Namespace BrusOgPotetgull.AirportLiberary

Classes

Aircraft
: The Aircraft-class is a blueprint for how an aircraft would look like.

Airport
: This class is used to configure an airport and holds all its components.

ArrivingEventArgs
: Contains the arguments needed to handle the event for when an aircraft is landing.

ConnectionPoint
: This class represents a point of connection on the airport roadsystem. This can hold the connection one taxiway has to several others. Each taxiway has two connection points.

DepartingEventArgs
: Contains the arguments needed to handle the event for when an aircraft is departing.

Flight
: The Flight-class is defined with the aircraft that is used in the flight, together with some components on the airports its using. Examples of components: taxiways, gates and runways.

Flight.Arriving
: The Arriving-class represents an arriving flight. The class inherits from the Flight-class.

Flight.Departing
: The Departuring-class represents a departing flight. The class inherits from the Flight-class.

Gate
: The gate class is used to define how a gate is designed. It holds fields for the status of the gate and allowed aircraft types.

Runway
: The runway class is used to define how a runway is designed. It is also used to conduct operations on the runway.

Taxiway
: The taxiway class is used to define how a taxiway is designed. It is also used to conduct operations on the taxiway.

Terminal
: The terminal class is an area in the airport that can host a set of gates.

Class Aircraft

Namespace: BrusOgPotetgull.AirportLiberary

Assembly: brusOgPotetgull.airportLiberary.dll

The Aircraft-class is a blueprint for how an aircraft would look like.

```
public class Aircraft
```

Inheritance

object ← Aircraft

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object)

Constructors

Aircraft(string, AircraftType, int, int, int, int)

Creates an aircraft.

```
public Aircraft(string name, AircraftType aircraftType, int maxSpeedInAir, int
accelerationInAir, int maxSpeedOnGround, int accelerationOnGround)
```

Parameters

`name` string

What the aircraft is called.

`aircraftType` AircraftType

The model of the aircraft.

`maxSpeedInAir` int

Maximum in-air speed (Kp/h).

`accelerationInAir` int

The accleration in-air (Kp/h).

`maxSpeedOnGround` int

Maximum on-ground speed (Kp/h).

`accelerationOnGround` int

acceleration on ground (Kp/h).

Properties

AccelerationInAir

Gets aircraft acceleration in air

```
public int AccelerationInAir { get; }
```

Property Value

int

int of the acceleration in the air

AccelerationOnGround

Gets aircraft acceleration on ground

```
public int AccelerationOnGround { get; }
```

Property Value

int

int of acceleration on ground for aircraft

AircraftType

Gets aircraft type

```
public string AircraftType { get; }
```

Property Value

string

string of the aircraft type

AircraftTypeId

Gets aircraft type Id

```
public int AircraftTypeId { get; }
```

Property Value

int

int of the aircraft type id

MaxSpeedInAir

Gets airraft max speed in air

```
public int MaxSpeedInAir { get; }
```

Property Value

int

int of max air speed of plane

MaxSpeedOnGround

Gets aircraft max speed on ground

```
public int MaxSpeedOnGround { get; }
```

Property Value

int

int of the max speed on ground for the aircraft

Name

Gets aircraft name

```
public string Name { get; }
```

Property Value

string

string of the name of the aircraft

OutOfService

Gets if aircraft is in service with bool

```
public bool OutOfService { get; }
```

Property Value

bool

bool if out of service

TailNumber

Gets aircraft tail number

```
public int TailNumber { get; }
```

Property Value

int

int of the tailnumber of the aircraft

Methods

AddHistoryToAircraft(DateTime, string, string)

logging an event to the history of the aircraft.

```
public void AddHistoryToAircraft(DateTime time, string location, string message)
```

Parameters

`time` DateTime

When the event took place.

`location` string

The location of the plane.

`message` string

The action of the plane.

GetFullAircraftHistory()

Gets the history of the aircraft

```
public string GetFullAircraftHistory()
```

Returns

string

returns aircraft history in string

PrintAircraftHistoryForDay(int, int, int)

Reads trough the list of the aircrafts history and prints out the log for that day

```
public void PrintAircraftHistoryForDay(int year, int month, int day)
```

Parameters

`year` int

The year it checks

`month` int

The month it checks

`day` int

The day it checks

PrintAircraftInformation()

Prints the information about the Aircraft.

```
public virtual void PrintAircraftInformation()
```

PrintFullAircraftHistory()

Prints the full history of the plane.

```
public void PrintFullAircraftHistory()
```

ToString()

This override the ToString() method that exists in all objects in c#

```
public override string ToString()
```

Returns

string

A String with simple details about the aircraft.

Class Airport

Namespace: BrusOgPotetgull.AirportLiberary

Assembly: brusOgPotetgull.airportLiberary.dll

This class is used to configure an airport and holds all its components.

```
public class Airport
```

Inheritance

object ← Airport

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object)

Constructors

Airport(string, string, string)

Creates an airport without any components.

```
public Airport(string airportCode, string name, string location)
```

Parameters

`airportCode` string

The code for the airport. typicaly 3 letters. Eksample: RYG

`name` string

The name of the airport.

`location` string

Where the airport is located at.

Properties

AirportCode

gets airport code

```
public string AirportCode { get; }
```

Property Value

string

string with the code of the airport

AirportId

Gets airport id

```
public int AirportId { get; }
```

Property Value

int

int value with airport id

Location

gets airport location

```
public string Location { get; }
```

Property Value

string

string with location of the airport

Name

gets airport name

```
public string Name { get; }
```

Property Value

string

string with the name of the airport

Methods

AddArrivingFlight(Arriving)

Adds an arriving flight to this airport.

```
public void AddArrivingFlight(Flight.Arriving flight)
```

Parameters

`flight` Flight.Arriving

The arriving flight that is added to the list.

AddConnectionPoint(ConnectionPoint)

Adds a connection point to the taxiwaysystem

```
public void AddConnectionPoint(ConnectionPoint connection)
```

Parameters

`connection` ConnectionPoint

ConnectionPoint point to connect two or more taxiways

AddDailyArrivingFlight(int, Aircraft, DateTime, int, Airport, Gate, Taxiway, Runway)

Generates daily arriving flights. The first flight starts 24 hours after the value of the datetimeFlight object.

```
public void AddDailyArrivingFlight(int numberOfDays, Aircraft activeAircraft, DateTime
dateTimeFlight, int length, Airport arrivalAirport, Gate arrivalGate, Taxiway
arrivalTaxiway, Runway arrivalRunway)
```

Parameters

`numberOfDays` int

The number of days the flight will do its flights.

