

Tree: 25424c6706 ▾

[airport\\_challenge](#) / [00\\_notes](#) / **Commits** **Descriptions.txt**

Find file

Copy path


Fetching contributors...

 Cannot retrieve contributors at this time

12 lines (10 sloc) | 379 Bytes

1	COMMIT & DESCRIPTION
2	
3	Updated README.md
4	Updated README.md part 2
5	Updated README.md part 3
6	Required folders, files and spec files created
7	Tests for airport responding to methods :dock and :launch
8	Tests for docking and launching a plane as well as plane count in airport
9	Bad Weather module added
10	Test for plane status and update dependancies on all files
11	A fully working airport

Fetching contributors...

 Cannot retrieve contributors at this time

132 lines (126 sloc) | 5.97 KB

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <ScappleDocument Version="1.1" ID="EE96C191-8745-46BE-8CA0-7998C8A41C72">
3      <Notes>
4          <Note ID="0" FontSize="12.0" Position="31.203125,55.0" Width="520.0">
5              <Appearance>
6                  <Alignment>Left</Alignment>
7                  <Border Weight="2"></Border>
8              </Appearance>
9              <String>As an air traffic controller
10 So I can get passengers to a destination
11 I want to instruct a plane to land at an airport
12
13 As an air traffic controller
14 So I can get passengers on the way to their destination
15 I want to instruct a plane to take off from an airport and confirm that it is no longer in the airport
16
17 As an air traffic controller
18 To ensure safety
19 I want to prevent takeoff when weather is stormy
20
21 As an air traffic controller
22 To ensure safety
23 I want to prevent landing when weather is stormy
24
25 As an air traffic controller
26 To ensure safety
27 I want to prevent landing when the airport is full
28
29 As the system designer
30 So that the software can be used for many different airports
31 I would like a default airport capacity that can be overridden as appropriate</String>
32 </Note>
33 <Note ID="1" FontSize="21.0" Position="229.0,14.0" Width="124.0">
34     <Appearance>
35         <Alignment>Center</Alignment>
36     </Appearance>
37     <String>User Stories</String>
38 </Note>
39 <Note ID="2" FontSize="21.0" Position="582.285156,14.136719" Width="87.0">
40     <Appearance>
41         <Alignment>Left</Alignment>
42     </Appearance>
43     <String>Analysis</String>
44 </Note>
45 <Note ID="3" FontSize="12.0" Position="574.28125,55.0" Width="115.0">
46     <Appearance>
47         <Alignment>Left</Alignment>
48         <Border Weight="2"></Border>
49     </Appearance>
50     <String>Nouns
51 Air Traffic Controller
52 Plane
53 Airport
54 Systems Designer
55 Weather
56
57 Verbs
58 land
59 take-off
60 confirm
61 prevent</String>
62     <Formatting>
63         <FormatRange Bold="Yes">0,5</FormatRange>
64         <FormatRange Bold="Yes">69,6</FormatRange>
65     </Formatting>
66 </Note>
67 <Note ID="5" FontSize="21.0" Position="34.0,416.570312" Width="291.0">
68     <Appearance>
69         <Alignment>Left</Alignment>
70     </Appearance>
71     <String>Possible Actions and Scenaria</String>
72 </Note>
73 <Note ID="6" FontSize="12.0" Position="33.664062,455.804688" Width="654.0">
74     <Appearance>
75         <Alignment>Left</Alignment>
76         <Border Weight="2"></Border>
77     </Appearance>
78     <String>
79         ◦ Plane status (flying/docked)
80         ◦ Plane landing (flying -&gt; docked)
81         ◦ Plane takeoff (docked - &gt; flying)
82         ◦ Storms prevent landing /take-off
83         ◦ Full airport cannot accept planes
84         ◦ Variable and default capacity
85         ◦ Errors raised for inconsistent actions (take-off of a flying plane / landing a docked plane)</String>
86 </Note>
87 </Notes>
88 <BackgroundShapes></BackgroundShapes>
89 <Stacks>
90     <Stack>5-6</Stack>
91 </Stacks>
92 <AutoFit>1-3, 5</AutoFit>
93 <NoteStyles>
94     <Style Name="Bubble (Blue)" ID="44F93AEF-237D-4000-AB8A-FBE9296755C9" AffectFontStyle="No" AffectAlignment="No" AffectT
95         <BorderThickness>1</BorderThickness>
96         <BorderColor>0.561855 0.71062 0.779401</BorderColor>
97         <FillColor>0.702319 0.888276 0.974252</FillColor>
98     </Style>
99     <Style Name="Bubble (Green)" ID="F7591117-C3A3-4B43-A13A-FDCEC1AA2ABC" AffectFontStyle="No" AffectAlignment="No" Affect
100         <BorderThickness>1</BorderThickness>
101         <BorderColor>0.572684 0.758969 0.558154</BorderColor>
102         <FillColor>0.715855 0.948712 0.697692</FillColor>
103     </Style>
104     <Style Name="Bubble (Pink)" ID="CB77CF33-09CF-4543-8C0C-957363DEC6C2" AffectFontStyle="No" AffectAlignment="No" AffectT
105         <BorderThickness>1</BorderThickness>
106         <BorderColor>0.794796 0.560965 0.58607</BorderColor>
107         <FillColor>0.957566 0.766747 0.999616</FillColor>
108     </Style>
109     <Style Name="Bubble (Red)" ID="A1BD25EB-3648-4E91-BB73-59AA96607070" AffectFontStyle="No" AffectAlignment="No" AffectTe
110         <BorderThickness>1</BorderThickness>
111         <BorderColor>0.794796 0.560965 0.58607</BorderColor>
112         <FillColor>0.993495 0.701207 0.732587</FillColor>
113     </Style>
114     <Style Name="Bubble (Yellow)" ID="0D779C4F-6D59-4733-A906-03C6C010F60A" AffectFontStyle="No" AffectAlignment="No" Affec
115         <BorderThickness>1</BorderThickness>
116         <BorderColor>0.798177 0.714184 0.522055</BorderColor>
117         <FillColor>0.997722 0.89273 0.652569</FillColor>
118     </Style>
119     <Style Name="Red Text" ID="27F899D2-39BF-4E48-B531-9946E600C5B9" AffectFontStyle="No" AffectAlignment="No" AffectTextCo
120         <TextColor>0.985948 0.0 0.026951</TextColor>
121     </Style>
122     <Style Name="Title Text" ID="519FB925-7E86-4DBA-B586-E9F81509156B" AffectFontStyle="Yes" AffectAlignment="Yes" AffectTe
123         <FontSize>28.0</FontSize>
124         <IsBold>Yes</IsBold>
125     </Style>
126 </NoteStyles>
127 <UISettings>
128     <BackgroundColor>0.999767 0.98837 0.949907</BackgroundColor>
129     <DefaultFont>Helvetica</DefaultFont>
130     <DefaultTextColor>0.0 0.0 0.0</DefaultTextColor>
131 </UISettings>
132 <PrintSettings PaperSize="595.0,842.0" LeftMargin="72.0" RightMargin="72.0" TopMargin="90.0" BottomMargin="90.0" PaperType=
```

Tree: 25424c6706 ▾

airport\_challenge / lib / airport.rb

Find file

Copy path

Fetching contributors...

 Cannot retrieve contributors at this time

49 lines (37 sloc) | 885 Bytes

