Airport Traffic Simulator

[Design Document]

Teacher:

Chung Kuah

ProCP

19/02/2018

Developers:Yoanna Borisova,Teodor Genov,Vladimir Katrandzhiev,Monika Kerulyte,Ignas Kybransas,Rostislav Tinchev

Contents

[Overview 3](#_Toc513414284)

[Class Diagrams 3](#_Toc513414285)

[UML Description 4](#_Toc513414286)

[Sequence Diagram 6](#_Toc513414287)

[Screen by Screen Specification 7](#_Toc513414288)

[Log in Screen 7](#_Toc513414289)

[Simulation Screen 7](#_Toc513414290)

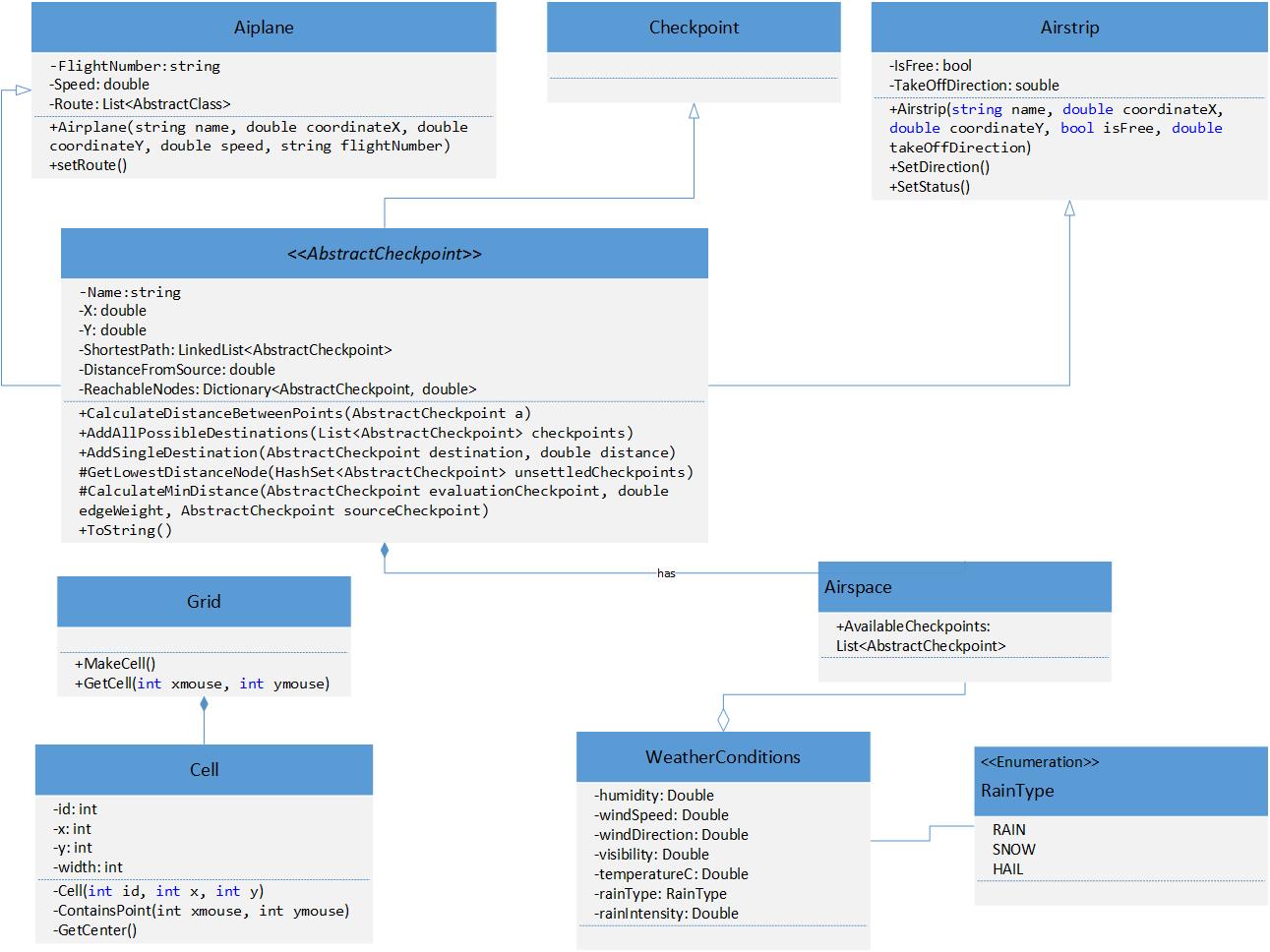
# Overview

Airport Traffic Simulator is an application that is used at the airport, by employees at the Control Tower to regulate air traffic.

