

Tree: 09eaac1204 ▼

airport_challenge / lib / Weather.rb

Find file

Copy path

Fetching contributors...




Cannot retrieve contributors at this time

9 lines (7 sloc) | 123 Bytes

```
1  require_relative 'Airport'
2  require_relative 'Plane'
3
4  class Weather
5    def stormy?
6      rand(5) == 1 ? true : false
7    end
8  end
```

Fetching contributors...

 Cannot retrieve contributors at this time

63 lines (47 sloc) | 1.12 KB

```
1  require_relative 'Plane'
2  require_relative 'Weather'
3
4  class Airport
5
6      DEFAULT_CAPACITY = 50
7      attr_reader :planes
8      attr_accessor :capacity, :weather
9
10
11     def initialize(capacity: DEFAULT_CAPACITY, weather: Weather.new)
12         @planes = []
13         @capacity = capacity
14         @weather = weather
15     end
16
17     def land(plane)
18         raise "This airport is full" if full?
19         raise "This plane cannot land until the weather clears up" if stormy?
20         expect_plane(plane)
21         plane.land(self)
22         @expected_plane = nil
23         @planes << plane
24     end
25
26     def takeoff(plane)
27         raise "That plane is not at this airport" unless @planes.include?(plane)
28         raise "This plane cannot take off until the weather clears up" if stormy?
29         send_to_runway(plane)
30         plane.takeoff
31         @on_runway = nil
32         @planes.delete(plane)
33     end
34
35     def send_to_runway(plane)
36         @on_runway = plane
37     end
38
39     def expect_plane(plane)
40         @expected_plane = plane
41     end
42
43     def plane_expected?(plane)
44         plane == @expected_plane
45     end
46
47     def on_runway?(plane)
48         plane == @on_runway
49     end
50
51     private
52
53     def stormy?
54         @weather.stormy?
55     end
56
57     def full?
58         @planes.length >= @capacity
59     end
60
61
62 end
```

Tree: 09eaac1204 ▾

airport_challenge / lib / plane.rb

Find file

Copy path

Fetching contributors...

 Cannot retrieve contributors at this time

30 lines (19 sloc) | 700 Bytes

```
1  require_relative 'Airport'
2  require_relative 'Weather'
3
4  class Plane
5
6      attr_reader :status
7
8      def initialize
9          @status = "In the air"
10     end
11
12     def land(airport)
13         raise "Plane is already on the ground" if @status != "In the air"
14         # If airport called this method, sets status to airport
15         # If plane called this method, notifies airport and makes airport call the method
16         # This way, both the plane and airport can initiate a landing
17         airport.plane_expected?(self) ? @status = airport : airport.land(self)
18     end
19
20     def takeoff
21         raise "Plane is already in the air" if @status == "In the air"
22         @status.on_runway?(self) ? @status = "In the air" : @status.takeoff(self)
23     end
24
25
26
27
28
29     end
```


Tree: 09eaac1204 ▾

airport_challenge / spec / airport_spec.rb

Find file

Copy path


Fetching contributors...

