# HPAir Flight Map

Generated by Doxygen 1.8.13

# Contents

# Chapter 1

# Class Index

## 1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

# Chapter 2

# File Index

## 2.1 File List

Here is a list of all documented files with brief descriptions:

# Chapter 3

# Class Documentation

## 3.1 City Struct Reference

### Public Attributes

- std::string **name**
- std::vector< Flight > **flights**
- bool **visited**

The documentation for this struct was generated from the following file:

- PA03/FlightMapV1.h

## 3.2 Flight Struct Reference

### Public Attributes

- City * **from**
- City * **to**
- unsigned int **id**
- unsigned int **cost**

The documentation for this struct was generated from the following file:

- PA03/FlightMapV1.h

## 3.3 FlightMap Class Reference

### Public Member Functions

- City * addCity (std::string &name)

  *Adds a new city to the map.*
- City * getCity (std::string &name)

  *Gets a city from the map with a specific name.*
- std::stack< Flight * > getPath (City *from, City *to)

  *Finds path of flights to get from one city to another.*

### 3.3.1 Member Function Documentation

#### 3.3.1.1 addCity()

```
City * FlightMap::addCity (
            std::string & name )
```

Adds a new city to the map.

**Parameters**

| | | |
|---|---|---|
| in | *name* | Name of the new city |

**Returns**

Pointer to the new city

#### 3.3.1.2 getCity()

```
City * FlightMap::getCity (
            std::string & name )
```

Gets a city from the map with a specific name.

**Parameters**

| | | |
|---|---|---|
| in | *name* | The name of the city to search for |

**Returns**

Pointer to the found city, or nullptr if not found

#### 3.3.1.3 getPath()

```
std::stack< Flight * > FlightMap::getPath (
            City * from,
            City * to )
```

Finds path of flights to get from one city to another.

**Parameters**

| | | |
|---|---|---|
| in | *from* | Pointer to the city to start searching from |
| in | *to* | Pointer to the city to search for |

**Algorithm**

Starting with from, searches through each adjacent unvisited city and then repeats until it either finds to or runs out of adjacent cities

**Returns**

A stack of flights from the origin city to the destination city Or an empty stack if no path could be found

The documentation for this class was generated from the following files:

- PA03/FlightMapV1.h
- PA03/FlightMapV1.cpp

# Chapter 4

# File Documentation

## 4.1 PA03/FlightMapV1.cpp File Reference

FlightMap class implementation.

```
#include "FlightMapV1.h"
```

### Functions

- void log (const std::string &text, bool newLine)
  *Logs information to file defined in LOG_FILE.*

### 4.1.1 Detailed Description

FlightMap class implementation.

**Version**

1.00 Alexander Novotny First version

### 4.1.2 Function Documentation

#### 4.1.2.1 log()

```
void log (
            const std::string & text,
            bool newLine )
```

Logs information to file defined in LOG_FILE.

**Parameters**

| | | |
|---|---|---|
| in | *text* | Text to log |
| in | *newLine* | Whether std::endl should be printed at the end. Defaults to true |

**Note**

If LOG_FILE not defined, nothing will happen

## 4.2 PA03/FlightMapV1.h File Reference

Header for FlightMap implementation.

```
#include <string>
#include <vector>
#include <stack>
#include <fstream>
```

### Classes

- struct City
- struct Flight
- class FlightMap

### Macros

- #define **LOG_FILE** "log.txt"

### Functions

- void log (const std::string &, bool=true)

  *Logs information to file defined in LOG_FILE.*

### 4.2.1 Detailed Description

Header for FlightMap implementation.

FlightMap class declaration, as well as City and Flight structs and log function used by FlightMap

**Version**

1.00 Alexander Novotny First version

### 4.2.2 Function Documentation

#### 4.2.2.1 log()

```
void log (
            const std::string & text,
            bool newLine )
```

Logs information to file defined in LOG_FILE.

**Parameters**

| | | |
|---|---|---|
| in | *text* | Text to log |
| in | *newLine* | Whether std::endl should be printed at the end. Defaults to true |

**Note**

If LOG_FILE not defined, nothing will happen

## 4.3 PA03/PA03.cpp File Reference

Program driver implementation file.

```
#include "PA03.h"
```

### Functions

- int **main** ()
- void loadCities (FlightMap &map, const std::string &fileName)
  *Loads a list of cities from fileName and adds them to map.*
- void loadFlights (FlightMap &map, const std::string &fileName)
  *Loads a list of flights from fileName and adds them to map.*
- void handleRequests (FlightMap &map, const std::string &fileName)
  *Loads a list of requests from fileName and then processes them.*
- void printFlightPlan (std::stack< Flight * > flightPlan)
  *Takes a flight plan and prints it on screen.*

### 4.3.1 Detailed Description

Program driver implementation file.

Reads in city and flight information, and then processes flight requests

**Version**

1.00 Alexander Novotny First version

### 4.3.2 Function Documentation

#### 4.3.2.1 handleRequests()

```
void handleRequests (
            FlightMap & map,
            const std::string & fileName )
```

Loads a list of requests from fileName and then processes them.