`activeAircraft` Aircraft

The aircraft that is used for this flight.

`dateTimeFlight` DateTime

Date of the flight.

`length` int

Length of the flight im KM.

`arrivalAirport` Airport

The airport that the aircraft is arriving at.

`arrivalGate` Gate

The gate that the aircraft is arriving at.

`arrivalTaxiway` Taxiway

The taxiway that the aircraft is arriving at.

`arrivalRunway` Runway

The runway that the aircraft is arriving at.

AddDailyDepartingFlight(int, Aircraft, DateTime, int, Airport, Gate, Taxiway, Runway)

Generates daily departuring flights. The first flight starts 24 hours after the value of the datetimeFlight object.

```
public void AddDailyDepartingFlight(int numberOfDays, Aircraft activeAircraft, DateTime
dateTimeFlight, int length, Airport departureAirport, Gate departureGate, Taxiway
departureTaxiway, Runway departureRunway)
```

Parameters

`numberOfDays` int

The number of days the flight will do its flights.

`activeAircraft` Aircraft

The aircraft that is used for this flight.

`dateTimeFlight` DateTime

Date of the flight.

`length` int

Length of the flight im KM.

`departureAirport` Airport

The airport that the aircraft departure from.

`departureGate` Gate

The gate that the aircraft departure from.

`departureTaxiway` Taxiway

The taxiway that the aircraft is using to departure from.

`departureRunway` Runway

The runway that the aircraft is departuring from.

AddDepartingFlight(Departing)

Adds an departuring flight to this airport.

```
public void AddDepartingFlight(Flight.Departing flight)
```

Parameters

`flight` Flight.Departing

The departuring flight that is added to the list.

AddGateToList(Gate)

Adds a gate to the airport.

```
public void AddGateToList(Gate gate)
```

Parameters

`gate` Gate

The gate that is added to the list of gates at this airport.

AddRunwayToList(Runway)

Adds a runway to the airport.

```
public void AddRunwayToList(Runway runway)
```

Parameters

`runway` Runway

The runway that is being added to the list of runways at this airport.

AddTaxiwayConnection(Taxiway, ConnectionPoint, ConnectionPoint)

creates the connection a taxiway has to connection points.

```
public void AddTaxiwayConnection(Taxiway taxiway, ConnectionPoint to, ConnectionPoint from)
```

Parameters

`taxiway` Taxiway

The taxiway you want to create a connection for.

`to` ConnectionPoint

Connection point B (to)

`from` ConnectionPoint

Connection point A (from)

AddTaxiwayToList(Taxiway)

Adds a taxiway to the airport.

```
public void AddTaxiwayToList(Taxiway taxiway)
```

Parameters

`taxiway` Taxiway

The taxiway that is added to the list of taxiways for this airport.

AddTerminalToList(Terminal)

Adds a terminal to the airport.

```
public void AddTerminalToList(Terminal terminal)
```

Parameters

`terminal` Terminal

The terminal that is added to the list of terminals for this airport.

AddWeeklyArrivingFlight(int, Aircraft, DateTime, int, Airport, Gate, Taxiway, Runway)

Generates weekly arriving flights. The first flight starts 1 week after the value of the datetimeFlight object.

```
public void AddWeeklyArrivingFlight(int numberOfWeeks, Aircraft activeAircraft, DateTime
dateTimeFlight, int length, Airport arrivalAirport, Gate arrivalGate, Taxiway
arrivalTaxiway, Runway arrivalRunway)
```

Parameters

`numberOfWeeks` int

The number of weeks the flight will do its flights.

`activeAircraft` Aircraft

The aircraft that is used for this flight.

`dateTimeFlight` DateTime

Date of the flight.

`length` int

Length of the flight im KM.

`arrivalAirport` Airport

The airport that the aircraft is arriving at.

`arrivalGate` Gate

The gate that the aircraft is arriving at.

`arrivalTaxiway` Taxiway

The taxiway that the aircraft is arriving at.

`arrivalRunway` Runway

The runway that the aircraft is arriving at.

AddWeeklyDepartingFlight(int, Aircraft, DateTime, int, Airport, Gate, Taxiway, Runway)

Generates weekly departuring flights. The first flight starts 1 week after the value of the datetimeFlight object.

```
public void AddWeeklyDepartingFlight(int numberOfWeeks, Aircraft activeAircraft, DateTime
dateTimeFlight, int length, Airport departureAirport, Gate departureGate, Taxiway
departureTaxiway, Runway departureRunway)
```

Parameters

`numberOfWeeks` int

The number of weeks the flight will do its flights.

`activeAircraft` Aircraft

The aircraft that is used for this flight.

`dateTimeFlight` DateTime

Date of the flight.

`length` int

Length of the flight im KM.

`departureAirport` Airport

The airport that the aircraft departure from.

`departureGate` Gate

The gate that the aircraft departure from.

`departureTaxiway` Taxiway

The taxiway that the aircraft is using to departure from.

`departureRunway` Runway

The runway that the aircraft is departuring from.

FindPath(Taxiway, Taxiway, List<Taxiway>)

Finds a path through the taxiway system from one taxiway to another.

```
public List<Taxiway> FindPath(Taxiway start, Taxiway end, List<Taxiway> calculatedRoute)
```

Parameters

`start` Taxiway

start taxiway of the path.

`end` Taxiway

end taxiway of the path.

`calculatedRoute` List<Taxiway>

An empty list of taxiways to be returned as a path

Returns

List<Taxiway>

Returns the path as a list of taxiway objects

GenerateArrivingFlightTaxiwayPath(Arriving)

Generates a path from one taxiway to another for an arriving flight.

```
public List<Taxiway> GenerateArrivingFlightTaxiwayPath(Flight.Arriving flight)
```

Parameters

`flight` Flight.Arriving

The Flight you want to generate a path for.

Returns

List<Taxiway>

Returns the path as a list of taxiway objects

Remarks

Only generates a path if the arrival gate on the flight is connected to one of the taxiways in the airport.

GenerateDeparturingFlightTaxiwayPath(Departing)

Generates a path from one taxiway to another for an departing flight.

```
public List<Taxiway> GenerateDeparturingFlightTaxiwayPath(Flight.Departing flight)
```

Parameters

`flight` Flight.Departing

The Flight you want to generate a path for.

Returns

List<Taxiway>

Returns the path as a list of taxiway objects

Remarks

Only generates a path if the depature gate on the flight is connected to one of the taxiways in the airport.

