CodeClan_
Bridging the digital skills gap.

Evidence Gathering Document for SQA Level 8 Professional Developer Award.

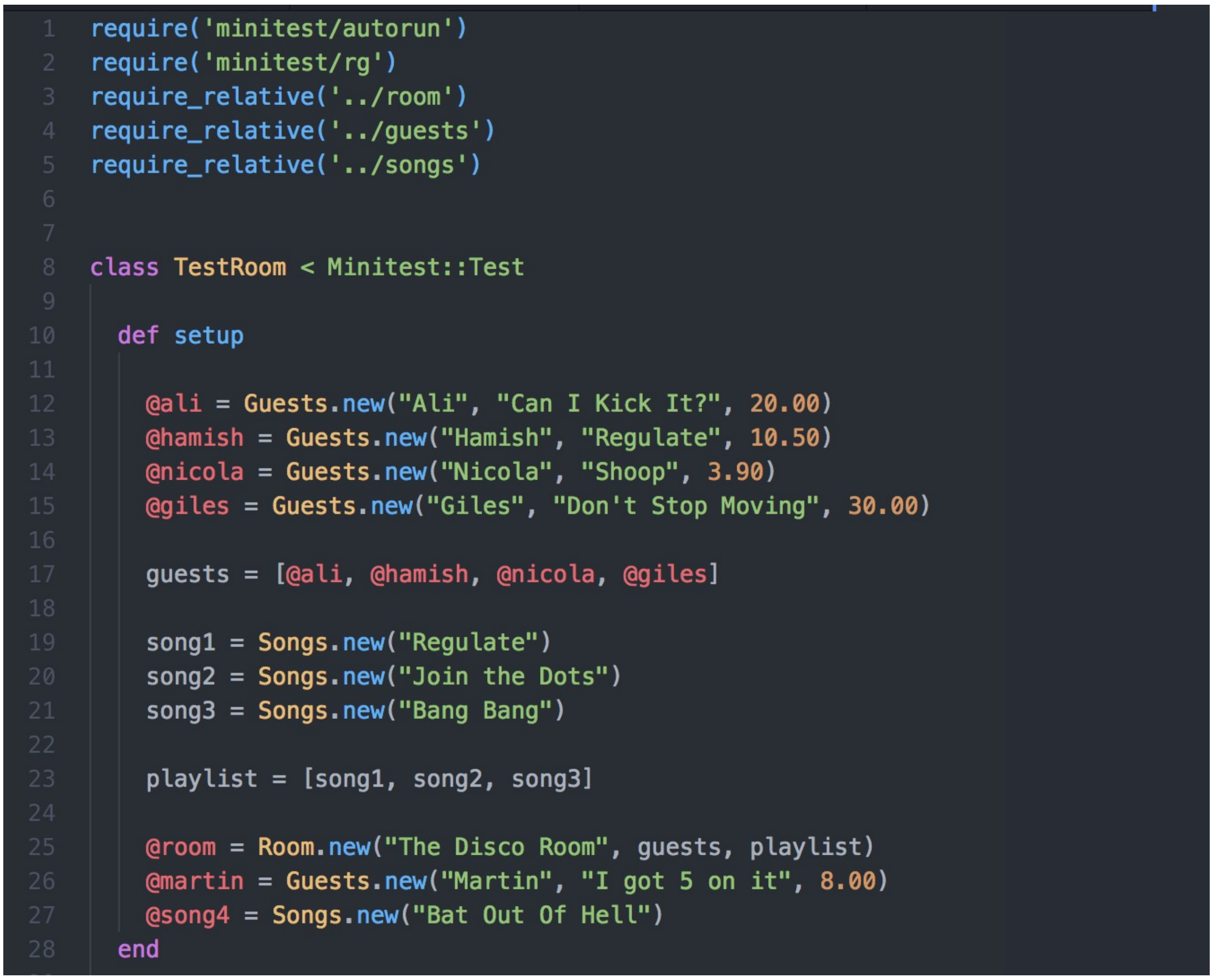
This document is designed for you to present your screenshots and diagrams relevant to the PDA and to also give a short description of what you are showing to clarify understanding for the assessor.