This spec is going to be updated throughout the entire course until realizing the project. The graphics and layout of the screens is shown here merely to illustrate the underlying functionality. The actual look and feel will be developed over time.

This spec discusses what the application will contain visually, how the user can interact with it and how the application is structured behind the visualization.

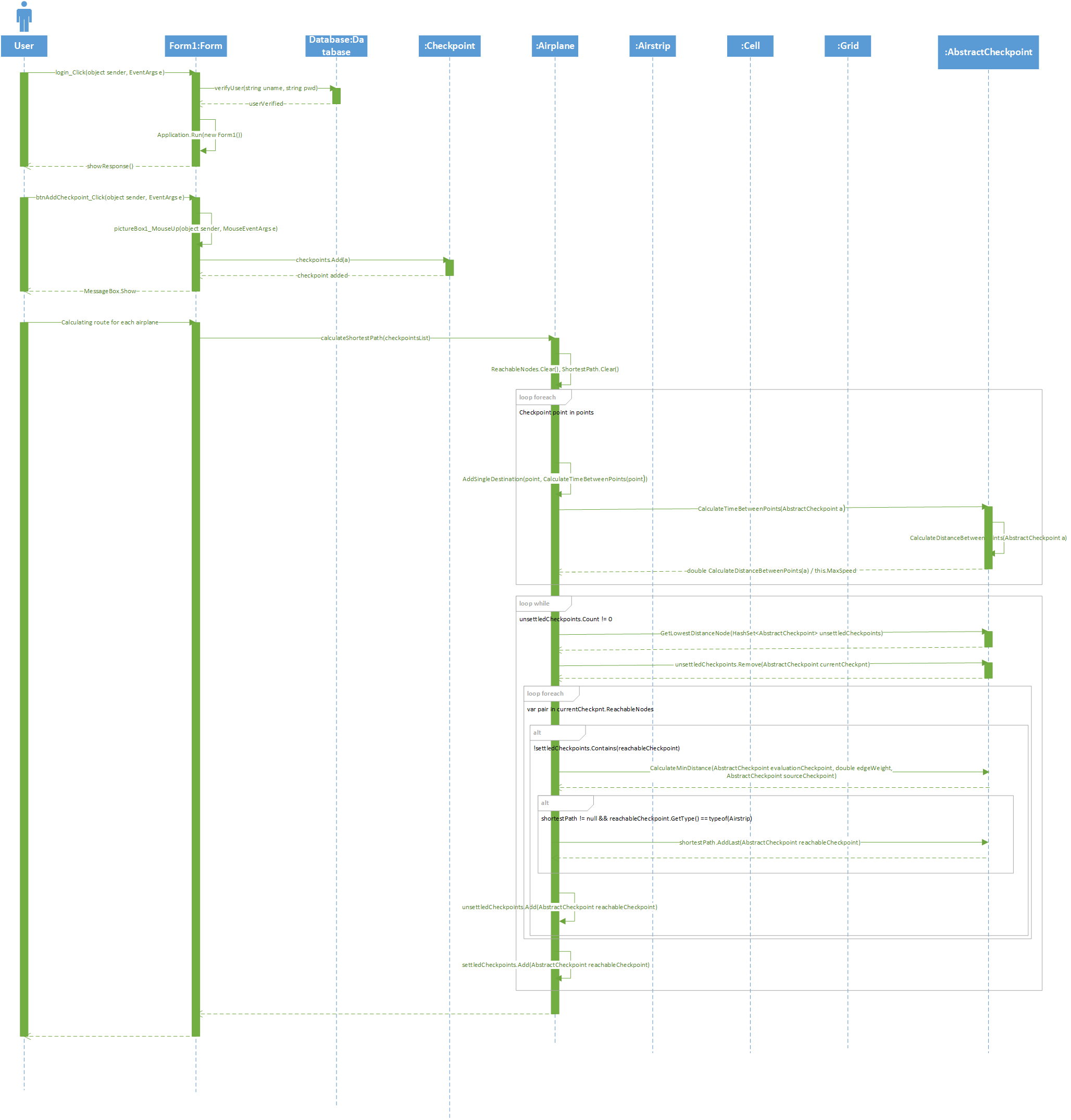
# Class Diagrams



# UML Description

|  |  |
| --- | --- |
| CLASS | CONTENTS |
| Airport   1. name 2. planeLand() 3. planeTakeOff() 4. addStrip() 5. removeStrip() | Name simply resembles the name of the airport, where planes will land. Method planeLand() is the method that is used to indicate if any plane is landing. Method planeTakeOff() is used to indicate if any plane is taking off. Addstrip() method would simply add another strip to the airport. RemoveStrip() would simply remove chosen strip from the aiport. Every airport must contain at least two airstrips. |
| Airstrip   1. type 2. isFree 3. takeOffDirection 4. landingDirection 5. setStatus() 6. switchDirections() | In order for an airstrip to be created there must be an airport. Type represents the type of the landing strip whether it is for landing or taking off. IsFree contains information whether or not the landing/taking off strip is free. TakeOffDirection is the direction of the take-off airstrip. LandingDirection is the direction of the landing strip.  Method setStatus() sets the status of each airstrip to either taken of free. SwitchDirections() switches the directions of each airstrip. |
| Airplane   1. name 2. coordinateX 3. coordinateY 4. speed 5. route 6. flightNumber 7. setroute() | Each airplane will be landing on an airstrip of type landing. CoordinateX and coordinateY are the coordinates of the plane itself. Speed and route are the speed of the plane and route respectively. FlightNumber would be its flight number. Method setRoute() sets the route ass chosen by the user. |
