

My Project

Generated by Doxygen 1.9.1

1 Namespace Index	1
1.1 Namespace List	1
2 Hierarchical Index	3
2.1 Class Hierarchy	3
3 Class Index	5
3.1 Class List	5
4 File Index	7
4.1 File List	7
5 Namespace Documentation	9
5.1 generate_uml Namespace Reference	9
5.1.1 Function Documentation	9
5.1.1.1 generate_uml_diagram()	9
5.1.2 Variable Documentation	9
5.1.2.1 input_file	9
5.1.2.2 output_file	9
6 Class Documentation	11
6.1 Airport Class Reference	11
6.1.1 Member Function Documentation	11
6.1.1.1 get_airport()	11
6.1.1.2 get_city()	12
6.1.1.3 get_fuel_cost_per_liter()	12
6.1.1.4 get_lane_lengths()	12
6.1.1.5 get_location()	12
6.1.1.6 get_scheduled_flights()	12
6.1.2 Friends And Related Function Documentation	12
6.1.2.1 AirSpaceManager	12
6.1.2.2 operator<<	12
6.2 AirSpaceManager Class Reference	13
6.2.1 Detailed Description	13
6.2.2 Constructor & Destructor Documentation	13
6.2.2.1 AirSpaceManager()	13
6.2.2.2 ~AirSpaceManager()	13
6.2.3 Member Function Documentation	14
6.2.3.1 input_airport()	14
6.2.3.2 input_flight()	14
6.2.3.3 input_plane()	14
6.2.3.4 run()	14
6.3 DateTime Class Reference	14
6.3.1 Detailed Description	15

6.3.2 Constructor & Destructor Documentation	15
6.3.2.1 DateTime() [1/2]	15
6.3.2.2 DateTime() [2/2]	15
6.3.3 Friends And Related Function Documentation	16
6.3.3.1 operator<<	16
6.4 Flight Class Reference	16
6.4.1 Detailed Description	17
6.4.2 Member Function Documentation	17
6.4.2.1 assign_plane()	17
6.4.2.2 exhaust_rate() [1/2]	17
6.4.2.3 exhaust_rate() [2/2]	18
6.4.2.4 flight_duration()	18
6.4.2.5 set_from()	18
6.4.2.6 set_to()	19
6.4.3 Friends And Related Function Documentation	19
6.4.3.1 AirSpaceManager	19
6.4.3.2 operator<<	19
6.5 IEnumerate Class Reference	20
6.5.1 Detailed Description	20
6.5.2 Constructor & Destructor Documentation	20
6.5.2.1 IEnumerate()	20
6.5.3 Member Function Documentation	20
6.5.3.1 get_id()	21
6.5.4 Friends And Related Function Documentation	21
6.5.4.1 AirSpaceManager	21
6.6 Plane Class Reference	21
6.6.1 Detailed Description	22
6.6.2 Member Function Documentation	22
6.6.2.1 can_land_on()	22
6.6.2.2 get_average_speed()	22
6.6.2.3 max_flight_distance()	23
6.6.3 Friends And Related Function Documentation	23
6.6.3.1 AirSpaceManager	23
6.6.3.2 operator<<	23
6.7 Time Class Reference	23
6.7.1 Detailed Description	24
6.7.2 Constructor & Destructor Documentation	24
6.7.2.1 Time() [1/2]	24
6.7.2.2 Time() [2/2]	24
6.7.3 Member Function Documentation	25
6.7.3.1 from()	25
6.7.4 Friends And Related Function Documentation	25