Please fill in each point with screenshot or diagram and description of what you are showing.

Each point requires details that cover each element of the Assessment Criteria, along with a brief description of the kind of things you should be showing.

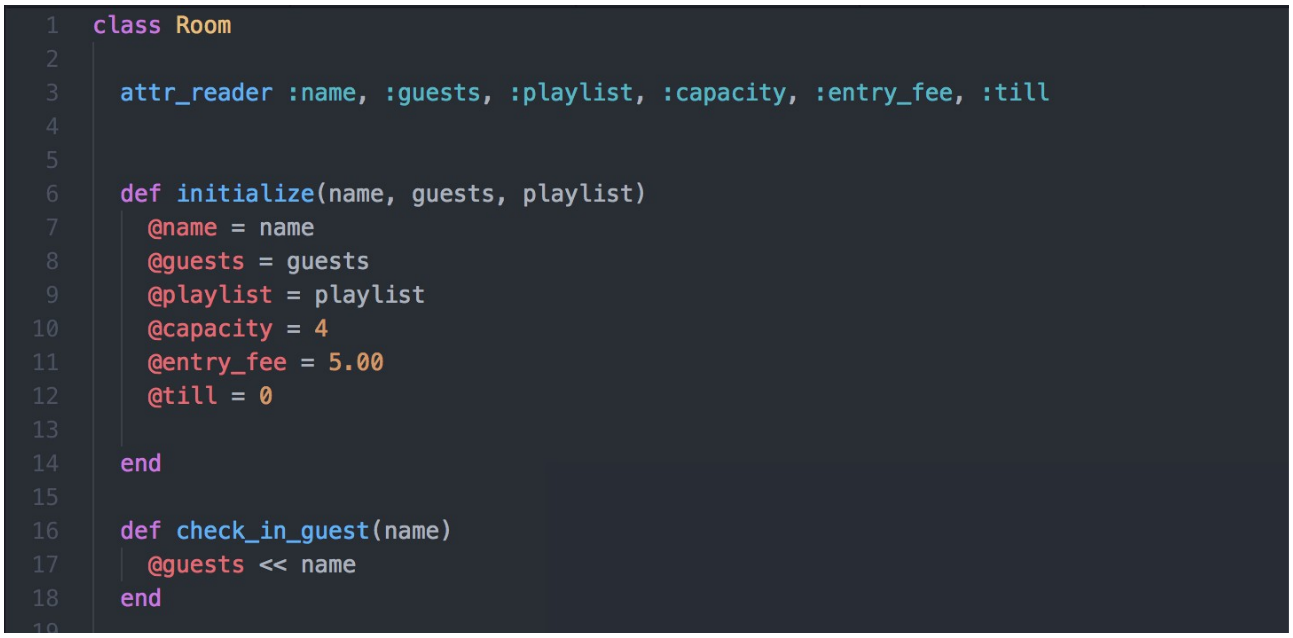
Week 2

Unit	Ref	Evidence	
I&T	I.T.5	Demonstrate the use of an array in a program. Take screenshots of: *An array in a program *A function that uses the array *The result of the function running	
		Description:	



