 Class Aircraft Factory

9.1.1 Member Variables

AircraftType Type;
Vector<AircraftType> TypeList;

9.1.2 Member Functions

9.1.2.1 AircraftFactory();

Actions Performed – Constructor
Arguments – None
Return Value – None, constructor.

9.1.2.2 ~AircraftFactory();

Actions Performed – Deconstructor
Arguments – None
Return Value – None, Deconstructor.

9.1.2.3 - void SetData(string InFile);

Actions Performed – Calls the parsing functions to populate 9.1.1 Variables.
Arguments – Name of the input file.
Return Value – Void.

Each of the following classes (9.2-9.4) will be similar to 9.1, only changing in name. They will use the same constructor as 9.0, just using different values inside of the constructor. 9.0 will call to sections 9.2-9.4 to set the data.

9.2 Class Passenger
9.3 Class Business
9.4 Class Single Engine

There will be get and set functions for each of the private member variables.