GetAnotherAvailabelGateAtTheSameTerminal(string)

Finds an availabel gate at the same terminal of the initially desired gate.

```
public Gate GetAnotherAvailabelGateAtTheSameTerminal(string nameOfDesiredGate)
```

Parameters

`nameOfDesiredGate` string

Gate you initially wanted to use.

Returns

Gate

A gate object

Remarks

Will return null if there is no availabel gate at the terminal.

GetArrivingFlights()

Gets all arriving flights for this airport.

```
public List<Flight> GetArrivingFlights()
```

Returns

List<Flight>

The list containing all arriving flights for this airport.

GetDepartingFlights()

Gets all departuring flights for this airport.

```
public List<Flight> GetDepartingFlights()
```

Returns

List<Flight>

A list of departuring flights.

GetGateBasedOnGateName(string)

Returns a gate object based on the gatename provided.

```
public Gate GetGateBasedOnGateName(string gateName)
```

Parameters

`gateName` string

Name of the gate you want to return.

Returns

Gate

gate object

Exceptions

InvalidOperationException

GetGatesById(int)

Gets a single gate based on id.

```
public Gate GetGatesById(int gateId)
```

Parameters

`gateId` int

The id of the gate that is desired.

Returns

Gate

The desired gate

Exceptions

InvalidOperationException

If airport has no gates or could not find any gates that matches the gates that exists in this airport.

GetListGates()

Returns a list of all the gates at this airport.

```
public List<Gate> GetListGates()
```

Returns

List<Gate>

A list of gates at this airport.

GetListTaxiways()

Returns a list of all the taxiways at this airport.

```
public List<Taxiway> GetListTaxiways()
```

Returns

List<Taxiway>

a list that contains all the taxiways at this airport.

GetListTerminals()

Returns a list of all the terminals at this airport.

```
public List<Terminal> GetListTerminals()
```

Returns

List<Terminal>

A list of Terminals at this airport.

GetRunwayList()

Returns a list of all the runways at this airport.

```
public List<Runway> GetRunwayList()
```

Returns

List<Runway>

A list of runways at this airport

GetTaxiwaySystem()

Gets the taxiway system at this airport. This is a list of connection points between taxiways.

```
public List<ConnectionPoint> GetTaxiwaySystem()
```

Returns

List<ConnectionPoint>

A List of connection points.

GetTerminalById(int)

Gets a single terminal based on id.

```
public Terminal GetTerminalById(int terminalId)
```

Parameters

`terminalId` int

The id of the termial that is desired.

Returns

Terminal

The desired terminal

Exceptions

InvalidOperationException

If airport has no terminals or could not find any terminals that matches the terminals that exists in this airport.

MakeAllGatesAllowAllAircraftTypes()

Makes all gates in this airport allow all aircraft types.

```
public void MakeAllGatesAllowAllAircraftTypes()
```

PrintAirportInformation()

Prints out the information about the airport.

```
public void PrintAirportInformation()
```

PrintListOfDepartingFlights()

Prints out information about every flight in the list of departuring flights for this airport.

```
public void PrintListOfDepartingFlights()
```

PrintTaxiwayRoute(List<Taxiway>)

Prints out the name of all the taxiways in the route, and the total number of taxiways.

```
public void PrintTaxiwayRoute(List<Taxiway> route)
```

Parameters

`route` List<Taxiway>

The route you want to print out to the console

PrintTaxiwaySystem()

Prints out the information about the taxiwaysystem (All the connected components).

```
public void PrintTaxiwaySystem()
```

RemoveArrivingFlight(Arriving)

Removes an arriving flight from this airport.

```
public void RemoveArrivingFlight(Flight.Arriving flight)
```

Parameters

`flight` Flight.Arriving

The arriving flight that is removed from the list.

RemoveDepartingFlight(Departing)

Removes a departuring flight from this airport.

```
public void RemoveDepartingFlight(Flight.Departing flight)
```

Parameters

`flight` Flight.Departing

The departuring flight that is removed from the list.

RemoveGateFromList(Gate)

Removes a gate from the airport.

```
public void RemoveGateFromList(Gate gate)
```

Parameters

`gate` Gate

The gate that is removed from the list of gates at this airport.

RemoveRunwayFromList(Runway)

Removes a runway from the airport.

```
public void RemoveRunwayFromList(Runway runway)
```

Parameters

`runway` Runway

The taxiway that is removed from the list of taxiways at this airport.

RemoveTaxiwayFromList(Taxiway)

Removes a taxiway from the airport.

```
public void RemoveTaxiwayFromList(Taxiway taxiway)
```

Parameters

`taxiway` Taxiway

The taxiway that is removed from the list of taxiways at this airport.

RemoveTerminalFromList(Terminal)

Removes a terminal from the airport.

```
public void RemoveTerminalFromList(Terminal terminal)
```

Parameters

`terminal` Terminal

The terminal that is removed from the list of terminals for this airport.

ToString()

This override the ToString() method that exists in all objects in c#

```
public override string ToString()
```

Returns

string

A String with simple details about the Airport.

Class ArrivingEventArgs

Namespace: BrusOgPotetgull.AirportLiberary

Assembly: brusOgPotetgull.airportLiberary.dll

Contains the arguments needed to handle the event for when an aircraft is landing.

```
public class ArrivingEventArgs : EventArgs
```

Inheritance

object ← EventArgs ← ArrivingEventArgs

Inherited Members

EventArgs.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

ArrivingEventArgs(Arriving, DateTime, string)

Sets the arguments for an arriving aircraft so that an event can be handled.

```
public ArrivingEventArgs(Flight.Arriving flight, DateTime time, string message)
```

Parameters

`flight` Flight.Arriving

The flight that is being handled by the event

`time` DateTime

The time of the event

`message` string

A message of what occured at the time of the event

Properties

Flight

Gets the arriving flight details.

```
public Flight.Arriving Flight { get; }
```

Property Value

Flight.Arriving

Arriving flight object of flight arriving at runnway

Message

Gets the message related to the event.

```
public string Message { get; }
```

Property Value

string

string of message related to event

Time

Gets the time associated with the event.

```
public DateTime Time { get; }
```

Property Value

DateTime

DateTime object of time aircraft arives at runway

Class ConnectionPoint

Namespace: BrusOgPotetgull.AirportLiberary

Assembly: brusOgPotetgull.airportLiberary.dll

This class represents a point of connection on the airport roadsystem. This can hold the connection one taxiway has to several others. Each taxiway has two connection points.

```
public class ConnectionPoint
```

Inheritance

object ← ConnectionPoint

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object)

Constructors

ConnectionPoint(string, Airport)

Creates a connection point in the taxiway system.

```
public ConnectionPoint(string name, Airport airport)
```

Parameters

`name` string

Name of the connection point

`airport` Airport

The airport that the ConnectionPoint will be located at.

