makersacademy / airport_challenge

Tree: 4d7fc759d2

airport_challenge / featuretests

Find file Copy path

Fetching contributors…

Cannot retrieve contributors at this time

```
0 lines (0 sloc) | 0 Bytes
```

makersacademy / **airport_challenge**

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

```ruby
require "./lib/airport"
require "./lib/plane"
require "./lib/weather"
# "allows a plane to land"

airport = Airport.new
plane = Plane.new
weather = Weather.new
airport.land(plane, weather)
p airport.planes_landed

# allows a plane to take off and then removes it from the airport

airport.take_off(plane, weather)
p airport.planes_landed

# prevents takeoff when weather is stormy

# airport.land(plane, weather)
# weather.stormy = true
# airport.take_off(plane, weather)
# p airport.planes_landed

# prevents landing when weather is stormy

# weather.stormy = false
# airport.take_off(plane, weather)
# weather.stormy = true
# airport.land(plane, weather)
# p airport.planes_landed

# prevent planes from landing when the airport is full

# airport1 = Airport.new
# Airport::DEFAULT_CAPACITY.times {airport1.land(Plane.new, weather)}
# airport1.land(plane, weather)
# p airport1.planes_landed.count

# allows a default airport capacity that can be overridden as appropriate
# airport1 = Airport.new(30)
# 30.times {airport1.land(Plane.new, weather)}
# airport1.land(plane, weather)
# p airport1.planes_landed.count
```

makersacademy / **airport_challenge**

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

```ruby
require_relative 'weather'
require_relative 'plane'

class Airport
  DEFAULT_CAPACITY = 20

  def initialize(capacity = DEFAULT_CAPACITY)
    @capacity = capacity
    @planes_landed = []
  end
  def land plane, weather
    raise "weather is too stormy for the plane to land" if weather.stormy?
    raise "cannot land plane due to full airport" if full?
    @planes_landed << plane
  end
  def take_off plane, weather
    raise "weather is too stormy for take off" if weather.stormy?
    @planes_landed.pop
  end
  def planes_landed
    @planes_landed
  end
  def capacity
    @capacity
  end

  private
  def full?
     @planes_landed.count == @capacity
  end
end
```

makersacademy / **airport_challenge**

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

```ruby
class Plane
end
```

makersacademy / **airport_challenge**

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

```ruby
class Weather
  def initialize
    @stormy = rng_weather.sample
  end
  def stormy?
    @stormy
  end
  def stormy= stormy
    @stormy = stormy
  end
  def rng_weather
    rng_weather = [true, true, true, false]
  end
end
```

makersacademy / **airport_challenge**

Tree: 4d7fc759d2 ▾ **airport_challenge** / spec / **airport_spec.rb** Find file Copy path

Fetching contributors…

Cannot retrieve contributors at this time

61 lines (52 sloc) | 2.02 KB

```ruby
require "./lib/airport"

describe Airport do
  # double(:weather, stormy?: false)
  it "allows a plane to land" do
    airport = Airport.new
    plane = double(:plane)
    weather = double(:weather)
    allow(weather).to receive(:stormy?).and_return(false)
    airport.land(plane, weather)
    expect(airport.planes_landed.count).to eq 1
  end

  it "allows a plane to take off and then removes it from the airport" do
    airport = Airport.new
    plane = double(:plane)
    weather = double(:weather)
    allow(weather).to receive(:stormy?).and_return(false)
    airport.land(plane, weather)
    airport.take_off(plane, weather)
    expect(airport.planes_landed.count).to eq 0
  end

  it "prevents a plane from taking off if the weather is stormy" do
    airport = Airport.new
    plane = double(:plane)
    weather = double(:weather)
    allow(weather).to receive(:stormy?).and_return(false)
    airport.land(plane, weather)
    allow(weather).to receive(:stormy?).and_return(true)
    expect { airport.take_off(plane, weather) }.to raise_error "weather is too stormy for take off"
  end

  it "prevents a plane from landing when weather is stormy" do
    airport = Airport.new
    plane = double(:plane)
    weather = double(:weather)
    allow(weather).to receive(:stormy?).and_return(true)
    expect {airport.land(plane, weather) }.to raise_error "weather is too stormy for the plane to land"
  end

  it "prevent planes from landing when the airport is full" do
    airport = Airport.new
    plane1 = double(:plane)
    weather = double(:weather, stormy?: false)
    Airport::DEFAULT_CAPACITY.times { airport.land(double(:plane), weather) }
    expect { airport.land(plane1, weather) }.to raise_error "cannot land plane due to full airport"
  end

  it "allows a default airport capacity that can be overridden as appropriate" do
    airport1 = Airport.new
    airport2 = Airport.new(30)
    expect { airport1 }.not_to raise_error
    expect { airport2 }.not_to raise_error
    expect(airport1.capacity).to eq 20
    expect(airport2.capacity).to eq 30
  end


end
```