

# Criterion C - Development

Kathapet Nawongs

**Overall Word Count:** 811 words

## Introduction

- For my project, Netbean was used to create functional codes that can store and calculate numerical data, and allow Java Swing tool to create the whole user interface of my program.
- This program will allow flight information to be inputted and calculated for my client's presentation.

Word count: 47 words

## All Techniques Used...

- Parameters passing
- Encapsulation
  - Privatizing variables
- For loop
- While loop
- Arraylist
- Nested loop
- Saving to a file
  - FileWriter
- Error handling
- Simple and compound selection
- Selection sort
- Linear search
- Joptionpane
- GUI tabs
- GUI popup menus
- Use of a flag value (eg. -999 or "not set yet")
- Overloaded constructor
- Parsing a file
  - StringTokenizer
- Inheritance
  - Abstraction
  - Inheritance
  - Super and Subclass

Word count: 56 words

# Structure of the program

## What

The central class is the GUI class which would allow the user to interact with the program and input the flights data - just like what GUI stands for "Graphics User Interface".

 mainGUI.java


There also a need for a Flight class which would allow the flight inputs to be stored as an ArrayList. Inheritance creates a hierarchy structure where the flight class is the parent class, and the domestic and international flight class is the child class.

 Flight.java

 DomesticFlight.java

 InternationalFlight.java

My other two classes are for the function of the GUI. The first is to search and sort my flights data, and the second is to save and read the flights into a file.

 SortAndSearchFlight.java

 SaveAndReadClass.java

## Why

**Inheritance** is needed to distinguish between the types of flight that my clients need to work with, which are domestic and international flights. Moreover, the international flight and domestic flight class can receive attributes from the flight class while having additional, unique attributes of its own such as having a visa.

Via **encapsulation**, the attributes of that “template” class can only be changed by the methods that is created publicly.

By splitting my program into multiple classes, I am able to have a template class using a **constructor**. This allows multiple instances of classes to be made, and other classes can access their methods.

The creation of sub classes also allows **abstraction**. In order for the debugging and maintenance to become easier due to the modularity of the program.

Word count: 240 words

## Data structures used

### What

#### 1. Arraylist

```
private ArrayList<InternationalFlight> flights = new ArrayList<InternationalFlight>();
```

#### 2. Files

```
String fileName = "Flight.txt";
FileWriter fw = new FileWriter(fileName);
fw.write("" + flights.size());
fw.write(":");
for(int i = 0; i < flights.size(); i++){
    fw.write("" + flights.get(i).getCityCode());
    fw.write(":");
    fw.write("" + flights.get(i).getCityDestination());
    fw.write(":");
    fw.write("" + flights.get(i).getTotalDistance());
    fw.write(":");
    fw.write("" + flights.get(i).getNumberOfFlights());
    fw.write(":");
    fw.write("" + flights.get(i).getTotalPassengers());
    fw.write(":");
    fw.write("" + flights.get(i).getTotalRevenue());
    fw.write(":");
    fw.write("" + flights.get(i).getRequireVisa());
    fw.write(":");
    if(flights.get(i).getRequireVisa() == true){
        fw.write("" + flights.get(i).getCountryDestination());
        fw.write(":");
    }
    else if(flights.get(i).getRequireVisa() == false){
        fw.write(" No visa required");
        fw.write(":");
    }
}
fw.close();
```

# Why

Arraylist can vary the amount of elements in the array, depending on how many flights my client work with. These flights can be iterated efficiently making it useful for sorting and searching the entire data structure.

**Example:** Flights with attributes of city code, country destination and etc.

Files can be used to save the data attributes and can be accessed later on when needed. This is viable due to the use of the class FileWriter and FileReader.

**Example:** Created a file for the flights called 'flight.txt'.

Word count: 91 words

## Main Unique Algorithms

### What

#### #1 - File writing<sup>1</sup>

Pseudocode for SaveFlights (not included in the word count)

```
fileName = "Flight.txt"
FileWriter takes in fileName
write "" + flights.size
write ":"
loop i from 0 to flights.size
    write "" + CityCode in i in flights
    write ":"
    write "" + CityDestination in i in flights
    write ":"
    write "" + TotalDistance in i in flights
    write ":"
    write "" + NumberOfFlights in i in flights
    write ":"
    write "" + TotalPassengers in i in flights
    write ":"
    write "" + TotalRevenue in i in flights
```

---

<sup>1</sup> Refer to SaveAndRead Class

```

write ":"
write "" + RequireVisa in i in flights
if RequireVisa in i in flights = true then
    write "" + CountryDestination in i in flights
    write ":"
else if RequireVisa in i in flights = true then
    write "No visa required"
    write ":"
close