| Airspace   1. airport 2. availableCheckpoitns 3. airplanesInTheAir 4. weatherConditions 5. addAirplane() 6. addCheckpoint() 7. removeCheckpoint() 8. changeWeatherConditions() | Airport is the airport itself situated in the airspace. AvailableCheckpoints represents the array of checkpoints that the airplanes will have to follow. AirplanesInTheAir represents an array of airplanes that are in the air at this moment. WeatherConditions symbolizes the weather conditions. Methods addAirplane() and addCheckpoint() respectively are responsible for adding airplanes and checkpoints to the airspace. Method removeCheckpoint is used by the user for removing checkpoints from the airspace. |
| WeatherConditions   1. humidity 2. windSpeed 3. windDirection 4. visibility 5. temperature 6. rainType 7. rainIntensity | Attribute humidity represents the level of humidity of the weather. WindSpeed and windDirection represent the speed of the wind and direction respectively. Visibility – the distance to which everything is visible. Temperature is the temperature of the air. |
| ICheckpoint   1. name 2. coordinateX 3. coordinateY |  |
| RainType   1. RAIN 2. SNOWFALL 3. HALE |  |

# Sequence Diagram



# Screen by Screen Specification

The final product will contain a total of two screens. Both screens will have the same design despite some changes that are going to be made, which suit the needs of each one so that they achieve their goals.

All of the screens are created in Windows Form Application.

# Log in Screen

The login screen’s purpose is to verify the employee’s data. It is going to be a simple login form where after the data is verified, the screen switches to the Simulation Screen

# Simulation Screen

The simulation screen contains left and right side menus and a grid. Through the left side menu, the user can control the outside variables like weather. On there, the user can also find a play button, when on click, initiates the whole simulation process. On the right side menu, the user can find buttons that implement the “add”, “remove, “save”, “upload”, “probability” functions. The “add” function is used for the user to enter editing mode so that they can add checkpoints. After the user has entered editing mode, the start button is substituted with a stop button that implements a function that exits the editing mode. The “remove” function is used to remove a checkpoint from the grid. “save” and “upload” functions are used to read checkpoints from a file or upload the current state to a file, respectively.