6.7.4.1 operator<<	25
7 File Documentation	27
7.1 airport.cpp File Reference	27
7.2 airport.hpp File Reference	27
7.2.1 Detailed Description	27
7.3 airspace_manager.cpp File Reference	27
7.4 airspace_manager.hpp File Reference	28
7.4.1 Detailed Description	28
7.5 date_time.cpp File Reference	28
7.5.1 Function Documentation	28
7.5.1.1 operator<<() [1/2]	28
7.5.1.2 operator<<() [2/2]	29
7.6 date_time.hpp File Reference	29
7.6.1 Detailed Description	29
7.7 enumerable.cpp File Reference	30
7.8 enumerable.hpp File Reference	30
7.8.1 Detailed Description	30
7.9 flight.cpp File Reference	30
7.9.1 Function Documentation	30
7.9.1.1 operator<<()	30
7.10 flight.hpp File Reference	31
7.10.1 Detailed Description	31
7.11 generate_uml.py File Reference	31
7.12 main.cpp File Reference	32
7.12.1 Function Documentation	32
7.12.1.1 main()	32
7.13 plane.cpp File Reference	32
7.13.1 Function Documentation	32
7.13.1.1 operator<<()	32
7.14 plane.hpp File Reference	33
7.14.1 Detailed Description	33
7.15 project_fwd_def.hpp File Reference	33
7.15.1 Detailed Description	33
Index	35

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

generate_uml	9
--	---

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

AirSpaceManager	13
DateTime	14
IEnumerable	20
Airport	11
Flight	16
Plane	21
Time	23

Chapter 3

Class Index

3.1 Class List

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

Airport	11
AirSpaceManager A class to manage airports, planes, and flights for optimizing airline resources	13
DateTime Represents a specific point in time with detailed components like year, month, day, hour, minute, and second	14
Flight Represents a flight, including its schedule, route, and assigned plane	16
IEnumerable A base class that provides unique IDs for objects	20
Plane Represents an aircraft with details for efficient airline management	21
Time Represents a time duration or specific time of day without date information	23

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

airport.cpp	27
airport.hpp	
Contains the declaration of the airport class and its methods	27
airspace_manager.cpp	27
airspace_manager.hpp	
Contains the declaration of the AirSpaceManager class and its methods for managing airports, planes, and flights	28
date_time.cpp	28
date_time.hpp	
Contains the declaration of the DateTime and Time classes and their methods for handling date and time information	29
enumerable.cpp	30
enumerable.hpp	
Contains the declaration of the IEnumerable class, which provides a unique identifier for derived objects	30
flight.cpp	30
flight.hpp	
Contains the declaration of the Flight class, representing a flight with associated data and operations	31
generate_uml.py	31
main.cpp	32
plane.cpp	32
plane.hpp	
Contains the declaration of the Plane class and its methods	33
project_fwd_def.hpp	
Contains the declaration of the project_fwd_def class and its methods	33

Chapter 5

Namespace Documentation

5.1 generate_uuml Namespace Reference

Functions

- def `generate_uuml_diagram` (`input_file`, `output_file`)

Variables

- string `input_file` = "uml_diagram.puml"
- string `output_file` = "uml_diagram.jpg"

5.1.1 Function Documentation

5.1.1.1 generate_uuml_diagram()

```
def generate_uuml.generate_uuml_diagram (  
    input_file,  
    output_file )
```

5.1.2 Variable Documentation

5.1.2.1 input_file

```
string generate_uuml.input_file = "uml_diagram.puml"
```

5.1.2.2 output_file

```
string generate_uuml.output_file = "uml_diagram.jpg"
```


Chapter 6

Class Documentation

6.1 Airport Class Reference

```
#include <airport.hpp>
```

Inheritance diagram for Airport:

Collaboration diagram for Airport:

Public Member Functions

- const std::string & [get_city](#) () const
- const std::string & [get_airport](#) () const
- const std::string & [get_location](#) () const
- const std::vector< double > & [get_lane_lengths](#) () const
- double [get_fuel_cost_per_liter](#) () const
- const std::vector< [Flight](#) * > & [get_scheduled_flights](#) () const

Friends

- class [AirSpaceManager](#)
- std::ostream & [operator<<](#) (std::ostream &out, const [Airport](#) &airport)

6.1.1 Member Function Documentation

6.1.1.1 [get_airport\(\)](#)

```
const std::string & Airport::get_airport ( ) const
```