Properties

Name

Gets the name of the connection poin

```
public string? Name { get; set; }
```

Property Value

string

string of name of connection point

taxiways

Gets or sets the list of taxiways connected to the connectionPoint.

```
public List<Taxiway> taxiways { get; set; }
```

Property Value

List <Taxiway>

Methods

ToString()

This override the ToString() method that exists in all objects in c#

```
public override string ToString()
```

Returns

string

A String with simple details about the ConnectionPoint.

Class DepartingEventArgs

Namespace: BrusOgPotetgull.AirportLiberary

Assembly: brusOgPotetgull.airportLiberary.dll

Contains the arguments needed to handle the event for when an aircraft is departing.

```
public class DepartingEventArgs : EventArgs
```

Inheritance

object ← EventArgs ← DepartingEventArgs

Inherited Members

EventArgs.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

DepartingEventArgs(Departing, DateTime, string)

Sets the arguments for a departing aircraft so that an event can be handled.

```
public DepartingEventArgs(Flight.Departing flight, DateTime time, string message)
```

Parameters

`flight` Flight.Departing

The flight that is being handled by the event

`time` DateTime

The time of the event

`message` string

A message of what occured at the time of the event

Properties

Flight

Gets the departing flight details.

```
public Flight.Departing Flight { get; }
```

Property Value

Flight.Departing

departing flight object of flight departing runway

Message

Gets the message related to the event.

```
public string Message { get; }
```

Property Value

string

string of message related to the event

Time

Gets the time associated with the event.

```
public DateTime Time { get; }
```

Property Value

DateTime

datetime object of time aircraft leaves runway

Class Flight

Namespace: BrusOgPotetgull.AirportLiberary

Assembly: brusOgPotetgull.airportLiberary.dll

The Flight-class is defined with the aircraft that is used in the flight, together with some components on the airports its using. Examples of components: taxiways, gates and runways.

```
public abstract class Flight
```

Inheritance

object ← Flight

Derived

Flight.Arriving, Flight.Departing

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object)

Constructors

Flight(Aircraft, DateTime, bool, int)

Creates a Flight-object. This must either be Arriving flight or departuring flight.

```
protected Flight(Aircraft activeAircraft, DateTime dateTimeFlight, bool isArrivingFlight,
int length)
```

Parameters

`activeAircraft` Aircraft

The aircraft that is used for this flight.

`dateTimeFlight` DateTime

Date of the flight.

`isArrivingFlight` bool

If the flight is an arriving flight, this value must be set to true.

`length` int

The lenght of the flight in KM.

Fields

taxiwayPath

Calculated route a flight takes on the taxiwaysystem to get from a gate to a runway, or wise-versa

```
public List<Taxiway> taxiwayPath
```

Field Value

List<Taxiway>

list of taxiways that makes up the path / route

Properties

ActiveAircraft

Gets active aircraft

```
public Aircraft ActiveAircraft { get; }
```

Property Value

Aircraft

Aircraft object that is set to fly

Clock

Gets or sets the time on clock

```
public DateTime Clock { get; set; }
```

Property Value

DateTime

DateTime value of the clock

DateTimeFlight

Gets the time of the flight

```
public DateTime DateTimeFlight { get; }
```

Property Value

DateTime

DateTime object that desides when the flight is.

FlightId

Gets flight id

```
public int FlightId { get; }
```

Property Value

int

Flight id that is assosiated with the flight

IsArrivingFlight

Gets bool to se if flight is ariving or taking off

```
public bool IsArrivingFlight { get; }
```

Property Value

bool

bool value that is true if flight is ariving and false if leaving

Length

Gets flight length

```
public int Length { get; }
```

Property Value

int

int value of the length of the flight

Methods

CalculateFlightMovement(int, int, int, int)

Calculates the movement of a flight object across a set length, and returning the time it took in seconds.

```
public double CalculateFlightMovement(int length, int initialSpeed, int speedChange,
int maxSpeed)
```

Parameters

`length` int

Traveldistance in meters.

`initialSpeed` int

The speed at which the aircraft starts traversing the lenght (Kp/h).

`speedChange` int

The change in speed per second (Kp/h).

`maxSpeed` int

Maximum speed of the aircraft (Kp/h).

Returns

double

The time it takes to do the movement in seconds.

Remarks

The time returned is based on the length, and the speed of the aircraft each second

CalculateTaxiwayPathTime()

This method claculates a the time it takes to go trough the taxiway-path.

```
public double CalculateTaxiwayPathTime()
```

Returns

double

The time it takes as a double-datatype.

PrintTaxiwayPathTime()

Prints out the time it takes to go trough the taxiway-path.

```
public void PrintTaxiwayPathTime()
```

ToString()

This override the ToString() method that exists in all objects in c#

```
public override string ToString()
```

Returns

string

A String with simple details about the Flight.

Class Flight.Arriving

Namespace: BrusOgPotetgull.AirportLiberary

Assembly: brusOgPotetgull.airportLiberary.dll

The Arriving-class represents an arriving flight. The class inherits from the Flight-class.

```
public class Flight.Arriving : Flight
```

Inheritance

object ← Flight ← Flight.Arriving

Inherited Members

Flight.FlightId , Flight.ActiveAircraft , Flight.DateTimeFlight , Flight.IsArrivingFlight , Flight.Length , Flight.taxiwayPath , Flight.Clock , Flight.CalculateFlightMovement(int, int, int, int) , Flight.CalculateTaxiwayPathTime() , Flight.PrintTaxiwayPathTime() , Flight.ToString() , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object)

Constructors

Arriving(Aircraft, DateTime, int, Airport, Gate, Taxiway, Runway)

Creates an arriving flight object.

```
public Arriving(Aircraft activeAircraft, DateTime dateTimeFlight, int length, Airport
arrivalAirport, Gate arrivalGate, Taxiway arrivalTaxiway, Runway arrivalRunway)
```

Parameters

`activeAircraft` Aircraft

The aircraft that is used for this flight

`dateTimeFlight` DateTime

Date of the flight.

`length` int

Length of the flight im KM.

`arrivalAirport` Airport

The airport that the aircraft is arriving at.

`arrivalGate` Gate

The gate that the aircraft is arriving at.

`arrivalTaxiway` Taxiway

The taxiway that the aircraft is arriving at.

`arrivalRunway` Runway

The runway that the aircraft is arriving at.