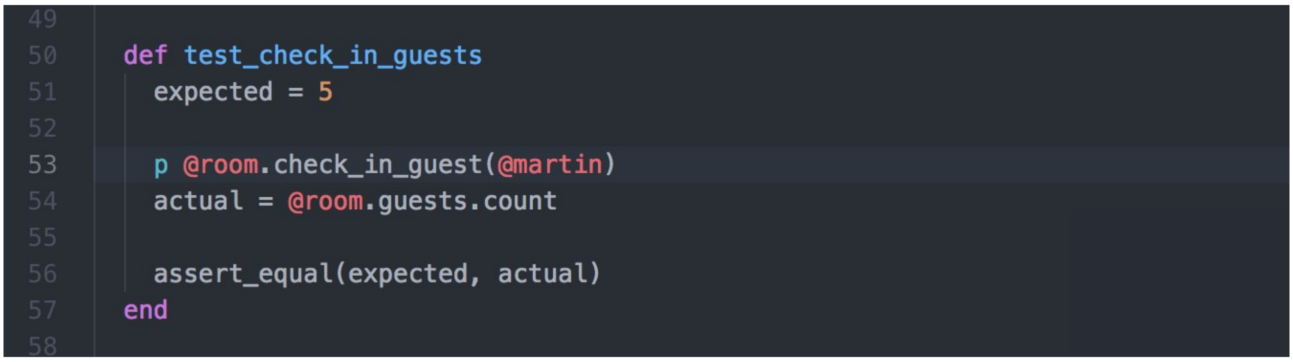
```ruby
require('minitest/autorun')
require('minitest/rg')
require_relative('../room')
require_relative('../guests')
require_relative('../songs')


class TestRoom < Minitest::Test

  def setup

    @ali = Guests.new("Ali", "Can I Kick It?", 20.00)
    @hamish = Guests.new("Hamish", "Regulate", 10.50)
    @nicola = Guests.new("Nicola", "Shoop", 3.90)
    @giles = Guests.new("Giles", "Don't Stop Moving", 30.00)

    guests = [@ali, @hamish, @nicola, @giles]

    song1 = Songs.new("Regulate")
    song2 = Songs.new("Join the Dots")
    song3 = Songs.new("Bang Bang")

    playlist = [song1, song2, song3]

    @room = Room.new("The Disco Room", guests, playlist)
    @martin = Guests.new("Martin", "I got 5 on it", 8.00)
    @song4 = Songs.new("Bat Out Of Hell")
  end
```

This screen shot of class TestRoom has examples of two arrays, we're going to be focusing on the array entitled “guests”. This array contains instances of the class Guests: @ali, @hamish, @nicola and @giles. Further down we can see @martin, another instance of the class Guests. We will be adding @martin to the guests array.



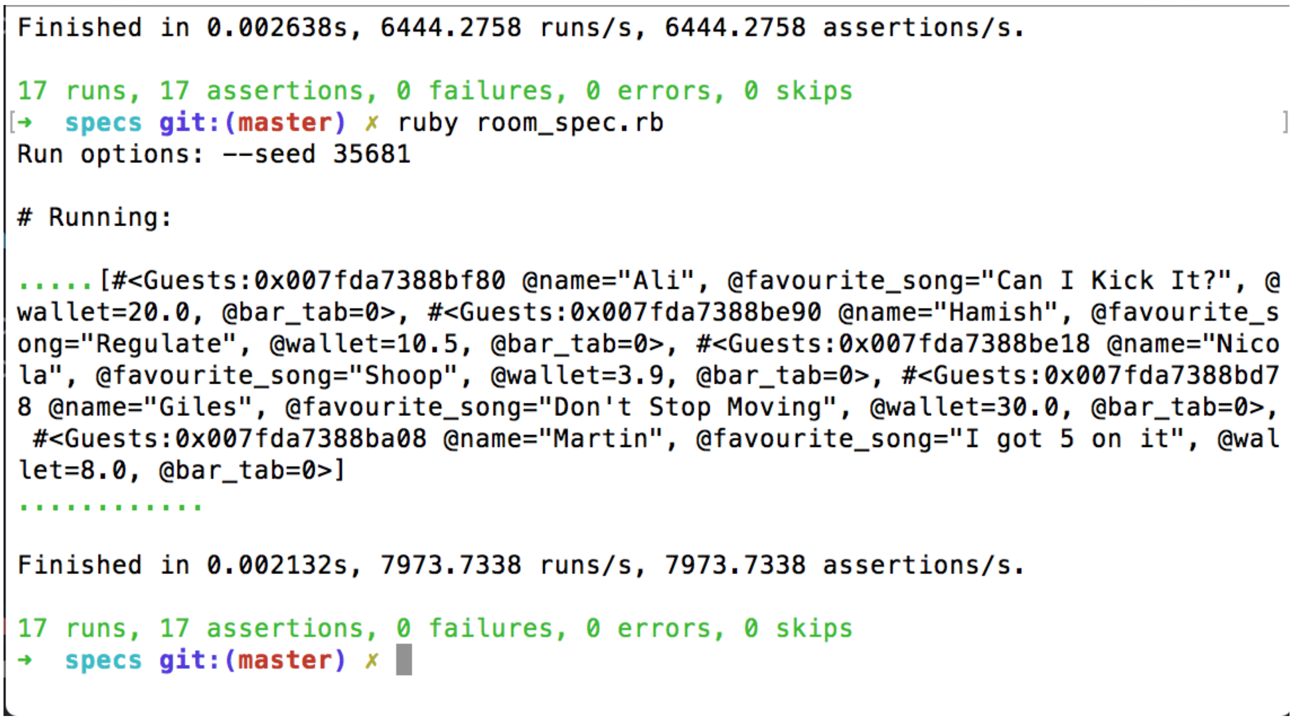
```ruby
class Room

  attr_reader :name, :guests, :playlist, :capacity, :entry_fee, :till


  def initialize(name, guests, playlist)
    @name = name
    @guests = guests
    @playlist = playlist
    @capacity = 4
    @entry_fee = 5.00
    @till = 0

  end

  def check_in_guest(name)
    @guests << name
  end
```