```

## #2 - File reading<sup>2</sup>

Pseudocode for ReadFlights (not included in the word count)

```

new FileReader to read "Flight.txt"
assign FileReader to BufferedReader
read whole ArrayList with String ReadInFile
Initialize StringTokenizer(readInFile, ":")
numFlights = st.nextToken
Initialize ArrayList flight
Loop i from 0 to flight.size()
    Add new InternationalFlight to flight
    CityCode = StringTokenizer.nextToken
    CityDestination = StringTokenizer.nextToken
    NumberOfFlights = StringTokenizer.nextToken
    TotalPassengers = StringTokenizer.nextToken
    TotalRevenue = StringTokenizer.nextToken
    Distance = StringTokenizer.nextToken
    CountryDestination = StringTokenizer.nextToken
    Assign each attribute to flights
returns flights

```

## #3 - Other algorithm used includes **selection sort** and **sequential search**<sup>3</sup>

Pseudocode sorting flights by code (not included in word count)

```

Loop for int i from 0 to size of arraylist flights - 1
    minIndex = i
    Loop for int j (= i + 1) from 0 to size of arraylist flights
        If(flights[i].getCityCode.compareTo(flights[minIndex].getCityC
ode) < 0

```

---

<sup>2</sup> Refer to SaveAndRead Class

<sup>3</sup> Refer to SortAndSearchFlight class

```

        minIndex = j
    End if
    If minIndex != i
        InternationalFlight temp = flights[i]
        set i = flights.get(minIndex)
        set minIndex = InternationalFlight temp
    End if
End loop

```

Pseudocode searching flights by code (not included in word count)

```

key = user input
Loop for int i from 0 to size of arraylist flights
    If flights[i].getCityCode().equals(key)
        then return i
    End if
End loop
Then return -1

```

## Why

### #1 - File writing

FileWriter is a fundamental constructor used to create FileWriter object given a file name called "Flight.txt". The nested "for" loop can accurately write all of the flights that have been added into the file.

### #2 - File reading

FileReader is another fundamental constructor which indicates that the file "Flight.txt" will be read. Together, the BufferedReader and ReadInFile method can read the whole file in. The most unique method is the StringTokenizer. It takes in semicolon as a parameter hence it can distinguish between each parameters.

### #3 - Selection sort and sequential search

I chose the selection sort than the bubble sort because selection sort only swaps once for every pass. But for the bubble sort, it has to swap multiple times for only one pass. Therefore, it would be less time-consuming and more efficient.

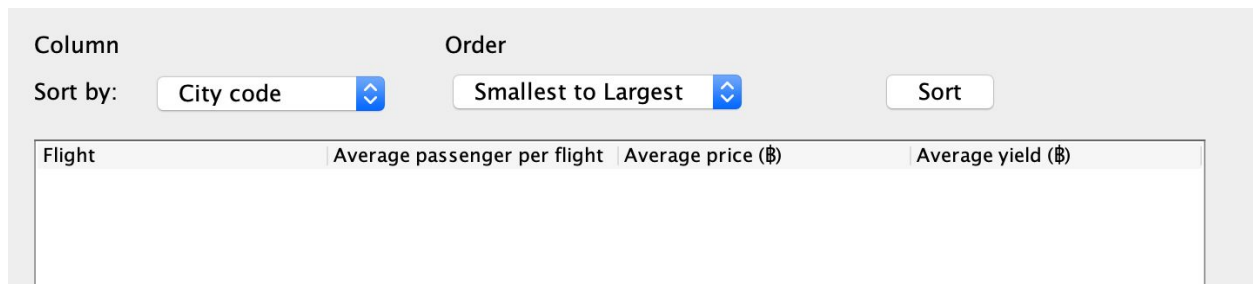
Sequential search is the better search algorithm in comparison to binary search because I know that the length of the arraylist will not be long. My client showed me a paper of all the flights that he will use for his presentation and it only contained 13 flights.