Loads a list of requests from fileName, then attempts to find a path for them using map.getPath() and either tells the client that HPAir doesn't fly there or gives them their flight plan

**Precondition**

fileName contains the name of a valid file
map contains a list of cities filled with flights

**Parameters**

| | | |
|---|---|---|
| in | *map* | The FlightMap to get cities and flights from |
| in | *fileName* | The name of the file to load cities from |

#### 4.3.2.2 loadCities()

```
void loadCities (
            FlightMap & map,
            const std::string & fileName )
```

Loads a list of cities from fileName and adds them to map.

**Precondition**

fileName contains the name of a valid file

**Postcondition**

map will be filled with any new cities found in the file

**Parameters**

| | | |
|---|---|---|
| in | *map* | The FlightMap to add cities to |
| in | *fileName* | The name of the file to load cities from |

#### 4.3.2.3 loadFlights()

```
void loadFlights (
            FlightMap & map,
            const std::string & fileName )
```

Loads a list of flights from fileName and adds them to map.

**Precondition**

fileName contains the name of a valid file

**Postcondition**

Adds any parsed flights from fileName into the cities contained in map

**Parameters**

| | | |
|---|---|---|
| in | *map* | The FlightMap to add flights to |
| in | *fileName* | The name of the file to load cities from |

#### 4.3.2.4 printFlightPlan()

```
void printFlightPlan (
            std::stack< Flight *> flightPlan )
```

Takes a flight plan and prints it on screen.

**Precondition**

flightPlan must not be empty

**Parameters**

| | | |
|---|---|---|
| in | *flightPlan* | A stack of flights in order of first to last from top to bottom. |
| in | *fileName* | The name of the file to load cities from |

## 4.4 PA03/PA03.h File Reference

Program driver header file.

```
#include <iostream>
#include <fstream>
#include <string>
#include <regex>
#include "FlightMapV1.h"
```

### Macros

- #define **CITY_FILE** "cityFile.txt"
- #define **FLIGHT_FILE** "flightFile.txt"
- #define **REQUEST_FILE** "requestFile.txt"

### Functions

- void loadCities (FlightMap &, const std::string &)

  *Loads a list of cities from fileName and adds them to map.*
- void loadFlights (FlightMap &, const std::string &)

  *Loads a list of flights from fileName and adds them to map.*

- void handleRequests (FlightMap &, const std::string &)

  *Loads a list of requests from fileName and then processes them.*
- void printFlightPlan (std::stack< Flight *>)

  *Takes a flight plan and prints it on screen.*
- const std::regex **regexFlight** ("^([[:alpha:] ]+),[\]([[:alpha:] ]+)[\]([[:digit:]]+)[\]([[:digit:]]+)$")
- const std::regex **regexRequest** ("^([[:alpha:] ]+),[\]([[:alpha:] ]+)$")

### 4.4.1 Detailed Description

Program driver header file.

Function prototypes, precompiler macros, and includes for program driver

#### Version

1.00 Alexander Novotny First version

### 4.4.2 Function Documentation

#### 4.4.2.1 handleRequests()

```
void handleRequests (
            FlightMap & map,
            const std::string & fileName )
```

Loads a list of requests from fileName and then processes them.

Loads a list of requests from fileName, then attempts to find a path for them using map.getPath() and either tells the client that HPAir doesn't fly there or gives them their flight plan

#### Precondition

fileName contains the name of a valid file
map contains a list of cities filled with flights

#### Parameters

| | | |
|---|---|---|
| in | *map* | The FlightMap to get cities and flights from |
| in | *fileName* | The name of the file to load cities from |

#### 4.4.2.2 loadCities()

```
void loadCities (
            FlightMap & map,
            const std::string & fileName )
```

Loads a list of cities from fileName and adds them to map.

**Precondition**

fileName contains the name of a valid file

**Postcondition**

map will be filled with any new cities found in the file

**Parameters**

| | | |
|---|---|---|
| in | *map* | The FlightMap to add cities to |
| in | *fileName* | The name of the file to load cities from |

#### 4.4.2.3 loadFlights()

```
void loadFlights (
            FlightMap & map,
            const std::string & fileName )
```

Loads a list of flights from fileName and adds them to map.

**Precondition**

fileName contains the name of a valid file

**Postcondition**

Adds any parsed flights from fileName into the cities contained in map

**Parameters**

| | | |
|---|---|---|
| in | *map* | The FlightMap to add flights to |
| in | *fileName* | The name of the file to load cities from |

#### 4.4.2.4 printFlightPlan()

```
void printFlightPlan (
            std::stack< Flight *> flightPlan )
```

Takes a flight plan and prints it on screen.

**Precondition**

flightPlan must not be empty

**Parameters**

| | | |
|---|---|---|
| in | *flightPlan* | A stack of flights in order of first to last from top to bottom. |
| in | *fileName* | The name of the file to load cities from |

# Index