6.1.1.2 get_city()

```
const std::string & Airport::get_city ( ) const
```

6.1.1.3 get_fuel_cost_per_liter()

```
double Airport::get_fuel_cost_per_liter ( ) const
```

6.1.1.4 get_lane_lengths()

```
const std::vector< double > & Airport::get_lane_lengths ( ) const
```

6.1.1.5 get_location()

```
const std::string & Airport::get_location ( ) const
```

6.1.1.6 get_scheduled_flights()

```
const std::vector< Flight * > & Airport::get_scheduled_flights ( ) const
```

6.1.2 Friends And Related Function Documentation

6.1.2.1 AirSpaceManager

```
friend class AirSpaceManager [friend]
```

6.1.2.2 operator<<

```
std::ostream& operator<< (
    std::ostream & out,
    const Airport & airport ) [friend]
```

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

- [airport.hpp](#)
- [airport.cpp](#)

6.2 AirSpaceManager Class Reference

A class to manage airports, planes, and flights for optimizing airline resources.

```
#include <airspace_manager.hpp>
```

Collaboration diagram for AirSpaceManager:

Public Member Functions

- [AirSpaceManager](#) ()
Default constructor for [AirSpaceManager](#).
- [~AirSpaceManager](#) ()
Destructor for [AirSpaceManager](#). Frees allocated memory.
- void [input_airport](#) ()
Prompts the user to input details for a new airport.
- void [input_plane](#) ()
Prompts the user to input details for a new plane.
- void [input_flight](#) ()
Prompts the user to input details for a new flight.
- void [run](#) ()
Main entry point to run the [AirSpaceManager](#) application.

6.2.1 Detailed Description

A class to manage airports, planes, and flights for optimizing airline resources.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 AirSpaceManager()

```
AirSpaceManager::AirSpaceManager ( )
```

Default constructor for [AirSpaceManager](#).

6.2.2.2 ~AirSpaceManager()

```
AirSpaceManager::~~AirSpaceManager ( )
```

Destructor for [AirSpaceManager](#). Frees allocated memory.

6.2.3 Member Function Documentation

6.2.3.1 input_airport()

```
void AirSpaceManager::input_airport ( )
```

Prompts the user to input details for a new airport.

6.2.3.2 input_flight()

```
void AirSpaceManager::input_flight ( )
```

Prompts the user to input details for a new flight.

6.2.3.3 input_plane()

```
void AirSpaceManager::input_plane ( )
```

Prompts the user to input details for a new plane.

6.2.3.4 run()

```
void AirSpaceManager::run ( )
```

Main entry point to run the [AirSpaceManager](#) application.

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

- [airspace_manager.hpp](#)
- [airspace_manager.cpp](#)

6.3 DateTime Class Reference

Represents a specific point in time with detailed components like year, month, day, hour, minute, and second.

```
#include <date_time.hpp>
```

Collaboration diagram for DateTime:

Public Member Functions

- [DateTime](#) ()
Default constructor initializing to an undefined date and time.
- [DateTime](#) (unsigned short year, unsigned short month, unsigned short day, unsigned short hour, unsigned short minute, unsigned short second)
Constructs a [DateTime](#) object with the specified date and time.

Friends

- std::ostream & [operator<<](#) (std::ostream &out, const [DateTime](#) &time)
Overloaded output stream operator for [DateTime](#).

6.3.1 Detailed Description

Represents a specific point in time with detailed components like year, month, day, hour, minute, and second.

6.3.2 Constructor & Destructor Documentation

6.3.2.1 DateTime() [1/2]

```
DateTime::DateTime ( )
```

Default constructor initializing to an undefined date and time.

6.3.2.2 DateTime() [2/2]

```
DateTime::DateTime (
    unsigned short year,
    unsigned short month,
    unsigned short day,
    unsigned short hour,
    unsigned short minute,
    unsigned short second )
```

Constructs a [DateTime](#) object with the specified date and time.

Parameters

<i>year</i>	The year of the date.
<i>month</i>	The month of the date (1-12).
<i>day</i>	The day of the date (1-31).
<i>hour</i>	The hour of the time (0-23).
<i>minute</i>	The minute of the time (0-59).
<i>second</i>	The second of the time (0-59).