```
1  require "plane"
2  require "hand_of_god"
3
4  class Airport
5
6      # Constants
7      CAPACITY = 5
8
9      # Attribute Readers/Writers
10     # We use an attr_accessor to accomodate for the ability
11     # to expand in the future
12     attr_accessor :airp_capacity
13     attr_accessor :planes
14
15     # Requirments / Modules
16     include Weather
17
18     # Methods
19     def initialize(airp_capacity = CAPACITY)
20         @airp_capacity = airp_capacity
21         @planes = []
22     end
23
24     def dock(plane)
25         raise "Weather Stormy!! Planes cannot land/dock" if stormy?
26         raise "Airport is full, cannot land" if full?
27         @planes << plane
28     end
29
30     def launch(plane)
31         raise "Weather Stormy!! Planes are Grounded" if stormy?
32         raise "There are no planes in the airport" if empty?
33         @planes.delete(plane)
34     end
35
36     def count_of_planes
37         @planes.count
38     end
39
40     def full?
41         @planes.count >= airp_capacity
42     end
43
44     def empty?
45         @planes.count <= 0
46     end
47
48     end
```

Tree: 25424c6706 ▾

airport\_challenge / lib / hand\_of\_god.rb

Find file

Copy path

Fetching contributors...



Cannot retrieve contributors at this time

10 lines (6 sloc) | 124 Bytes

```
1  module Weather
2
3      CHANCES_OF_GOOD_WEATHER = 50
4
5      def stormy?
6          rand(1..101) > CHANCES_OF_GOOD_WEATHER
7      end
8
9  end
```

Tree: 25424c6706 ▾

airport\_challenge / lib / plane.rb

Find file

Copy path

Fetching contributors...

🔄 Cannot retrieve contributors at this time

20 lines (14 sloc) | 293 Bytes