Word count: 203 words

## User Interface/GUI work

### What

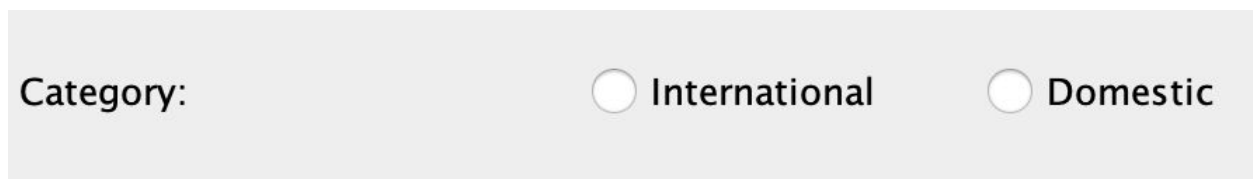
Combo boxes:



The screenshot shows a sorting interface with two dropdown menus and a button. The first dropdown is labeled 'Sort by:' and has 'City code' selected. The second dropdown is labeled 'Order' and has 'Smallest to Largest' selected. A 'Sort' button is to the right. Below these controls is a table with the following headers:

Flight	Average passenger per flight	Average price (€)	Average yield (€)
--------	------------------------------	-------------------	-------------------

Radio Button Group:

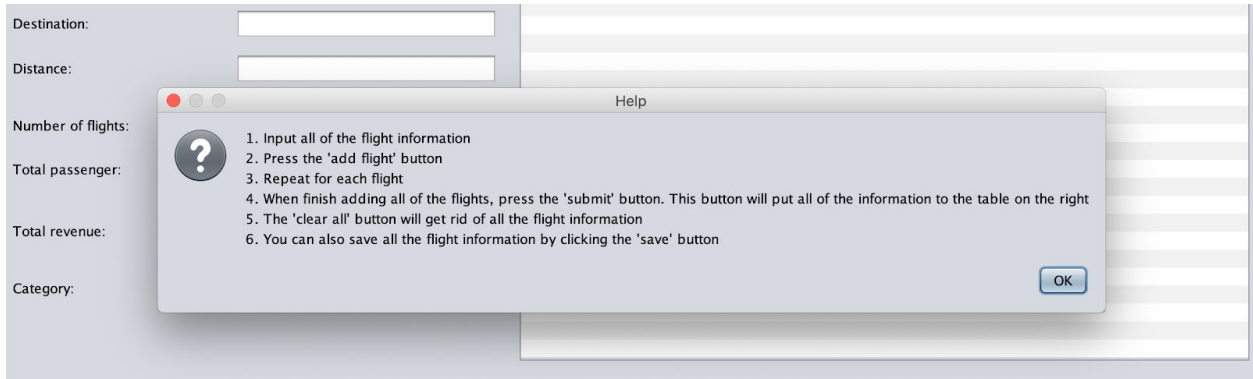


The screenshot shows a radio button group with the label 'Category:' followed by two options: 'International' and 'Domestic'. Both radio buttons are currently unselected.

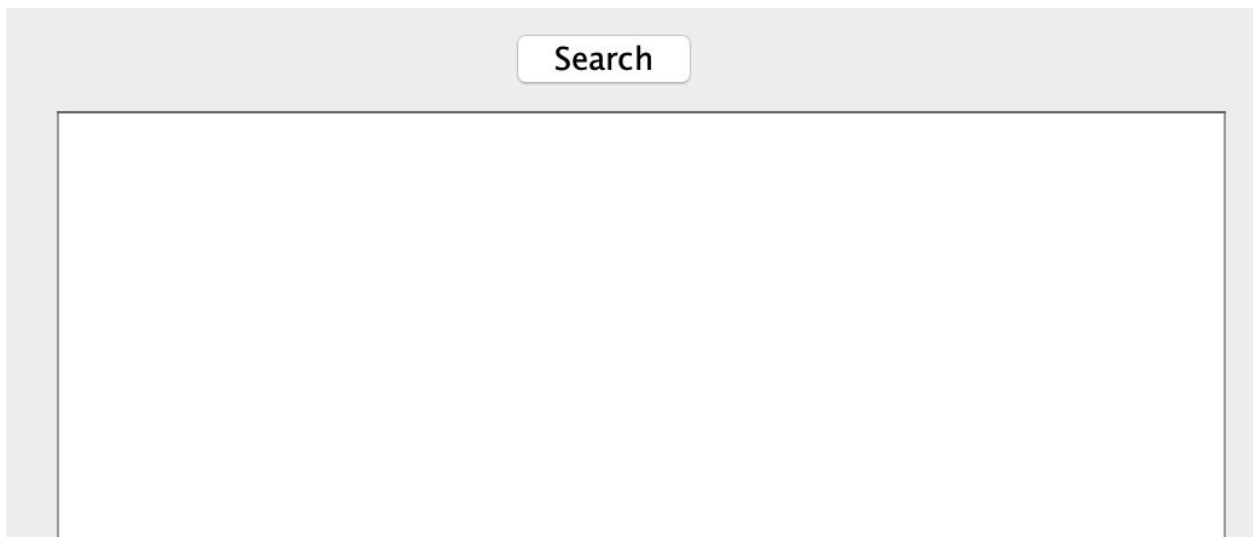
Table:

Flight	Destination	Distance	No. of Flt	Total pas.	Total Revenue	Country	Visa required
--------	-------------	----------	------------	------------	---------------	---------	---------------

OptionPane:



## TextField:



## Why

### **Combo boxes:**

Can select which Flight attribute to sort in the table and whether my client wants to prioritize the lowest or highest value first.