Properties

ArrivalAirport

Gets the arival airport

```
public Airport ArrivalAirport { get; }
```

Property Value

Airport

Airport object of the arivaø airport of the flight

ArrivalGate

Gets the arival gate

```
public Gate ArrivalGate { get; }
```

Property Value

Gate

Gate object of the arival gate of the flight

ArrivalRunway

gets the arival runnway

```
public Runway ArrivalRunway { get; }
```

Property Value

Runway

Runway object of the arival runway of the flight

ArrivalTaxiway

Gets the arival taxiway

```
public Taxiway ArrivalTaxiway { get; }
```

Property Value

Taxiway

Taxiway object of the arival taxiway of the flight

Methods

PrintFlightInformation()

Prints information about the flight. This includes the date, flights ID, length of the flight, modelname, runway id, taxiway id and gate id.

```
public void PrintFlightInformation()
```

Class Flight.Departing

Namespace: BrusOgPotetgull.AirportLiberary

Assembly: brusOgPotetgull.airportLiberary.dll

The Departuring-class represents a departing flight. The class inherits from the Flight-class.

```
public class Flight.Departing : Flight
```

Inheritance

object ← Flight ← Flight.Departing

Inherited Members

Flight.FlightId , Flight.ActiveAircraft , Flight.DateTimeFlight , Flight.IsArrivingFlight , Flight.Length , Flight.taxiwayPath , Flight.Clock , Flight.CalculateFlightMovement(int, int, int, int) , Flight.CalculateTaxiwayPathTime() , Flight.PrintTaxiwayPathTime() , Flight.ToString() , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object)

Constructors

Departing(Aircraft, DateTime, int, Airport, Gate, Taxiway, Runway)

Creates a departing flight object.

```
public Departing(Aircraft activeAircraft, DateTime dateTimeFlight, int length, Airport
departureAirport, Gate departureGate, Taxiway departureTaxiway, Runway departureRunway)
```

Parameters

`activeAircraft` Aircraft

The aircraft that is used for this flight.

`dateTimeFlight` DateTime

Date of the flight.

`length` int

Length of the flight im KM.

`departureAirport` Airport

The airport that the aircraft departure from.

`departureGate` Gate

The gate that the aircraft departure from.

`departureTaxiway` Taxiway

The taxiway that the aircraft is using to departure from.

`departureRunway` Runway

The runway that the aircraft is departuring from.

Properties

DepartureAirport

Gets departure airport

```
public Airport DepartureAirport { get; }
```

Property Value

Airport

Airport object of the departure airport for the flight

DepartureGate

Gets departure gate

```
public Gate DepartureGate { get; }
```

Property Value

Gate

Gate object of the departure gate for the flight

DepartureRunway

Gets departure runway

```
public Runway DepartureRunway { get; }
```

Property Value

Runway

Runway object of the departure runway for the flight

DepartureTaxiway

Gets departure taxiway

```
public Taxiway DepartureTaxiway { get; }
```

Property Value

Taxiway

Taxiway object of the departure taxiway for the flight

Methods

PrintFlightInformation()

Prints information about the flight. This includes the date, flights ID, length of the flight, modelname, runway id, taxiway id and gate id.

```
public void PrintFlightInformation()
```

Class Gate

Namespace: BrusOgPotetgull.AirportLiberary

Assembly: brusOgPotetgull.airportLiberary.dll

The gate class is used to define how a gate is designed. It holds fields for the status of the gate and allowed aircraft types.

```
public class Gate
```

Inheritance

object ← Gate

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object)

Constructors

Gate(string, Airport)

Creates a gate.

```
public Gate(string name, Airport airport)
```

Parameters

`name` string

string of gate name

`airport` Airport

Airport object

Properties

Id

Gets Id of gate

```
public int Id { get; }
```

Property Value

int

int of gate id

IsAvailable

Gets if gate is available with bool

```
public bool IsAvailable { get; }
```

Property Value

bool

bool of if gate is available

Name

gets gate name

```
public string Name { get; }
```

Property Value

string

string of gate name

Methods

AddAircraftAllowedAtGate(AircraftType)

Adds an aircraft that will be able to use the gate.

```
public void AddAircraftAllowedAtGate(AircraftType aircraftType)
```

Parameters

`aircraftType` AircraftType

An Enum that represents the id of an aircraftType that you want to enable accsess for the gate.

AddMultipleAircraftAllowedAtGate(List<AircraftType>)

Adds multiple aircrafts that will be granted access to use the gate.

```
public void AddMultipleAircraftAllowedAtGate(List<AircraftType> aircraftTypeIds)
```

Parameters

`aircraftTypeIds` List<AircraftType>

A list of ids of aircrafts that you want to enable accsess for the gate

BookGate(Aircraft, DateTime)

An aircraft occupies a gate. And saves it in aircrafthistory for the aircraft. The gate is now unavalible for other aircrafts to use it.

```
public void BookGate(Aircraft aircraft, DateTime time)
```

Parameters

`aircraft` Aircraft

The aircraft that is going to book the gate.

`time` DateTime

Used to log the history for the aircraft.

CheckAircraftAllowedAtGate(Aircraft)

Checks if an aircraft can use the gate.

```
public bool CheckAircraftAllowedAtGate(Aircraft aircraft)
```

Parameters

`aircraft` Aircraft

The aircraft you want to check if it has access or not.

Returns

bool

'true' if it has access or 'false' if it does not.

LeaveGate(Aircraft, DateTime)

An aircraft leaves the gate. And saves it in the aircrafthistory for the aircraft. The gate is now avalible for other aircrafts.

```
public void LeaveGate(Aircraft aircraft, DateTime time)
```

Parameters

`aircraft` Aircraft

The aircraft that is going to leave the gate.

`time` DateTime

Used to log the history for the aircraft.

MakeAllAircraftTypesAllowedForThisGate()

Grants all of the existing aircrafttypes access to use the gate.

```
public void MakeAllAircraftTypesAllowedForThisGate()
```

PrintGateInformation()

Prints out the information about the gate.

```
public void PrintGateInformation()
```

RemoveAircraftAllowedAtGate(AircraftType)

Removes an aircraft from being able to use the gate.

```
public void RemoveAircraftAllowedAtGate(AircraftType aircraftTypeId)
```

Parameters

`aircraftTypeId` AircraftType

The id of an type of aircraft that you want to deny accsess to the gate.

ToString()

This override the ToString() method that exists in all objects in c#

```
public override string ToString()
```

Returns

string

A String with simple details about the Gate.