6.3.3 Friends And Related Function Documentation

6.3.3.1 operator<<

```
std::ostream& operator<< (
    std::ostream & out,
    const DateTime & time ) [friend]
```

Overloaded output stream operator for [DateTime](#).

Parameters

<i>out</i>	The output stream.
<i>time</i>	The DateTime object to be output.

Returns

Reference to the output stream.

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

- [date_time.hpp](#)
- [date_time.cpp](#)

6.4 Flight Class Reference

Represents a flight, including its schedule, route, and assigned plane.

```
#include <flight.hpp>
```

Inheritance diagram for Flight:

Collaboration diagram for Flight:

Public Member Functions

- bool [assign_plane](#) (const [Plane](#) &candidate)
Assigns a plane to the flight if it meets the necessary criteria.
- bool [set_from](#) (const [Airport](#) &from)
Sets the origin airport for the flight.
- bool [set_to](#) (const [Airport](#) &to)
Sets the destination airport for the flight.
- double [exhaust_rate](#) () const
Calculates the workload for the pilot(s) during this flight.
- double [exhaust_rate](#) (const [Plane](#) * _plane) const
Calculates the workload for a given plane during this flight.
- double [flight_duration](#) () const
Calculates the flight duration based on the route length and plane speed.

Friends

- class [AirSpaceManager](#)
Declares [AirSpaceManager](#) as a friend class to access private members.
- `std::ostream & operator<< (std::ostream &out, const Flight &flight)`
Overloaded output stream operator for [Flight](#).

6.4.1 Detailed Description

Represents a flight, including its schedule, route, and assigned plane.

The [Flight](#) class encapsulates details about a specific flight, such as the takeoff time, route length, origin and destination airports, and the plane assigned to the flight. It also provides methods to calculate workload and flight duration.

6.4.2 Member Function Documentation

6.4.2.1 `assign_plane()`

```
bool Flight::assign_plane (  
    const Plane & candidate )
```

Assigns a plane to the flight if it meets the necessary criteria.

Parameters

<i>candidate</i>	The plane to be considered for assignment.
------------------	--

Returns

True if the plane was successfully assigned, false otherwise.

6.4.2.2 `exhaust_rate()` [1/2]

```
double Flight::exhaust_rate ( ) const
```

Calculates the workload for the pilot(s) during this flight.

Returns

The workload as a percentage.

6.4.2.3 exhaust_rate() [2/2]

```
double Flight::exhaust_rate (
    const Plane * _plane ) const
```

Calculates the workload for a given plane during this flight.

Parameters

<code>_plane</code>	The plane to evaluate.
---------------------	------------------------

Returns

The workload as a percentage for the specified plane.

6.4.2.4 flight_duration()

```
double Flight::flight_duration ( ) const
```

Calculates the flight duration based on the route length and plane speed.

Returns

The flight duration in hours.

6.4.2.5 set_from()

```
bool Flight::set_from (
    const Airport & from )
```

Sets the origin airport for the flight.

Parameters

<code>from</code>	The airport object representing the flight's origin.
-------------------	--

Returns

True if the origin was set successfully, false otherwise.

6.4.2.6 set_to()

```
bool Flight::set_to (
    const Airport & to )
```

Sets the destination airport for the flight.

Parameters

<i>to</i>	The airport object representing the flight's destination.
-----------	---

Returns

True if the destination was set successfully, false otherwise.

6.4.3 Friends And Related Function Documentation

6.4.3.1 AirSpaceManager

```
friend class AirSpaceManager [friend]
```

Declares [AirSpaceManager](#) as a friend class to access private members.

6.4.3.2 operator<<

```
std::ostream& operator<< (
    std::ostream & out,
    const Flight & flight ) [friend]
```

Overloaded output stream operator for [Flight](#).

Parameters

<i>out</i>	The output stream.
<i>flight</i>	The Flight object to be output.

Returns

Reference to the output stream.

