

Tree: 44f79a35c9 ▾

airport_challenge / lib / **airport.rb**

Find file

Copy path

Fetching contributors...

 Cannot retrieve contributors at this time

42 lines (32 sloc) | 967 Bytes

```
1  require_relative 'plane'
2  require_relative 'weather'
3
4  class Airport
5    attr_reader :planes, :weather, :capacity
6    DEFAULT_CAPACITY = 100
7
8    def initialize(capacity = DEFAULT_CAPACITY, weather = Weather.new)
9      @capacity = capacity
10     @planes = []
11     @weather = weather
12   end
13
14   def land(plane)
15     raise "Cannot land plane that has landed!" if @planes.include? plane
16     raise "Cannot land plane while weather is stormy!" if @weather.conditions == "STORMY"
17     raise "Cannot land more planes on site than maximum capacity!" if full?
18     add_plane(plane)
19   end
20
21   def takeoff(plane)
22     raise "Cannot takeoff a plane that isn't on site!" unless @planes.include? plane
23     raise "Cannot takeoff plane while weather is stormy!" if @weather.conditions == "STORMY"
24     remove_plane(plane)
25   end
26
27   private
28
29   def full?
30     @planes.count >= @capacity
31   end
32
33   def remove_plane(plane)
34     @planes.delete(plane)
35   end
36
37   def add_plane(plane)
38     @planes << plane
39   end
40
41 end
```

Tree: 44f79a35c9 ▼

airport_challenge / lib / **plane.rb**

Find file

Copy path

Fetching contributors...

 Cannot retrieve contributors at this time

4 lines (3 sloc) | 43 Bytes

```
1  require_relative 'airport'
2  class Plane
3  end
```

Tree: 44f79a35c9 ▼

airport_challenge / lib / weather.rb

Find file

Copy path


Fetching contributors...

 Cannot retrieve contributors at this time

8 lines (5 sloc) | 106 Bytes

```
1  class Weather
2
3    def conditions
4      rand(5) > 1 ? @weather = "CLEAR" : @weather = "STORMY"
5    end
6
7  end
```

Fetching contributors...

 Cannot retrieve contributors at this time

71 lines (57 sloc) | 2.49 KB

```
1  require './lib/airport'
2  describe Airport do
3    let(:test_plane) { double(:test_plane) }
4    let(:weather) { double(:weather, :conditions => "CLEAR") }
5    subject(:airport) { described_class.new(Airport::DEFAULT_CAPACITY, weather) }
6
7    it "Should have a land method with an object to land as an argument" do
8      is_expected.to respond_to(:land).with(1).argument
9    end
10
11   it "Should not be able to land planes that have already landed" do
12     subject.land(test_plane)
13     expect { subject.land(test_plane) }.to raise_error("Cannot land plane that has landed!")
14   end
15
16   it "Should be able to view planes that have landed at it's site" do
17     subject.land(test_plane)
18     expect(subject.planes).to include test_plane
19   end
20
21   it "Should not land planes while the weather is stormy" do
22     allow(weather).to receive(:conditions).and_return("STORMY")
23     expect { subject.land(test_plane) }.to raise_error("Cannot land plane while weather is stormy!")
24   end
25
26   it "Should have a takeoff method with an object to do so as an argument" do
27     is_expected.to respond_to(:takeoff).with(1).argument
28   end
29
30   it "Should not be able to takeoff a plane that is not at the airport" do
31     subject.land(test_plane)
32     subject.takeoff(test_plane)
33     expect { subject.takeoff(test_plane) }.to raise_error("Cannot takeoff a plane that isn't on site!")
34   end
35
36   it "Should no longer have any planes that have taken off at site" do
37     subject.land(test_plane)
38     expect(subject.planes).to include test_plane
39     subject.takeoff(test_plane)
40     expect(subject.planes).not_to include test_plane
41   end
42
43   it "Should not takeoff planes while the weather is stormy" do
44     subject.land(test_plane)
45     allow(weather).to receive(:conditions).and_return("STORMY")
46     weather.conditions
47     expect { subject.takeoff(test_plane) }.to raise_error("Cannot takeoff plane while weather is stormy!")
48   end
49
50   it "Should be able to set capacity of airport" do
51     expect((Airport.new(1)).capacity).to eq 1
52   end
53
54   it "Should have a default capacity" do
55     expect((Airport.new).capacity).to eq Airport::DEFAULT_CAPACITY
56   end
57
58   it "Should not be able to land more planes on site than maximum capacity" do
59     subject = Airport.new(0)
60     expect { subject.land(test_plane) }.to raise_error("Cannot land more planes on site than maximum capacity!")
61   end
62
63   it "Should receive multiple planes" do
64     3.times { subject.land(Plane.new) }
65     expect(subject.planes.length).to eq 3
66     3.times { subject.takeoff(subject.planes[0]) }
67     expect(subject.planes.length).to eq 0
68   end
69
70 end
```

Tree: 44f79a35c9 ▼

airport_challenge / spec / **plane_spec.rb**

Find file

Copy path

Fetching contributors...



Cannot retrieve contributors at this time

4 lines (3 sloc) | 44 Bytes

```
1  require './lib/Plane'
2  describe Plane do
3    end
```