```
1  class Plane
2  attr_accessor :flying
3
4      def initialize(flying = false)
5          @flying = flying
6      end
7
8      def taking_off
9          raise "Plane is already in the air" if flying
10         @flying = true
11     end
12
13     def landing
14         raise "Plane is already docked" if !flying
15         @flying = false
16     end
17
18
19 end
```

Tree: 25424c6706 ▾

airport\_challenge / spec / airport\_spec.rb

Find file

Copy path

Fetching contributors...

 Cannot retrieve contributors at this time

91 lines (76 sloc) | 3.27 KB

```
1  require "airport"
2  require "plane"
3  require "hand_of_god"
4
5  describe Airport do
6    # Name the subject explicitly
7    subject(:airport) {described_class.new}
8
9    # Define Doubles
10
11   # Test for initializing an airport
12   context "Airport initialization can work both with arguments and without" do
13     it "accepts a capacity argument, if not capacity defaults to DEFAULT_CAPACITY" do
14       expect{Airport.new(30)}.not_to raise_error
15       expect{Airport.new}.not_to raise_error
16       expect(subject.airp_capacity).to eq Airport::CAPACITY
17     end
18   end
19
20   # Tests for Airport responding to methods
21   context "Airport responds to various methods" do
22     it "plane can land" do
23       expect(subject).to respond_to (:dock)
24     end
25
26     it "plane can take off" do
27       expect(subject).to respond_to (:launch)
28     end
29
30     it "airport logs how many planes are docked" do
31       expect(subject).to respond_to (:count_of_planes)
32     end
33   end
34
35   # Test for docking and launching a plane and checking
36   # how many planes are in the airport at any given time
37   context "Airport is docking & launching planes" do
38     # Set stormy? to be false for tests of just docking & launching planes
39     before (:each) do allow(airport).to receive(:stormy?).and_return(false) end
40     it " - a plane lands" do
41       # airport = Airport.new --- Omitted as we have used subject(:airport) {described_class.new}
42       plane = Plane.new
43       airport.dock(plane)
44       expect(airport.count_of_planes).to eq 1
45     end
46
47     it " - a plane takes off" do
48       # airport = Airport.new --- Omitted as we have used subject(:airport) {described_class.new}
49       plane = Plane.new
50       airport.dock(plane)
51       airport.launch(plane)
52       expect(airport.count_of_planes).to eq 0
53     end
54   end
55
56   # Tests for preventing a plane launching if weather is stormy
57   context "Weather conditions are affecting docking & launching" do
58
59     it "stops a plane from launching if stormy" do
60       # We need to temporarily set stormy to false,so we can dock a plane
61       allow(airport).to receive(:stormy?).and_return(false)
62       plane = Plane.new
63       airport.dock(plane)
64       # We set stormy to true,so we can run our test
65       allow(airport).to receive(:stormy?).and_return(true)
66       expect{airport.launch(plane)}.to raise_error "Weather Stormy!! Planes are Grounded"
67     end
68
69     it "stops a plane from docking if stormy" do
70       allow(airport).to receive(:stormy?).and_return(true)
71       plane = Plane.new
72       expect {airport.dock(plane)}.to raise_error "Weather Stormy!! Planes cannot land/dock"
73     end
74   end
75
76   # Tests for preventing a plane docking if airport full
77   context "Traffic Control gives permission for landing pending on capacity" do
78     it "stops a plane for landing/docking if airport is full"do
79       allow(airport).to receive(:stormy?).and_return(false)
80       5.times{airport.dock(Plane.new)}
81       expect {airport.dock(Plane.new)}.to raise_error "Airport is full, cannot land"
82     end
83
84     it "a plane cannot launch if there are no planes in airport"do
85       allow(airport).to receive(:stormy?).and_return(false)
86       expect {airport.launch(Plane.new)}.to raise_error "There are no planes in the airport"
87     end
88   end
89
90 end
```




Tree 25424c6706 ▾

airport\_challenge / spec / bonus\_airport\_spec.rb

Find file

Copy path

Fetching contributors...

 Cannot retrieve contributors at this time

114 lines (93 sloc) | 4.07 KB