UpdateLocation(string)

Updates the airport the gate is located at.

```
public void UpdateLocation(string airportName)
```

Parameters

`airportName` string

Name of the airport that the gate will be updated to.

Class Runway

Namespace: BrusOgPotetgull.AirportLiberary

Assembly: brusOgPotetgull.airportLiberary.dll

The runway class is used to define how a runway is designed. It is also used to conduct operations on the runway.

```
public class Runway
```

Inheritance

object ← Runway

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object)

Constructors

Runway(string, int, Airport)

creates a runway.

```
public Runway(string name, int length, Airport airport)
```

Parameters

`name` string

The name of the runway (meters).

`length` int

The length of the runway (meters).

`airport` Airport

The airport that the taxiway will be located at.

Properties

Id

Gets the Id of the runway.

```
public int Id { get; }
```

Property Value

int

int of id of runwat

InUse

Gets a bool value i wheter the runway is in use or not.

```
public bool InUse { get; }
```

Property Value

bool

bool if runway is in use

Length

Gets the length of the runway.

```
public int Length { get; }
```

Property Value

int

int of length of runway

Name

Gets the name of the runway.

```
public string Name { get; }
```

Property Value

string

string of name of runway

RunwayQueue

Gets the queue of flights waiting for the runway.

```
public Queue<Flight> RunwayQueue { get; }
```

Property Value

Queue <Flight>

Methods

AddFlightToQueue(Flight)

Adds a flight to the runway-queue.

```
public void AddFlightToQueue(Flight flight)
```

Parameters

`flight` Flight

The flight you want to add to the runwayqueue.

CheckNextFlightInQueue()

Returns the first flight in the runwayqueue.

```
public Flight CheckNextFlightInQueue()
```

Returns

Flight

Flight object that is first in line at the queue.

ExitRunway(Flight, DateTime)

Method to signal that an aircraft has left the runway. Set the field inUse to false and logs to event

```
public void ExitRunway(Flight flight, DateTime time)
```

Parameters

`flight` Flight

Is the aircraft that is leaving the runway.

`time` DateTime

Is used to log the history of the aircraft.

Remarks

If the flight is a departing flight, the method RaiseFlightDeparted() triggers the FlightDeparted event.

GetAirportNameAndRunwayId()

Returns the airport location aswell as the runwayname and id.

```
public string GetAirportNameAndRunwayId()
```

Returns

string

String that contain information about the runway.

NextFlightEntersRunway()

This method lets the next flight in queue enter the runway.

```
public void NextFlightEntersRunway()
```

PrintRunwayInformation()

Prints the information about the Runway.

```
public void PrintRunwayInformation()
```

RaiseFlightArrived(Arriving, DateTime, string)

Method to trigger the event FlightArrived

```
protected virtual void RaiseFlightArrived(Flight.Arriving flight, DateTime time,
string message)
```

Parameters

`flight` Flight.Arriving

The flight which triggers the event

`time` DateTime

Time of the event

`message` string

Message of what occured at the time of the event

RaiseFlightDeparted(Departing, DateTime, string)

Method to trigger the event FlightDeparted

```
protected virtual void RaiseFlightDeparted(Flight.Departing flight, DateTime time,
string message)
```

Parameters

`flight` Flight.Departing

The flight which triggers the event

`time` DateTime

Time of the event

`message` string

Message of what occured at the time of the event

RemoveFromQueue()

Removes the first flight in the runwayqueue and returns it.

```
public Flight RemoveFromQueue()
```

Returns

Flight

Flight object that is removed from the beginning of the queue.

SimulateRunwayTime(Flight, int, int, int)

Returns the time in seconds that an aircraft uses on the runway. Given the length of runway is meters, and speed / speedChange is kph.

```
public double SimulateRunwayTime(Flight flight, int initialSpeed, int speedChange,
```

```
int maxSpeed)
```

Parameters

`flight` Flight

The current flight.

`initialSpeed` int

The speed at which the aircraft starts with (Kp/h).

`speedChange` int

The change in speed (Kp/h).

`maxSpeed` int

Maximum speed for this calculation (Kp/h).

Returns

double

Returns the method flight.CalculateFlightMovement() which is the time spent on the runway in seconds.

ToString()

This override the ToString() method that exists in all objects in c#

```
public override string ToString()
```

Returns

string

A String with simple details about the Runway.

UpdateLocation(string)

Updates the airport the Runway is located at.

```
public void UpdateLocation(string airportName)
```

Parameters

`airportName` string

Name of the airport that you want to update the location to.

UseRunway(Flight, DateTime)

Method to signal that an aircraft is using the runway. Sets the field inUse to true and logs the event.

```
public void UseRunway(Flight flight, DateTime time)
```

Parameters

`flight` Flight

Is the aircraft that uses the runway.

`time` DateTime

Is used to log the history of the aircraft.

Remarks

If the flight is an arriving flight. The method RaiseFlightArrived is used to handle the event and logging.

Events

FlightArrived

Event to be used when a flight is ariving

```
public event EventHandler<ArrivingEventArgs>? FlightArrived
```

Event Type

EventHandler <ArrivingEventArgs>

FlightDeparted

Event to be used when a flight is departing

```
public event EventHandler<DepartingEventArgs>? FlightDeparted
```

Event Type

EventHandler <DepartingEventArgs>

Class Taxiway

Namespace: BrusOgPotetgull.AirportLiberary

Assembly: brusOgPotetgull.airportLiberary.dll

The taxiway class is used to define how a taxiway is designed. It is also used to conduct operations on the taxiway.

```
public class Taxiway
```

Inheritance

object ← Taxiway

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object)

Remarks

This is used to create the network of taxiways through connectionpoints and the lists of runway and gate connections.

Constructors

Taxiway(string, int, int, Airport)

Creates a taxiway.

```
public Taxiway(string name, int length, int maxSpeed, Airport airport)
```

Parameters

`name` string

The name of the taxiway.

`length` int

Length of the taxiway (meters).

`maxSpeed` int

Legal maxspeed for the taxiway (Kp/h).

`airport` Airport

The airport that the taxiway will be located at.