This second screen shot features a function entitled “check_in_guest”. check_in_guest takes one argument “name” and uses the method << to add the argument to the end of the array.



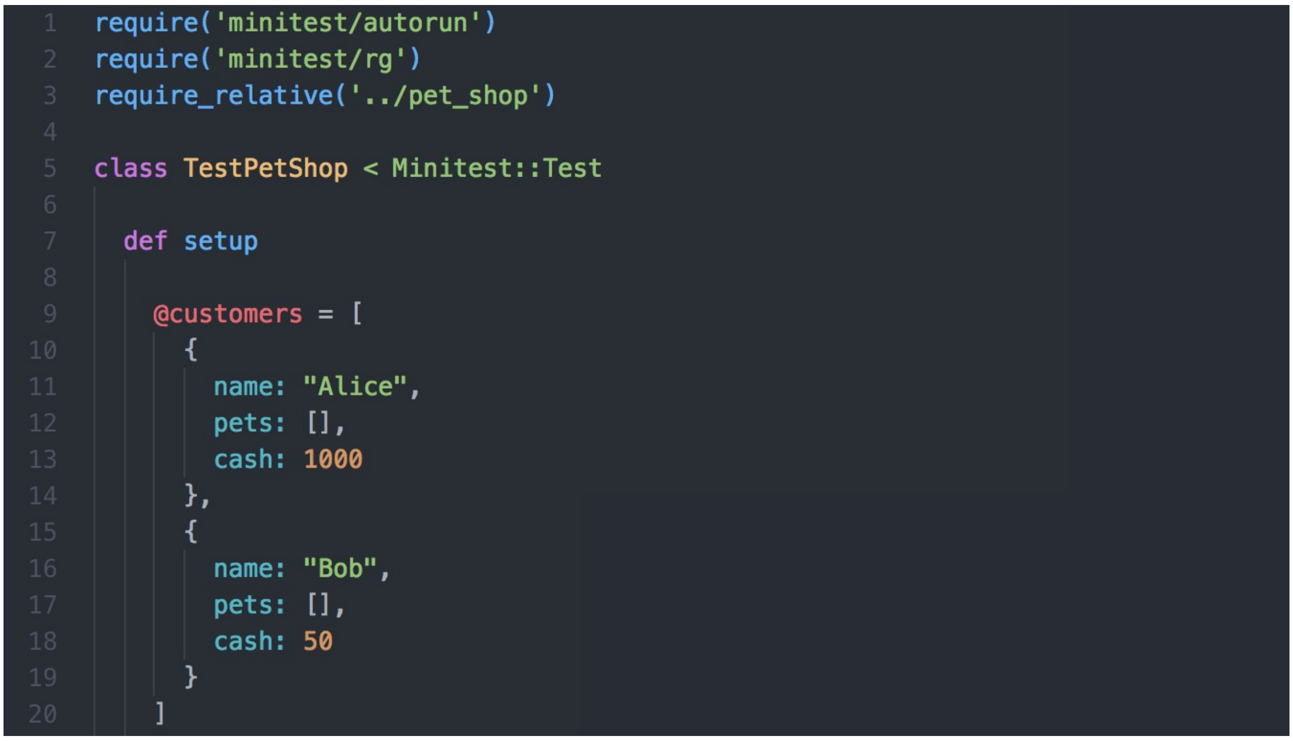
```ruby
def test_check_in_guests
  expected = 5

  p @room.check_in_guest(@martin)
  actual = @room.guests.count

  assert_equal(expected, actual)
end
```

Finally, we can see in this screen shot the result of running this function.



