makersacademy / **airport_challenge**

Tree: 047c3cccaa ▾ **airport_challenge** / **User_stories.txt**

Find file Copy path

Fetching contributors…

Cannot retrieve contributors at this time

42 lines (28 sloc) | 1.07 KB

```
As an air traffic controller
So I can get passengers to a destination
I want to instruct a plane to land at an airport

1. plane <--(land)--> airport

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

2. plane <--(take_off)--> airport

As an air traffic controller
To ensure safety
I want to prevent takeoff when weather is stormy

3. plane --(take_off)--(stormy?)--> can't take off

As an air traffic controller
To ensure safety
I want to prevent landing when weather is stormy

4. plane --(land)--(stormy? = true)--> can't land

As an air traffic controller
To ensure safety
I want to prevent landing when the airport is full

5. plane --(land)--(capacity? = full)--> can't land

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

6. capacity = DEFAULT_CAPACITY = 15

Classes
Plane/Airport/Weather

methods
landing/taking off/stormy?/full?
```

makersacademy / **airport_challenge**

Tree: 047c3cccaa ▾ **airport_challenge** / lib / **airport.rb** Find file Copy path

Fetching contributors…

Cannot retrieve contributors at this time

41 lines (33 sloc) | 878 Bytes

```ruby
require_relative "plane"
require_relative "weather"

class Airport
  DEFAULT_CAPACITY = 15
  attr_reader :planes
  attr_accessor :capacity, :weather
    # hashes rather than = as have more than one argument!!!!!
  def initialize(capacity: DEFAULT_CAPACITY, weather: Weather.new)
    @planes = []
    @weather = weather
    @capacity = DEFAULT_CAPACITY
  end

  def land(a_plane)
    fail "Bad weather, too dangerous to land" if stormy?
    fail "Airport full" if full?
    @planes.push(a_plane) # undefined method << and .push???
    "The plane has landed"
  end

  def take_off(a_plane)
    fail "No planes" if @planes.empty?
    fail "Bad weather, all planes grounded" if stormy?
    @planes.delete(a_plane)
    "The plane has taken off"
  end

private

  def stormy?
    @weather.the_weather == :stormy
  end

  def full?
    @planes.length == @capacity
  end
end

# done!!!
```

makersacademy / **airport_challenge**

Tree: 047c3cccaa ▾ airport_challenge / lib / **plane.rb**

Find file Copy path

Fetching contributors…

Cannot retrieve contributors at this time

8 lines (4 sloc) 56 Bytes

```ruby
require_relative "airport"

class Plane

end

# done!!!
```

makersacademy / **airport_challenge**

Tree: 047c3cccaa ▾ **airport_challenge** / lib / **weather.rb** **Find file** **Copy path**

Fetching contributors…

Cannot retrieve contributors at this time

15 lines (10 sloc) | 157 Bytes

```ruby
class Weather

  def the_weather
    stormy? ? :stormy : :sunny
  end

private

  def stormy?
    rand(6) == 1  # 1/6 chance of a storm
  end
end

# done!!!
```

makersacademy / **airport_challenge**

Tree: 047c3cccaa ▾ **airport_challenge** / spec / **airport_spec.rb**

Find file Copy path

Fetching contributors…

Cannot retrieve contributors at this time

57 lines (45 sloc) | 1.71 KB

```ruby
require "airport"
require "weather"
require "plane"

describe Airport do
  # This allows all tests to have "sunny" weather.
  let :a_plane {double(:a_plane, land: nil, take_off: nil)}
  let :a_weather {double(:a_weather, the_weather: :Sunny)}
  subject {Airport.new(weather: a_weather)}

  describe "#land" do

    it "wont let a plane land due to bad weather" do
      allow(subject).to receive(:stormy?) { true } # same as .and_return (true)
      expect { subject.land(a_plane) }.to raise_error("Bad weather, too dangerous to land")
    end

    it "wont let a plane land as airport is full(default)" do
      Airport::DEFAULT_CAPACITY.times { subject.land(Plane.new) }
      expect { subject.land(Plane.new) }.to raise_error("Airport full")
    end

    it "can instruct a plane to land" do
      expect(subject.land(a_plane)).to eq "The plane has landed"
    end

    it "changes the capacity of the airport to 5" do
      subject.capacity = 5
      expect(subject.capacity).to eq 5
    end

    it "wont let a plane land due to overide capacity" do
      subject.capacity = 20
      subject.capacity.times { subject.land(Plane.new) }
      expect { subject.land(Plane.new) }.to raise_error("Airport full")
    end
  end

  describe "#take_off" do

    it "can allow a plane take off" do
      subject.land(a_plane)
      expect(subject.take_off(a_plane)).to eq("The plane has taken off")
    end

    it "can confirm a plane has taken off" do
      expect(subject.planes).to be_empty
    end

    it "wont allow a plane to take off due to bad weather" do
      subject.land(a_plane)
      allow(subject).to receive(:stormy?) { true }
      expect { subject.take_off(a_plane) }.to raise_error("Bad weather, all planes grounded")
    end
  end
end
```

makersacademy / **airport_challenge**

Tree: 047c3cccaa ▾ **airport_challenge** / spec / **feature_spec.rb**

Find file Copy path

Fetching contributors…

Cannot retrieve contributors at this time

14 lines (13 sloc) | 354 Bytes

```
# require "plane"
# require "airport"
# require "weather"
# a_plane = Plane.new
# a_airport = Airport.new
# a_airport.land(a_plane)
# a_airport.take_off(a_plane)
# a_airport.take_off(a_plane)
# 15.times{ a_airport.land(Plane.new) }
# 10.times{ a_airport.take_off(plane) }
# a_airport.capacity = 10
# 5.times{ a_aiport.land(Plane.new) }
# a_airport.full?
```

Tree: **047c3cccaa** ▾ **airport_challenge** / spec / **plane_spec.rb** Find file Copy path

Fetching contributors…

Cannot retrieve contributors at this time

```
8 lines (4 sloc) | 51 Bytes
```

```ruby
require "plane"

describe Plane do

end

# done!!!
```

makersacademy / **airport_challenge**

Tree: 047c3cccaa ▾ **airport_challenge** / spec / **weather_spec.rb** Find file Copy path

Fetching contributors…

Cannot retrieve contributors at this time

20 lines (14 sloc) | 380 Bytes

```ruby
require "weather"

describe Weather do

  describe "#the_weather" do

    it "When equal to 1, stormy weather" do
      allow(subject).to receive(:rand) { 1 }
      expect(subject.the_weather).to eq :stormy
    end

    it "When equal to 2, sunny weather" do
      allow(subject).to receive(:rand) { 2 }
      expect(subject.the_weather).to eq :sunny
    end
  end
end

# done!!!
```