**Airport Challenge - Task**

We have a request from a client to write the software to control the flow of planes at an airport. The planes can land and take off provided that the weather is sunny. Occasionally it may be stormy, in which case no planes can land or take off. Here are the user stories that we worked out in collaboration with the client:

As an air traffic controller

So I can get passengers to a destination

I want to instruct a plane to land at an airport

|  |  |
| --- | --- |
| **Object** | **Message** |
| * air traffic controller |  |
| * plane |  |
| * airport | * **land-plane** |

airport 🡨 **land-plane** 🡪 plane landed

Looking at the message above, I now know that we will need to have an **airport** to **land** a plane.

Tests required:

* Do I have an airport?
* Do I have a plane?
* Can I land plane in airport?

1. Create a test file ‘*airport\_spec.rb*’ to test if we have a class airport.
2. **Run rspec:** got (NameError – uninitialized constant Airport) (**RED**)
3. The error above occurred because there isn’t yet a file ‘*airport.rb*’ with the class ‘Airport’.

To resolve this error, a file ‘*airport.rb*’ needs to be created in the lib folder.

The ‘*airport\_spec.rb*’ file also needs to have (required ‘filename’) added to it.

1. **Run rspec:** got no errors, (**GREEN**)

* When running Feature test in pry, Ruby was now able to find the Airport class and created a new instance of it.

A screenshot of a cell phone

Description automatically generated

1. **Time to refactor**
2. Still in Feature test tried to ‘.land\_plane’ in our new ‘lisbon\_airport’

(got NoMethodError: undefined method ‘land\_plane’)

This happened because there’s no method in ‘*airport.rb*’ to land a plane yet.

1. Added a new test to ‘*airport\_spec.rb*’, can a plane land in ‘*airport.rb*’ ?

it "allows a plane to land" do

expect(subject.land(plane)).to eq plane

end

1. **Run rspec:** got (NameError – undefined local variable or method) (**RED**)

From my understanding this error occurred because there’s no method ‘.land(argument)’ in ‘*airport.rb*’.

1. Next step is to add method ‘.land(argument)’ to ‘*airport.rb*’
2. **Run rspec:** still in the (**RED**) zone, I have a feeling I went too quick with what I needed to test for, back to step 5.

As an air traffic controller

So I can get passengers on the way to their destination

I want to instruct a plane to take off from an airport and confirm that it is no longer in the airport

|  |  |
| --- | --- |
| **Object** | **Message** |
|  |  |
|  |  |
|  |  |

As an air traffic controller

To ensure safety

I want to prevent landing when the airport is full

|  |  |
| --- | --- |
| **Object** | **Message** |
|  |  |
|  |  |
|  |  |

As the system designer

So that the software can be used for many different airports

I would like a default airport capacity that can be overridden as appropriate

|  |  |
| --- | --- |
| **Object** | **Message** |
|  |  |
|  |  |
|  |  |

As an air traffic controller

To ensure safety

I want to prevent takeoff when weather is stormy

|  |  |
| --- | --- |
| **Object** | **Message** |
|  |  |
|  |  |
|  |  |

As an air traffic controller

To ensure safety

I want to prevent landing when weather is stormy

|  |  |
| --- | --- |
| **Object** | **Message** |
|  |  |
|  |  |
|  |  |

Your task is to test drive the creation of a set of classes/modules to satisfy all the above user stories. You will need to use a random number generator to set the weather (it is normally sunny but on rare occasions it may be stormy). In your tests, you'll need to use a stub to override random weather to ensure consistent test behaviour.