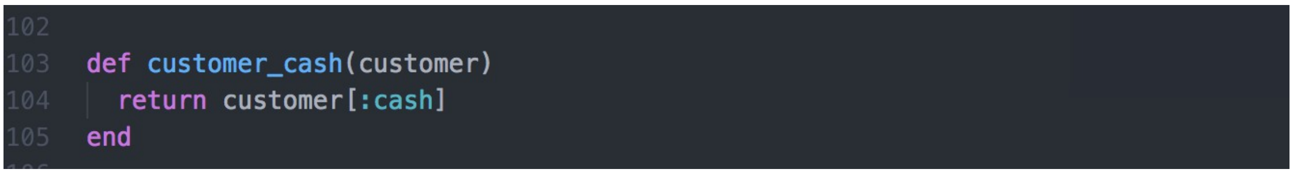
```
Finished in 0.002638s, 6444.2758 runs/s, 6444.2758 assertions/s.

17 runs, 17 assertions, 0 failures, 0 errors, 0 skips
➜  specs git:(master) ✗ ruby room_spec.rb
Run options: --seed 35681

# Running:

.....[#<Guests:0x007fda7388bf80 @name="Ali", @favourite_song="Can I Kick It?", @
wallet=20.0, @bar_tab=0>, #<Guests:0x007fda7388be90 @name="Hamish", @favourite_s
ong="Regulate", @wallet=10.5, @bar_tab=0>, #<Guests:0x007fda7388be18 @name="Nico
la", @favourite_song="Shoop", @wallet=3.9, @bar_tab=0>, #<Guests:0x007fda7388bd7
8 @name="Giles", @favourite_song="Don't Stop Moving", @wallet=30.0, @bar_tab=0>,
 #<Guests:0x007fda7388ba08 @name="Martin", @favourite_song="I got 5 on it", @wal
let=8.0, @bar_tab=0>]
............

Finished in 0.002132s, 7973.7338 runs/s, 7973.7338 assertions/s.

17 runs, 17 assertions, 0 failures, 0 errors, 0 skips
➜  specs git:(master) ✗ 
```

Unit	Ref	Evidence	
I&T	I.T.6	Demonstrate the use of a hash in a program. Take screenshots of: *A hash in a program *A function that uses the hash *The result of the function running	
		Description:	



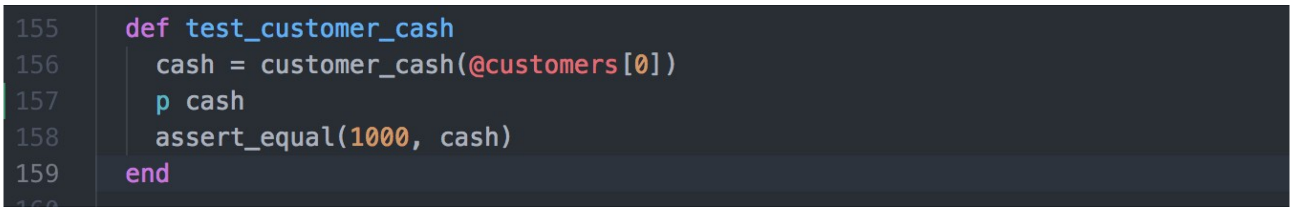
```ruby
require('minitest/autorun')
require('minitest/rg')
require_relative('../pet_shop')

class TestPetShop < Minitest::Test

  def setup

    @customers = [
      {
        name: "Alice",
        pets: [],
        cash: 1000
      },
      {
        name: "Bob",
        pets: [],
        cash: 50
      }
    ]
```

