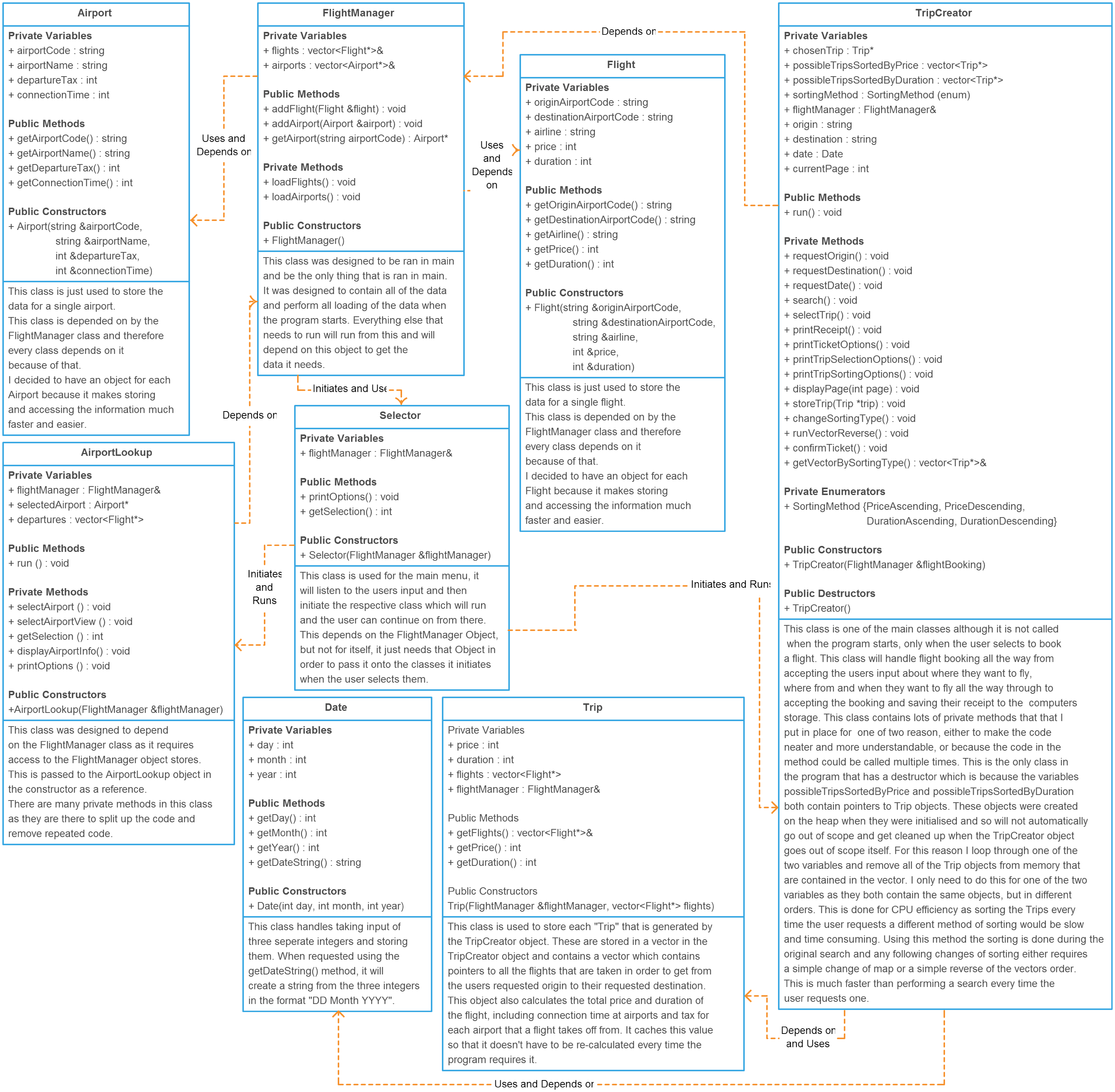

Airport

Private Variables
+ airportCode : string
+ airportName : string
+ departureTax : int
+ connectionTime : int

Public Methods
+ getAirportCode() : string
+ getAirportName() : string
+ getDepartureTax() : int
+ getConnectionTime() : int

Public Constructors
+ Airport(string &airportCode,
string &airportName,
int &departureTax,
int &connectionTime)

This class is just used to store the data for a single airport.
This class is depended on by the FlightManager class and therefore every class depends on it because of that.
I decided to have an object for each Airport because it makes storing and accessing the information much faster and easier.

FlightManager

Private Variables
+ flights : vector<Flight*>&
+ airports : vector<Airport*>&

Public Methods
+ addFlight(Flight &flight) : void
+ addAirport(Airport &airport) : void
+ getAirport(string airportCode) : Airport*

Private Methods
+ loadFlights() : void
+ loadAirports() : void

Public Constructors
+ FlightManager()

This class was designed to be ran in main and be the only thing that is ran in main.
It was designed to contain all of the data and perform all loading of the data when the program starts. Everything else that needs to run will run from this and will depend on this object to get the data it needs.