Fields

connectedGates

List of connected gates

```
public List<Gate> connectedGates
```

Field Value

List<Gate>

list of gate objects

connectedRunways

list of connected runways

```
public List<Runway> connectedRunways
```

Field Value

List<Runway>

list of runway objects

Properties

A

Gets or sets connectionpoint A

```
public ConnectionPoint A { get; set; }
```

Property Value

ConnectionPoint

connection point objet conected to end of road

B

Gets or sets connectionpoint B

```
public ConnectionPoint B { get; set; }
```

Property Value

ConnectionPoint

connection point object connected to other end of road

Id

gets Id of taxiway

```
public int Id { get; }
```

Property Value

int

int of id of taxiway

Length

Gets the length of the Taxiway

```
public int Length { get; }
```

Property Value

int

int of length of taxiway

MaxSpeed

Gets max speed on taxiway

```
public int MaxSpeed { get; }
```

Property Value

int

int of max sped on taxiway

Name

Gets name of taxiway

```
public string Name { get; }
```

Property Value

string

string of taxiway name

Methods

AddConnectedGate(Gate)

Adds a gate to the list of connected gates for this taxiway.

```
public void AddConnectedGate(Gate gate)
```

Parameters

`gate` Gate

The gate that will be added to the list.

AddConnectedRunway(Runway)

Adds a runway to the list of connected runways for this taxiway.

```
public void AddConnectedRunway(Runway runway)
```

Parameters

`runway` Runway

The runway that will be added to the list.

AddFlightToQueue(Flight, DateTime)

Adds an flight to the taxiwayqueue.

```
public void AddFlightToQueue(Flight flight, DateTime time)
```

Parameters

`flight` Flight

The flight that is insertet into the queue.

`time` DateTime

Used to log the time the aircraft entered the queue.

CheckNextFlightInQueue()

This method checks which flight is next in line for this taxiway.

```
public Flight CheckNextFlightInQueue()
```

Returns

Flight

The next flight object in the taxiwayqueue.

GetNumberOfAircraftsInQueue()

Gets the number of aircrafts in the queue.

```
public int GetNumberOfAircraftsInQueue()
```

Returns

int

Returns the number of aircrafts as an int value

NextFlightLeavesTaxiway(Flight, DateTime)

The aircraft first in line for taxiwayqueue leaves the taxiwayqueue.

```
public void NextFlightLeavesTaxiway(Flight flight, DateTime time)
```

Parameters

`flight` Flight

Used to log correct history for the used aircraft.

`time` DateTime

Used to log correct history for the used aircraft.

PrintTaxiwayInformation()

Prints the information about the taxiway.

```
public void PrintTaxiwayInformation()
```

RemoveConnectedGate(Gate)

Removes a gate from the list of connected gates for this taxiway.

```
public void RemoveConnectedGate(Gate gate)
```

Parameters

`gate` Gate

The gate that will be removed from the list.

RemoveConnectedRunway(Runway)

Removes a runway from the list of connected runways for this taxiway.

```
public void RemoveConnectedRunway(Runway runway)
```

Parameters

`runway` Runway

The runway that will be removed from the list.

SimulateTaxiwayTime(Flight, int, int, int, DateTime)

Simulates an aircraft using the taxiway and returns the time spent in seconds. This also logs when the aircraft starts using the taxiway.

```
public double SimulateTaxiwayTime(Flight flight, int initialSpeed, int speedChange, int
```

```
maxSpeed, DateTime time)
```

Parameters

`flight` Flight

The flight thats is using the taxiway.

`initialSpeed` int

The speed at which the aircraft starts with (Kp/h).

`speedChange` int

The change in speed (Kp/h).

`maxSpeed` int

Maximum speed for this calculation (Kp/h).

`time` DateTime

Time when the aircraft starts using the taxiway.

Returns

double

Returns the time spent on the taxiway in seconds (type = double)

ToString()

This override the ToString() method that exists in all objects in c#

```
public override string ToString()
```

Returns

string

A String with simple details about the Taxiway.

UpdateLocation(string)

Updates the airport the taxiway is located at.

```
public void UpdateLocation(string airportName)
```

Parameters

`airportName` string

Name of the airport the taxiway updates to.

Class Terminal

Namespace: BrusOgPotetgull.AirportLiberary

Assembly: brusOgPotetgull.airportLiberary.dll

The terminal class is an area in the airport that can host a set of gates.

```
public class Terminal
```

Inheritance

object ← Terminal

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object)

Constructors

Terminal(string, Airport)

Creates a terminal.

```
public Terminal(string name, Airport airport)
```

Parameters

`name` string

The name for the terminal.

`airport` Airport

The airport that the terminal will be located at.

Properties

Id

Gets Id of Terminal

```
public int Id { get; }
```

Property Value

int

int value of the id of the terminal

Name

Gets Name of Terminal

```
public string Name { get; }
```

Property Value

string

string of name of terminal

Methods

AddAircraftAllowedAtGatesAtTerminal(AircraftType)

Adds an aircraft that will be able to use all of the gates in this terminal.

```
public void AddAircraftAllowedAtGatesAtTerminal(AircraftType aircraftType)
```

Parameters

`aircraftType` AircraftType

An Enum that represents the id of an aircraftType that you want to enable accsess for the gate.

AddGateToList(Gate)

Adds a gate to the list of gates on this terminal.

```
public void AddGateToList(Gate gate)
```

Parameters

`gate` Gate

The gate that will be added to the list.

CreateMultipleGatesToTerminal(string, int, int, Airport)

Creating multiple gates and adding them to this terminal.

```
public void CreateMultipleGatesToTerminal(string gateLetter, int startNumber, int
numberOfGates, Airport airport)
```

Parameters

`gateLetter` string

The letter thats a part of the gatename.

`startNumber` int

The start-number for generating all the gates. This is gonna be the first generated gate.

`numberOfGates` int

The number of gates thats gonna be created.

`airport` Airport

The airport that these gates will be added to.

GetgatesInTerminal()

Gets the list that contains all gates for this Terminal.

```
public List<Gate> GetgatesInTerminal()
```

Returns

List<Gate>

A list of gates for this terminal.

PrintTaxiwayInformation()

Prints out information about the terminal.

```
public void PrintTaxiwayInformation()
```

ToString()

This override the ToString() method that exists in all objects in c#

```
public override string ToString()
```

Returns

string

A String with simple details about the Terminal.

UpdateGateLocation(string)

Updates the airport the terminal is located at.

```
public void UpdateGateLocation(string airportName)
```

Parameters

`airportName` string

Name of the airport that the terminal will be updated to.

UpdateLocation(string)

Updates the information for which airport the terminal is located at.

```
public void UpdateLocation(string airportName)
```

Parameters

`airportName` string

Name of the airport that the terminal will be located at.

Namespace BrusOgPotetgull.AirportLiberary.AircraftTypes

Classes

AircraftType

This class represents a spesific type of aircraft.