 Cannot retrieve contributors at this time

93 lines (67 sloc) | 2.44 KB

```
1  require 'airport'
2
3  describe Airport do
4    let :plane {double(:plane, takeoff: nil, land: nil, status: subject)}
5    let :weather {double(:weather, stormy?: false)}
6    subject {Airport.new(weather: weather)}
7
8    it "has no planes when it first opens" do
9      expect(subject.planes).to be_empty
10   end
11
12   describe '#land' do
13     it "can have a plane land at it" do
14       expect(subject.land(plane)).to include plane
15     end
16
17     it "should be able to order planes to land" do
18       subject.land(plane)
19       expect(plane.status).to eq subject
20     end
21
22     it "stops planes landing in stormy conditions" do
23
24       allow(weather).to receive(:stormy?) { true }
25       expect{ subject.land(plane) }.to raise_error "This plane cannot land until the weather clears up"
26     end
27
28   end
29
30   describe '#takeoff' do
31     it "lets a plane take off from the airport" do
32       subject.land(plane)
33       expect(subject.takeoff(plane)).to eq plane
34     end
35
36     it "orders the correct plane to take off" do
37       another_plane = double(:another_plane, takeoff: nil, land: nil)
38       subject.land(plane)
39       subject.land(another_plane)
40       subject.takeoff(plane)
41       expect(subject.planes).to include another_plane
42     end
43
44     it "cannot order an absent plane to take off" do
45       expect { subject.takeoff(plane) }.to raise_error "That plane is not at this airport"
46     end
47
48     it "stops planes taking off in stormy conditions" do
49       subject.land(plane)
50       allow(weather).to receive(:stormy?) { true }
51       expect{ subject.takeoff(plane) }.to raise_error "This plane cannot take off until the weather clears up"
52     end
53
54   end
55
56   it "should be able to expect a plane" do
57     expect(subject).to respond_to :expect_plane
58   end
59
60   it "should be able to send planes to the runway" do
61     expect(subject).to respond_to :send_to_runway
62   end
63
64   describe 'initialisation' do
65
66     it "defaults the capacity to 50 if other capacity not specified" do
67       expect(subject.capacity).to eq Airport::DEFAULT_CAPACITY
68     end
69
70     it "sets the capacity to 10" do
71       airport = Airport.new(capacity: 10)
72       expect(airport.capacity).to eq 10
73     end
74
75     it "has a variable capacity" do
76       Airport::DEFAULT_CAPACITY.times { subject.land(plane) }
77       expect { subject.land(plane) }.to raise_error "This airport is full"
78     end
79
80     it "has a variable capacity" do
81       subject.capacity = 100
82       100.times { subject.land(plane) }
83       expect { subject.land(plane) }.to raise_error "This airport is full"
84     end
85
86
87   end
88
89
90
91
92 end
```

Fetching contributors...

 Cannot retrieve contributors at this time

55 lines (40 sloc) | 1.47 KB

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

```
require 'plane'

describe Plane do

  it "should be flying when first created" do
    expect(subject.status).to eq "In the air"
  end

  describe '#land' do

    it "shouldnt be able to land if it is on the ground" do
      airport = double(:airport, plane_expected?: true)
      subject.land(airport)
      expect { subject.land(airport) }.to raise_error "Plane is already on the ground"
    end

    it "should be able to land at a specific airport" do
      airport = double(:airport, plane_expected?: true)
      subject.land(airport)
      expect(subject.status).to eq airport
    end

    it "checks a plane has been expected before landing it" do
      airport = double(:airport, plane_expected?: true)
      expect(airport).to receive(:plane_expected?).with(subject)
      subject.land(airport)
    end
  end

  describe '#takeoff' do

    it "shouldnt be able to take off if already in the air" do
      expect { subject.takeoff }.to raise_error "Plane is already in the air"
    end

    it "should be in the air after taking off" do
      airport = double(:airport, plane_expected?: true, on_runway?: true)
      subject.land(airport)
      subject.takeoff
      expect(subject.status).to eq "In the air"
    end

    it "should check the plane is on the runway before letting it take off" do
      airport = double(:airport, on_runway?: true, plane_expected?: true)
      subject.land(airport)
      expect(airport).to receive(:on_runway?).with(subject)
      subject.takeoff
    end
  end
end
```

Tree: 09eaac1204 ▼

airport_challenge / spec / weather_spec.rb

Find file

Copy path

Fetching contributors...



Cannot retrieve contributors at this time

13 lines (11 sloc) | 277 Bytes

```
1  require 'weather'
2
3  describe Weather do
4    it "returns true when 1" do
5      allow(subject).to receive(:rand) {1}
6      expect(subject.stormy?).to eq true
7    end
8    it "returns false when 2" do
9      allow(subject).to receive(:rand) {2}
10     expect(subject.stormy?).to eq false
11   end
12 end
```