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

- [flight.hpp](#)
- [flight.cpp](#)

6.5 IEnumerable Class Reference

A base class that provides unique IDs for objects.

```
#include <enumerable.hpp>
```

Inheritance diagram for IEnumerable:

Collaboration diagram for IEnumerable:

Public Member Functions

- [IEnumerable](#) ()
Default constructor. Assigns a unique ID to the instance.
- virtual unsigned int [get_id](#) () const final
Retrieves the unique ID of the instance.

Friends

- class [AirSpaceManager](#)
Declares [AirSpaceManager](#) as a friend class, allowing it to access private members.

6.5.1 Detailed Description

A base class that provides unique IDs for objects.

This class is designed to assign a unique identifier to each instance of a derived class.

6.5.2 Constructor & Destructor Documentation

6.5.2.1 IEnumerable()

```
IEnumerable::IEnumerable ( ) [inline]
```

Default constructor. Assigns a unique ID to the instance.

6.5.3 Member Function Documentation

6.5.3.1 `get_id()`

```
virtual unsigned int IEnumerable::get_id ( ) const [inline], [final], [virtual]
```

Retrieves the unique ID of the instance.

Returns

The unique ID of the object.

6.5.4 Friends And Related Function Documentation

6.5.4.1 `AirSpaceManager`

```
friend class AirSpaceManager [friend]
```

Declares `AirSpaceManager` as a friend class, allowing it to access private members.

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

- [enumerable.hpp](#)
- [enumerable.cpp](#)

6.6 Plane Class Reference

Represents an aircraft with details for efficient airline management.

```
#include <plane.hpp>
```

Inheritance diagram for Plane:

Collaboration diagram for Plane:

Public Member Functions

- `bool can_land_on (const Airport &airport) const`
Checks if the plane can land on a given airport.
- `double max_flight_distance () const`
Calculates the maximum flight distance for the plane.
- `double get_average_speed () const`
Retrieves the average speed of the plane.

Friends

- class [AirSpaceManager](#)
Declares [AirSpaceManager](#) as a friend class. Allows access to the private and protected members of [Plane](#).
- `std::ostream & operator<<` (`std::ostream &out`, `const Plane &flight`)
Overloaded output stream operator for [Plane](#). Outputs the details of the plane to the provided stream.

6.6.1 Detailed Description

Represents an aircraft with details for efficient airline management.

The [Plane](#) class encapsulates data about the manufacturer, model, seating capacity, minimum runway length, operational costs, fuel consumption, tank volume, and average speed.

6.6.2 Member Function Documentation

6.6.2.1 `can_land_on()`

```
bool Plane::can_land_on (
    const Airport & airport ) const
```

Checks if the plane can land on a given airport.

Parameters

<i>airport</i>	The airport to check compatibility with.
----------------	--

Returns

True if the plane can land, false otherwise.

6.6.2.2 `get_average_speed()`

```
double Plane::get_average_speed ( ) const
```

Retrieves the average speed of the plane.

Returns

The average speed in kilometers per hour.

6.6.2.3 max_flight_distance()

```
double Plane::max_flight_distance ( ) const
```

Calculates the maximum flight distance for the plane.

Returns

The maximum distance the plane can travel in kilometers.

6.6.3 Friends And Related Function Documentation

6.6.3.1 AirSpaceManager

```
friend class AirSpaceManager [friend]
```

Declares [AirSpaceManager](#) as a friend class. Allows access to the private and protected members of [Plane](#).

6.6.3.2 operator<<

```
std::ostream& operator<< (
    std::ostream & out,
    const Plane & flight ) [friend]
```

Overloaded output stream operator for [Plane](#). Outputs the details of the plane to the provided stream.

Parameters

<i>out</i>	The output stream.
<i>flight</i>	The plane object to be output.

Returns

Reference to the output stream.