### **Radio button group:**

Can choose whether the flight is domestic or international to categorize the flight as either one.

### **Table:**

Represent the inputted data and the calculated data of the various flights. Making it easier for my client to work with those data.

### **OptionPane:**



Show a message to my client when he/she needs help. Can also give an error message when my client inputs incorrectly.

### TextField:

Another way of displaying data, but only for a specific flight that the client wants/searches for.

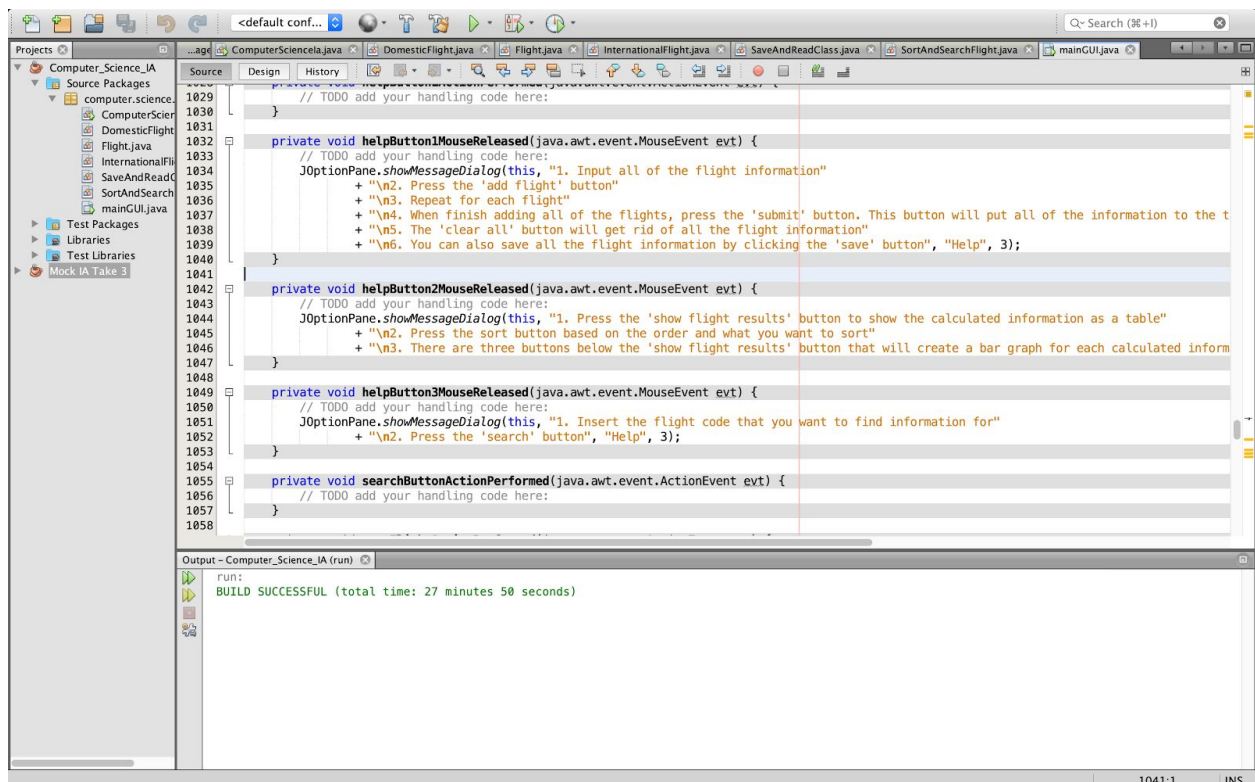
Word count: 119 words

## Software Tools Used

### What

#### Netbeans

- A popular integrated development environment for Java used programmers across the world.



# Why

1. The GUI components create a very convenient and not overly complicated interface for my client to use.
2. Have widely accessible pre-programmed codes that I used for graphs and files.
3. Can be coded in Java - a powerful and easy-to-follow programming language.

Word count: 55 words