```
1  require "airport"
2  require "plane"
3  require "hand_of_god"
4
5  describe Airport do
6    # Name the subject explicitly
7      subject(:airport) {described_class.new}
8
9    # Define Doubles
10
11    # Test for initializing an airport
12    context "Airport initialization can work both with arguments and without" do
13      it "accepts a capacity argument, if not capacity defaults to DEFAULT_CAPACITY" do
14        expect{Airport.new(30)}.not_to raise_error
15        expect{Airport.new}.not_to raise_error
16        expect(subject.airp_capacity).to eq Airport::CAPACITY
17      end
18    end
19
20    # Tests for Airport responding to methods
21    context "Airport responds to various methods" do
22      it "plane can land" do
23        expect(subject).to respond_to (:dock)
24      end
25
26      it "plane can take off" do
27        expect(subject).to respond_to (:launch)
28      end
29
30      it "airport logs how many planes are docked" do
31        expect(subject).to respond_to (:count_of_planes)
32      end
33    end
34
35    # Test for docking and launching a plane and checking
36    # how many planes are in the airport at any given time
37    context "Airport is docking & launching planes" do
38      # Set stormy? to be false for tests of just docking & launching planes
39      before (:each) do allow(airport).to receive(:stormy?).and_return(false) end
40      it " - a plane lands" do
41        # airport = Airport.new --- Omitted as we have used subject(:airport) {described_class.new}
42        plane = Plane.new
43        airport.dock(plane)
44        expect(airport.count_of_planes).to eq 1
45      end
46
47      it " - a plane takes off" do
48        # airport = Airport.new --- Omitted as we have used subject(:airport) {described_class.new}
49        plane = Plane.new
50        airport.dock(plane)
51        airport.launch(plane)
52        expect(airport.count_of_planes).to eq 0
53      end
54    end
55
56    # Tests for preventing a plane launching if weather is stormy
57    context "Weather conditions are affecting docking & launching" do
58
59      it "stops a plane from launching if stormy" do
60        # We need to temporarily set stormy to false,so we can dock a plane
61        allow(airport).to receive(:stormy?).and_return(false)
62        plane = Plane.new
63        airport.dock(plane)
64        # We set stormy to true,so we can run our test
65        allow(airport).to receive(:stormy?).and_return(true)
66        expect{airport.launch(plane)}.to raise_error "Weather Stormy!! Planes are Grounded"
67      end
68
69      it "stops a plane from docking if stormy" do
70        allow(airport).to receive(:stormy?).and_return(true)
71        plane = Plane.new
72        expect {airport.dock(plane)}.to raise_error "Weather Stormy!! Planes cannot land/dock"
73      end
74    end
75
76    # Tests for preventing a plane docking if airport full
77    context "Traffic Control gives permission for landing pending on capacity" do
78      it "stops a plane for landing/docking if airport is full"do
79        allow(airport).to receive(:stormy?).and_return(false)
80        5.times{airport.dock(Plane.new)}
81        expect {airport.dock(Plane.new)}.to raise_error "Airport is full, cannot land"
82      end
83
84      it "a plane cannot launch if there are no planes in airport"do
85        allow(airport).to receive(:stormy?).and_return(false)
86        expect {airport.launch(Plane.new)}.to raise_error "There are no planes in the airport"
87      end
88    end
89
90    # Test for landing many planes
91    context "Land a number of planes equal to default capacity (5) and then get them to take off" do
92      it "Dock all the planes prior and get them ready to take off" do
93        allow(airport).to receive(:stormy?).and_return(false)
94        airport.airp_capacity.times { airport.dock(Plane.new)}
95        expect(subject.count_of_planes).to eq airport.airp_capacity
96      end
97
98      it "All planes are taking off" do
99        allow(airport).to receive(:stormy?).and_return(false)
100        airport.airp_capacity.times { airport.dock(Plane.new)}
101        # We need to use the until as each time the each runs, it alters the array is it running on
102        until airport.planes.empty?
103          airport.planes.each{|plane| airport.launch(plane)}
104        end
105        expect(subject.count_of_planes).to eq 0
106
107      end
108
109    end
110
111  end
112
113 end
```

Fetching contributors...

 Cannot retrieve contributors at this time

39 lines (30 sloc) | 1.02 KB

```
1  require "plane"
2
3  describe Plane do
4    subject(:plane) {described_class.new}
5
6    context "When a plane is created it is not flying" do
7      it "is not flying" do
8        expect(subject.flying).to eq false
9      end
10   end
11
12   context "Changing flying attributes when landing/taking off" do
13     it "when docking, its flying attribute changes to false" do
14       plane.flying = true
15       plane.landing
16       expect(plane.flying).to eq false
17     end
18
19
20     it "when launching, its flying attribute changes to true" do
21       plane.flying
22       plane.taking_off
23       expect(plane.flying).to eq true
24     end
25   end
26
27   context "Plane in flight cannot be launched / Plane docked cannot be docked" do
28     it "when a plane is flying, it cannot be launched" do
29       plane.taking_off
30       expect{plane.taking_off}.to raise_error "Plane is already in the air"
31     end
32
33     it "when a plane is docked, it cannot be docked again" do
34       expect{plane.landing}.to raise_error "Plane is already docked"
35     end
36   end
37
38   end
```