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

- [plane.hpp](#)
- [plane.cpp](#)

6.7 Time Class Reference

Represents a time duration or specific time of day without date information.

```
#include <date_time.hpp>
```

Collaboration diagram for Time:

Public Member Functions

- [Time](#) ()
Default constructor initializing to an undefined time.
- [Time](#) (unsigned short hour, unsigned short minute, unsigned short second)
Constructs a [Time](#) object with the specified hours, minutes, and seconds.

Static Public Member Functions

- static [Time from](#) (double hours)
Converts a time duration in hours (as a double) to a [Time](#) object.

Friends

- std::ostream & [operator<<](#) (std::ostream &out, const [Time](#) &time)
Overloaded output stream operator for [Time](#).

6.7.1 Detailed Description

Represents a time duration or specific time of day without date information.

6.7.2 Constructor & Destructor Documentation

6.7.2.1 [Time](#)() [1/2]

```
Time::Time ( )
```

Default constructor initializing to an undefined time.

6.7.2.2 [Time](#)() [2/2]

```
Time::Time (
    unsigned short hour,
    unsigned short minute,
    unsigned short second )
```

Constructs a [Time](#) object with the specified hours, minutes, and seconds.

Parameters

<i>hour</i>	The hour of the time (0-23).
<i>minute</i>	The minute of the time (0-59).
<i>second</i>	The second of the time (0-59).

6.7.3 Member Function Documentation

6.7.3.1 from()

```
Time Time::from (
    double hours ) [static]
```

Converts a time duration in hours (as a double) to a [Time](#) object.

Parameters

<i>hours</i>	The time duration in hours.
--------------	-----------------------------

Returns

A [Time](#) object representing the duration.

6.7.4 Friends And Related Function Documentation

6.7.4.1 operator<<

```
std::ostream& operator<< (
    std::ostream & out,
    const Time & time ) [friend]
```

Overloaded output stream operator for [Time](#).

Parameters

<i>out</i>	The output stream.
<i>time</i>	The Time object to be output.

Returns

Reference to the output stream.

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

- [date_time.hpp](#)
- [date_time.cpp](#)

Chapter 7

File Documentation

7.1 airport.cpp File Reference

```
#include "airport.hpp"  
#include "date_time.hpp"  
#include "flight.hpp"  
#include "plane.hpp"  
#include <ostream>
```

Include dependency graph for airport.cpp:

7.2 airport.hpp File Reference

Contains the declaration of the airport class and its methods.

```
#include "project_fwd_def.hpp"  
#include "enumerable.hpp"  
#include <iostream>  
#include <string>  
#include <vector>
```

Include dependency graph for airport.hpp: This graph shows which files directly or indirectly include this file:

Classes

- class [Airport](#)

7.2.1 Detailed Description

Contains the declaration of the airport class and its methods.

7.3 airspace_manager.cpp File Reference

```
#include "airspace_manager.hpp"  
#include "plane.hpp"
```

Include dependency graph for airspace_manager.cpp:

7.4 airspace_manager.hpp File Reference

Contains the declaration of the [AirSpaceManager](#) class and its methods for managing airports, planes, and flights.

```
#include <iostream>
#include <string>
#include <sstream>
#include <vector>
#include "date_time.hpp"
#include "airport.hpp"
#include "plane.hpp"
#include "flight.hpp"
```

Include dependency graph for `airspace_manager.hpp`: This graph shows which files directly or indirectly include this file:

Classes

- class [AirSpaceManager](#)
A class to manage airports, planes, and flights for optimizing airline resources.

7.4.1 Detailed Description

Contains the declaration of the [AirSpaceManager](#) class and its methods for managing airports, planes, and flights.

7.5 date_time.cpp File Reference

```
#include "date_time.hpp"
#include <iomanip>
Include dependency graph for date_time.cpp:
```

Functions

- `std::ostream & operator<< (std::ostream &out, const DateTime &time)`
- `std::ostream & operator<< (std::ostream &out, const Time &time)`

7.5.1 Function Documentation

7.5.1.1 `operator<<()` [1/2]

```
std::ostream& operator<< (
    std::ostream & out,
    const DateTime & time )
```

Parameters

<i>out</i>	The output stream.
<i>time</i>	The DateTime object to be output.