Flight

Private Variables
+ originAirportCode : string
+ destinationAirportCode : string
+ airline : string
+ price : int
+ duration : int

Public Methods
+ getOriginAirportCode() : string
+ getDestinationAirportCode() : string
+ getAirline() : string
+ getPrice() : int
+ getDuration() : int

Public Constructors
+ Flight(string &originAirportCode,
string &destinationAirportCode,
string &airline,
int &price,
int &duration)

This class is just used to store the data for a single flight.
This class is depended on by the FlightManager class and therefore every class depends on it because of that.
I decided to have an object for each Flight because it makes storing and accessing the information much faster and easier.

TripCreator

Private Variables
+ chosenTrip : Trip*
+ possibleTripsSortedByPrice : vector<Trip*>
+ possibleTripsSortedByDuration : vector<Trip*>
+ sortingMethod : SortingMethod (enum)
+ flightManager : FlightManager&
+ origin : string
+ destination : string
+ date : Date
+ currentPage : int

Public Methods
+ run() : void

Private Methods
+ requestOrigin() : void
+ requestDestination() : void
+ requestDate() : void
+ search() : void
+ selectTrip() : void
+ printReceipt() : void
+ printTicketOptions() : void
+ printTripSelectionOptions() : void
+ printTripSortingOptions() : void
+ displayPage(int page) : void
+ storeTrip(Trip *trip) : void
+ changeSortingType() : void
+ runVectorReverse() : void
+ confirmTicket() : void
+ getVectorBySortingType() : vector<Trip*>&

Private Enumerators
+ SortingMethod {PriceAscending, PriceDescending,
DurationAscending, DurationDescending}

Public Constructors
+ TripCreator(FlightManager &flightBooking)

Public Destructors
+ TripCreator()

This class is one of the main classes although it is not called when the program starts, only when the user selects to book a flight. This class will handle flight booking all the way from accepting the users input about where they want to fly, where from and when they want to fly all the way through to accepting the booking and saving their receipt to the computers storage. This class contains lots of private methods that that I put in place for one of two reason, either to make the code neater and more understandable, or because the code in the method could be called multiple times. This is the only class in the program that has a destructor which is because the variables possibleTripsSortedByPrice and possibleTripsSortedByDuration both contain pointers to Trip objects. These objects were created on the heap when they were initialised and so will not automatically go out of scope and get cleaned up when the TripCreator object goes out of scope itself. For this reason I loop through one of the two variables and remove all of the Trip objects from memory that are contained in the vector. I only need to do this for one of the two variables as they both contain the same objects, but in different orders. This is done for CPU efficiency as sorting the Trips every time the user requests a different method of sorting would be slow and time consuming. Using this method the sorting is done during the original search and any following changes of sorting either requires a simple change of map or a simple reverse of the vectors order. This is much faster than performing a search every time the user requests one.

Selector

Private Variables
+ flightManager : FlightManager&

Public Methods
+ printOptions() : void
+ getSelection() : int

Public Constructors
+ Selector(FlightManager &flightManager)

This class is used for the main menu, it will listen to the users input and then initiate the respective class which will run and the user can continue on from there. This depends on the FlightManager Object, but not for itself, it just needs that Object in order to pass it onto the classes it initiates when the user selects them.

AirportLookup

Private Variables
+ flightManager : FlightManager&
+ selectedAirport : Airport*
+ departures : vector<Flight*>

Public Methods
+ run () : void

Private Methods
+ selectAirport () : void
+ selectAirportView () : void
+ getSelection () : int
+ displayAirportInfo() : void
+ printOptions () : void

Public Constructors
+AirportLookup(FlightManager &flightManager)

This class was designed to depend on the FlightManager class as it requires access to the FlightManager object stores. This is passed to the AirportLookup object in the constructor as a reference.
There are many private methods in this class as they are there to split up the code and remove repeated code.

Date

Private Variables
+ day : int
+ month : int
+ year : int

Public Methods
+ getDay() : int
+ getMonth() : int
+ getYear() : int
+ getDateString() : string

Public Constructors
+ Date(int day, int month, int year)

This class handles taking input of three seperate integers and storing them. When requested using the getDateString() method, it will create a string from the three integers in the format "DD Month YYYY".

Trip

Private Variables
+ price : int
+ duration : int
+ flights : vector<Flight*>
+ flightManager : FlightManager&

Public Methods
+ getFlights() : vector<Flight*>&
+ getPrice() : int
+ getDuration() : int

Public Constructors
Trip(FlightManager &flightManager, vector<Flight*> flights)

This class is used to store each "Trip" that is generated by the TripCreator object. These are stored in a vector in the TripCreator object and contains a vector which contains pointers to all the flights that are taken in order to get from the users requested origin to their requested destination.
This object also calculates the total price and duration of the flight, including connection time at airports and tax for each airport that a flight takes off from. It caches this value so that it doesn't have to be re-calculated every time the program requires it.