

Tree: 4d7fc759d2 ▼

airport_challenge / featuretests

Find file

Copy path

Fetching contributors...

 Cannot retrieve contributors at this time

0 lines (0 sloc) | 0 Bytes

Tree: 4d7fc759d2 ▾

airport_challenge / featuretests.rb

Find file

Copy path

Fetching contributors...

 Cannot retrieve contributors at this time

44 lines (33 sloc) | 1.09 KB

```
1  require "./lib/airport"
2  require "./lib/plane"
3  require "./lib/weather"
4  # "allows a plane to land"
5
6  airport = Airport.new
7  plane = Plane.new
8  weather = Weather.new
9  airport.land(plane, weather)
10 p airport.planes_landed
11
12 # allows a plane to take off and then removes it from the airport
13
14 airport.take_off(plane, weather)
15 p airport.planes_landed
16
17 # prevents takeoff when weather is stormy
18
19 # airport.land(plane, weather)
20 # weather.stormy = true
21 # airport.take_off(plane, weather)
22 # p airport.planes_landed
23
24 # prevents landing when weather is stormy
25
26 # weather.stormy = false
27 # airport.take_off(plane, weather)
28 # weather.stormy = true
29 # airport.land(plane, weather)
30 # p airport.planes_landed
31
32 # prevent planes from landing when the airport is full
33
34 # airport1 = Airport.new
35 # Airport::DEFAULT_CAPACITY.times {airport1.land(Plane.new, weather)}
36 # airport1.land(plane, weather)
37 # p airport1.planes_landed.count
38
39 # allows a default airport capacity that can be overridden as appropriate
40 # airport1 = Airport.new(30)
41 # 30.times {airport1.land(Plane.new, weather)}
42 # airport1.land(plane, weather)
43 # p airport1.planes_landed.count
```

Tree: 4d7fc759d2 ▾

airport_challenge / lib / **airport.rb**

Find file

Copy path

Fetching contributors...

 Cannot retrieve contributors at this time

32 lines (28 sloc) | 664 Bytes

```
1  require_relative 'weather'
2  require_relative 'plane'
3
4  class Airport
5    DEFAULT_CAPACITY = 20
6
7    def initialize(capacity = DEFAULT_CAPACITY)
8      @capacity = capacity
9      @planes_landed = []
10   end
11   def land plane, weather
12     raise "weather is too stormy for the plane to land" if weather.stormy?
13     raise "cannot land plane due to full airport" if full?
14     @planes_landed << plane
15   end
16   def take_off plane, weather
17     raise "weather is too stormy for take off" if weather.stormy?
18     @planes_landed.pop
19   end
20   def planes_landed
21     @planes_landed
22   end
23   def capacity
24     @capacity
25   end
26
27   private
28   def full?
29     @planes_landed.count == @capacity
30   end
31 end
```

Tree: 4d7fc759d2 ▾

airport_challenge / lib / plane.rb

Find file

Copy path

Fetching contributors...

 Cannot retrieve contributors at this time

3 lines (2 sloc) | 16 Bytes

```
1 class Plane
2 end
```

Tree: 4d7fc759d2 ▾

airport_challenge / lib / weather.rb

Find file

Copy path

Fetching contributors...

 Cannot retrieve contributors at this time

15 lines (14 sloc) | 222 Bytes


```
1  class Weather
2    def initialize
3      @stormy = rng_weather.sample
4    end
5    def stormy?
6      @stormy
7    end
8    def stormy= stormy
9      @stormy = stormy
10   end
11   def rng_weather
12     rng_weather = [true, true, true, false]
13   end
14 end
```

Tree: 4d7fc759d2 ▾

airport_challenge / spec / airport_spec.rb

Find fileCopy path

Fetching contributors...

 Cannot retrieve contributors at this time

61 lines (52 sloc) | 2.02 KB

```
1  require "./lib/airport"
2
3  describe Airport do
4    # double(:weather, stormy?: false)
5    it "allows a plane to land" do
6      airport = Airport.new
7      plane = double(:plane)
8      weather = double(:weather)
9      allow(weather).to receive(:stormy?).and_return(false)
10     airport.land(plane, weather)
11     expect(airport.planes_landed.count).to eq 1
12   end
13
14   it "allows a plane to take off and then removes it from the airport" do
15     airport = Airport.new
16     plane = double(:plane)
17     weather = double(:weather)
18     allow(weather).to receive(:stormy?).and_return(false)
19     airport.land(plane, weather)
20     airport.take_off(plane, weather)
21     expect(airport.planes_landed.count).to eq 0
22   end
23
24   it "prevents a plane from taking off if the weather is stormy" do
25     airport = Airport.new
26     plane = double(:plane)
27     weather = double(:weather)
28     allow(weather).to receive(:stormy?).and_return(false)
29     airport.land(plane, weather)
30     allow(weather).to receive(:stormy?).and_return(true)
31     expect { airport.take_off(plane, weather) }.to raise_error "weather is too stormy for take off"
32   end
33
34   it "prevents a plane from landing when weather is stormy" do
35     airport = Airport.new
36     plane = double(:plane)
37     weather = double(:weather)
38     allow(weather).to receive(:stormy?).and_return(true)
39     expect { airport.land(plane, weather) }.to raise_error "weather is too stormy for the plane to land"
40   end
41
42   it "prevent planes from landing when the airport is full" do
43     airport = Airport.new
44     plane1 = double(:plane)
45     weather = double(:weather, stormy?: false)
46     Airport::DEFAULT_CAPACITY.times { airport.land(double(:plane), weather) }
47     expect { airport.land(plane1, weather) }.to raise_error "cannot land plane due to full airport"
48   end
49
50   it "allows a default airport capacity that can be overridden as appropriate" do
51     airport1 = Airport.new
52     airport2 = Airport.new(30)
53     expect { airport1 }.not_to raise_error
54     expect { airport2 }.not_to raise_error
55     expect(airport1.capacity).to eq 20
56     expect(airport2.capacity).to eq 30
57   end
58
59
60 end
```