Returns

Reference to the output stream.

7.5.1.2 operator<<() [2/2]

```
std::ostream& operator<< (
    std::ostream & out,
    const Time & time )
```

Parameters

<i>out</i>	The output stream.
<i>time</i>	The Time object to be output.

Returns

Reference to the output stream.

7.6 date_time.hpp File Reference

Contains the declaration of the [DateTime](#) and [Time](#) classes and their methods for handling date and time information.

```
#include "project_fwd_def.hpp"
#include <ostream>
```

Include dependency graph for date_time.hpp: This graph shows which files directly or indirectly include this file:

Classes

- class [DateTime](#)
Represents a specific point in time with detailed components like year, month, day, hour, minute, and second.
- class [Time](#)
Represents a time duration or specific time of day without date information.

7.6.1 Detailed Description

Contains the declaration of the [DateTime](#) and [Time](#) classes and their methods for handling date and time information.

7.7 enumerable.cpp File Reference

```
#include "enumerable.hpp"
```

Include dependency graph for enumerable.cpp:

7.8 enumerable.hpp File Reference

Contains the declaration of the [IEnumerable](#) class, which provides a unique identifier for derived objects.

```
#include "project_fwd_def.hpp"
```

Include dependency graph for enumerable.hpp: This graph shows which files directly or indirectly include this file:

Classes

- class [IEnumerable](#)

A base class that provides unique IDs for objects.

7.8.1 Detailed Description

Contains the declaration of the [IEnumerable](#) class, which provides a unique identifier for derived objects.

7.9 flight.cpp File Reference

```
#include "airport.hpp"
```

```
#include "date_time.hpp"
```

```
#include "flight.hpp"
```

```
#include "plane.hpp"
```

Include dependency graph for flight.cpp:

Functions

- `std::ostream & operator<< (std::ostream &out, const Flight &flight)`

7.9.1 Function Documentation

7.9.1.1 `operator<<()`

```
std::ostream& operator<< (  
    std::ostream & out,  
    const Flight & flight )
```

Parameters

<i>out</i>	The output stream.
<i>flight</i>	The Flight object to be output.

Returns

Reference to the output stream.

7.10 flight.hpp File Reference

Contains the declaration of the [Flight](#) class, representing a flight with associated data and operations.

```
#include "project_fwd_def.hpp"
#include "date_time.hpp"
#include "enumerable.hpp"
#include <iostream>
```

Include dependency graph for flight.hpp: This graph shows which files directly or indirectly include this file:

Classes

- class [Flight](#)

Represents a flight, including its schedule, route, and assigned plane.

7.10.1 Detailed Description

Contains the declaration of the [Flight](#) class, representing a flight with associated data and operations.

7.11 generate_uml.py File Reference

Namespaces

- [generate_uml](#)

Functions

- def [generate_uml.generate_uml_diagram](#) (input_file, output_file)

Variables

- string [generate_uml.input_file](#) = "uml_diagram.puml"
- string [generate_uml.output_file](#) = "uml_diagram.jpg"

7.12 main.cpp File Reference

```
#include "airspace_manager.hpp"
Include dependency graph for main.cpp:
```

Functions

- int [main](#) ()

7.12.1 Function Documentation

7.12.1.1 main()

```
int main ( )
```

7.13 plane.cpp File Reference

```
#include "airport.hpp"
#include "enumerable.hpp"
#include "plane.hpp"
Include dependency graph for plane.cpp:
```

Functions

- std::ostream & [operator<<](#) (std::ostream &out, const [Plane](#) &plane)

7.13.1 Function Documentation

7.13.1.1 operator<<()

```
std::ostream& operator<< (
    std::ostream & out,
    const Plane & plane )
```

Parameters

<i>out</i>	The output stream.
<i>flight</i>	The plane object to be output.

Returns

Reference to the output stream.