Class AircraftType

Namespace: BrusOgPotetgull.AirportLiberary.AircraftTypes

Assembly: brusOgPotetgull.airportLiberary.dll

This class represents a spesific type of aircraft.

```
public class AircraftType
```

Inheritance

object ← AircraftType

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object)

Examples

This is how you can instansiate the aircraft type for a boeing 737. `AircraftType boeing737 = new AircraftType("Boeing 737");`

Remarks

This is used together with the creation of aircrafts to define their type.

Constructors

AircraftType(string)

Creates an Aircraft type to be used in the creation of aircraft objects

```
public AircraftType(string name)
```

Parameters

`name` string

This is the name of the aircraft type. An Example could be "Airbus A330"

Properties

Name

Gets the name of the aircraft type

```
public string Name { get; }
```

Property Value

string

string of name of aircraft type

TypeId

Gets the TypeId of the aircraft type

```
public int TypeId { get; }
```

Property Value

int

int of aircraft type id

Methods

ToString()

This override the ToString() method that exists in all objects in c#

```
public override string ToString()
```

Returns

string

A String with simple details about the AircraftType.

Namespace BrusOgPotetgull.AirportLiberary.CustomExceptions

Classes

DuplicateOfContentException

Initializes an exception that is used when dulpication of content happens.

NegativeNumberException

Initializes an exception that is used when a negative number appears.

Class DuplicateOfContentException

Namespace: BrusOgPotetgull.AirportLiberary.CustomExceptions

Assembly: brusOgPotetgull.airportLiberary.dll

Initializes an exception that is used when dulpication of content happens.

```
public class DuplicateOfContentException : Exception, ISerializable
```

Inheritance

object ← Exception ← DuplicateOfContentException

Implements

ISerializable

Inherited Members

Exception.GetBaseException() , Exception.GetType() , Exception.ToString() , Exception.Data , Exception.HelpLink , Exception.HResult , Exception.InnerException , Exception.Message , Exception.Source , Exception.StackTrace , Exception.TargetSite , Exception.SerializeObjectState , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.MemberwiseClone() , object.ReferenceEquals(object, object)

Constructors

DuplicateOfContentException()

Creates an DuplicateOfCotentException.

```
public DuplicateOfContentException()
```

DuplicateOfContentException(string)

Creates an DuplicateOfCotentException with a specified error message.

```
public DuplicateOfContentException(string message)
```

Parameters

`message` string

This error message is explaining the reason for the exception.

DuplicateOfContentException(string, Exception)

Creates an DuplicateOfCotentException with a specified error message. It has also a reference to the inner exception that is the cause of this exception.

```
public DuplicateOfContentException(string message, Exception inner)
```

Parameters

`message` string

This error message is explaining the reason for the exception.

`inner` Exception

The exception that is the reason for the current exception.

Class NegativeNumberException

Namespace: BrusOgPotetgull.AirportLiberary.CustomExceptions

Assembly: brusOgPotetgull.airportLiberary.dll

Initializes an exception that is used when a negative number appears.

```
public class NegativeNumberException : ArgumentOutOfRangeException, ISerializable
```

Inheritance

object ← Exception ← SystemException ← ArgumentException ← ArgumentOutOfRangeException ← NegativeNumberException

Implements

ISerializable

Inherited Members

ArgumentOutOfRangeException.ThrowIfEqual<T>(T, T, string) ,
ArgumentOutOfRangeException.ThrowIfGreaterThan<T>(T, T, string) ,
ArgumentOutOfRangeException.ThrowIfGreaterThanOrEqual<T>(T, T, string) ,
ArgumentOutOfRangeException.ThrowIfLessThan<T>(T, T, string) ,
ArgumentOutOfRangeException.ThrowIfLessThanOrEqual<T>(T, T, string) ,
ArgumentOutOfRangeException.ThrowIfNegative<T>(T, string) ,
ArgumentOutOfRangeException.ThrowIfNegativeOrZero<T>(T, string) ,
ArgumentOutOfRangeException.ThrowIfNotEqual<T>(T, T, string) ,
ArgumentOutOfRangeException.ThrowIfZero<T>(T, string) ,
ArgumentOutOfRangeException.ActualValue , ArgumentOutOfRangeException.Message ,
ArgumentException.ThrowIfNullOrEmpty(string, string) ,
ArgumentException.ThrowIfNullOrWhiteSpace(string, string) , ArgumentException.ParamName ,
Exception.GetBaseException() , Exception.GetType() , Exception.ToString() , Exception.Data ,
Exception.HelpLink , Exception.HResult , Exception.InnerException , Exception.Source ,
Exception.StackTrace , Exception.TargetSite , Exception.SerializeObjectState ,
object.Equals(object) , object.Equals(object, object) , object.GetHashCode() ,
object.MemberwiseClone() , object.ReferenceEquals(object, object)

Constructors

NegativeNumberException()

Creates an NegativeNumberException

```
public NegativeNumberException()
```

NegativeNumberException(string)

Creates an NegativeNumberException with a specified error message.

```
public NegativeNumberException(string message)
```

Parameters

`message` string

This error message is explaining the reason for the exception.

NegativeNumberException(string, Exception)

Creates an NegativeNumberException with a specified error message. It has also a reference to the inner exception that is the cause of this exception.

```
public NegativeNumberException(string message, Exception inner)
```

Parameters

`message` string

This error message is explaining the reason for the exception.

`inner` Exception

The exception that is the reason for the current exception.

Namespace BrusOgPotetgull.AirportLiberary.Simulation

Classes

Simulation

A simulation that is used to simulate how an airport works.

Class Simulation

Namespace: BrusOgPotetgull.AirportLiberary.Simulation

Assembly: brusOgPotetgull.airportLiberary.dll

A simulation that is used to simulate how an airport works.

```
public class Simulation
```

Inheritance

object ← Simulation

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

Simulation(Airport, DateTime, DateTime)

Creates an simulation of the choosen airport.

```
public Simulation(Airport airport, DateTime startTime, DateTime endTime)
```

Parameters

`airport` Airport

Which airport that is using the simulation.

`startTime` DateTime

The day that is the start of the simulation.

`endTime` DateTime

The day that is the end of the simulation.

Properties

Airport

Gets the airport sim is running on

```
public Airport Airport { get; }
```

Property Value

Airport

airport object that the sim is running on

EndTime

gets end time of simulation

```
public DateTime EndTime { get; }
```

Property Value

DateTime

DateTime object of the time the sim is ending

StartTime

Gets start time of simulation

```
public DateTime StartTime { get; }
```

Property Value

DateTime

DateTime object of the start time of the sim

Methods

RunSimulation()

This method is starting the simulation.

```
public void RunSimulation()
```