7.14 plane.hpp File Reference

Contains the declaration of the [Plane](#) class and its methods.

```
#include "project_fwd_def.hpp"
#include "enumerable.hpp"
#include <string>
```

Include dependency graph for plane.hpp: This graph shows which files directly or indirectly include this file:

Classes

- class [Plane](#)
Represents an aircraft with details for efficient airline management.

7.14.1 Detailed Description

Contains the declaration of the [Plane](#) class and its methods.

7.15 project_fwd_def.hpp File Reference

Contains the declaration of the project_fwd_def class and its methods.

This graph shows which files directly or indirectly include this file:

7.15.1 Detailed Description

Contains the declaration of the project_fwd_def class and its methods.

Index

- ~AirSpaceManager
 - AirSpaceManager, 13
- Airport, 11
 - AirSpaceManager, 12
 - get_airport, 11
 - get_city, 11
 - get_fuel_cost_per_liter, 12
 - get_lane_lengths, 12
 - get_location, 12
 - get_scheduled_flights, 12
 - operator<<, 12
- airport.cpp, 27
- airport.hpp, 27
- airspace_manager.cpp, 27
- airspace_manager.hpp, 28
- AirSpaceManager, 13
 - ~AirSpaceManager, 13
 - Airport, 12
 - AirSpaceManager, 13
 - Flight, 19
 - IEnumerable, 21
 - input_airport, 14
 - input_flight, 14
 - input_plane, 14
 - Plane, 23
 - run, 14
- assign_plane
 - Flight, 17
- can_land_on
 - Plane, 22
- date_time.cpp, 28
 - operator<<, 28, 29
- date_time.hpp, 29
- DateTime, 14
 - DateTime, 15
 - operator<<, 16
- enumerable.cpp, 30
- enumerable.hpp, 30
- exhaust_rate
 - Flight, 17
- Flight, 16
 - AirSpaceManager, 19
 - assign_plane, 17
 - exhaust_rate, 17
 - flight_duration, 18
 - operator<<, 19
 - set_from, 18
 - set_to, 18
- flight.cpp, 30
 - operator<<, 30
- flight.hpp, 31
- flight_duration
 - Flight, 18
- from
 - Time, 25
- generate_uuml, 9
 - generate_uuml_diagram, 9
 - input_file, 9
 - output_file, 9
- generate_uuml.py, 31
- generate_uuml_diagram
 - generate_uuml, 9
- get_airport
 - Airport, 11
- get_average_speed
 - Plane, 22
- get_city
 - Airport, 11
- get_fuel_cost_per_liter
 - Airport, 12
- get_id
 - IEnumerable, 20
- get_lane_lengths
 - Airport, 12
- get_location
 - Airport, 12
- get_scheduled_flights
 - Airport, 12
- IEnumerable, 20
 - AirSpaceManager, 21
 - get_id, 20
 - IEnumerable, 20
- input_airport
 - AirSpaceManager, 14
- input_file
 - generate_uuml, 9
- input_flight
 - AirSpaceManager, 14
- input_plane
 - AirSpaceManager, 14
- main
 - main.cpp, 32
- main.cpp, 32

- main, [32](#)
- max_flight_distance
 - Plane, [22](#)
- operator<<
 - Airport, [12](#)
 - date_time.cpp, [28](#), [29](#)
 - DateTime, [16](#)
 - Flight, [19](#)
 - flight.cpp, [30](#)
 - Plane, [23](#)
 - plane.cpp, [32](#)
 - Time, [25](#)
- output_file
 - generate_uuml, [9](#)
- Plane, [21](#)
 - AirSpaceManager, [23](#)
 - can_land_on, [22](#)
 - get_average_speed, [22](#)
 - max_flight_distance, [22](#)
 - operator<<, [23](#)
- plane.cpp, [32](#)
 - operator<<, [32](#)
- plane.hpp, [33](#)
- project_fwd_def.hpp, [33](#)
- run
 - AirSpaceManager, [14](#)
- set_from
 - Flight, [18](#)
- set_to
 - Flight, [18](#)
- Time, [23](#)
 - from, [25](#)
 - operator<<, [25](#)
 - Time, [24](#)