This screen shot of class TestPetShop shops an example called "@customers" of a hash within an array. The array has two hashes, each with the keys name, pets and cash.
Description here

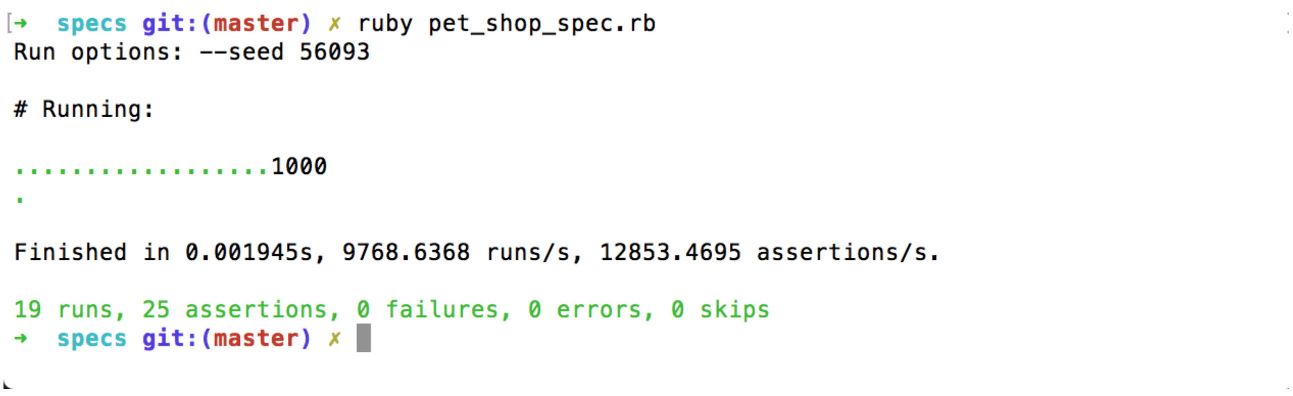


```ruby

def customer_cash(customer)
  return customer[:cash]
end
```

Here we have a function called "customer_cash". This function is passed an argument "customer". It specifies to return the value of the key "cash".



```ruby
  def test_customer_cash
    cash = customer_cash(@customers[0])
    p cash
    assert_equal(1000, cash)
  end
```

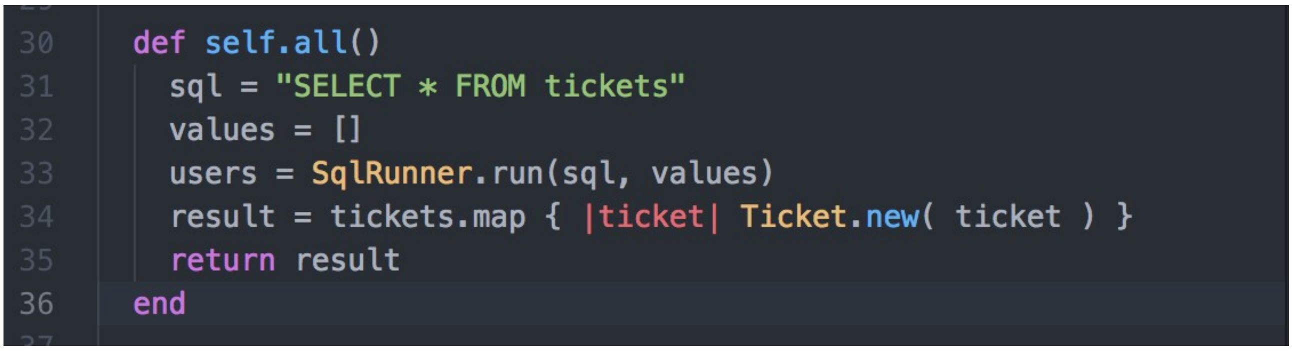


```
[➜  specs git:(master) ✗ ruby pet_shop_spec.rb
Run options: --seed 56093

# Running:

.................1000
.

Finished in 0.001945s, 9768.6368 runs/s, 12853.4695 assertions/s.

19 runs, 25 assertions, 0 failures, 0 errors, 0 skips
➜  specs git:(master) ✗
```

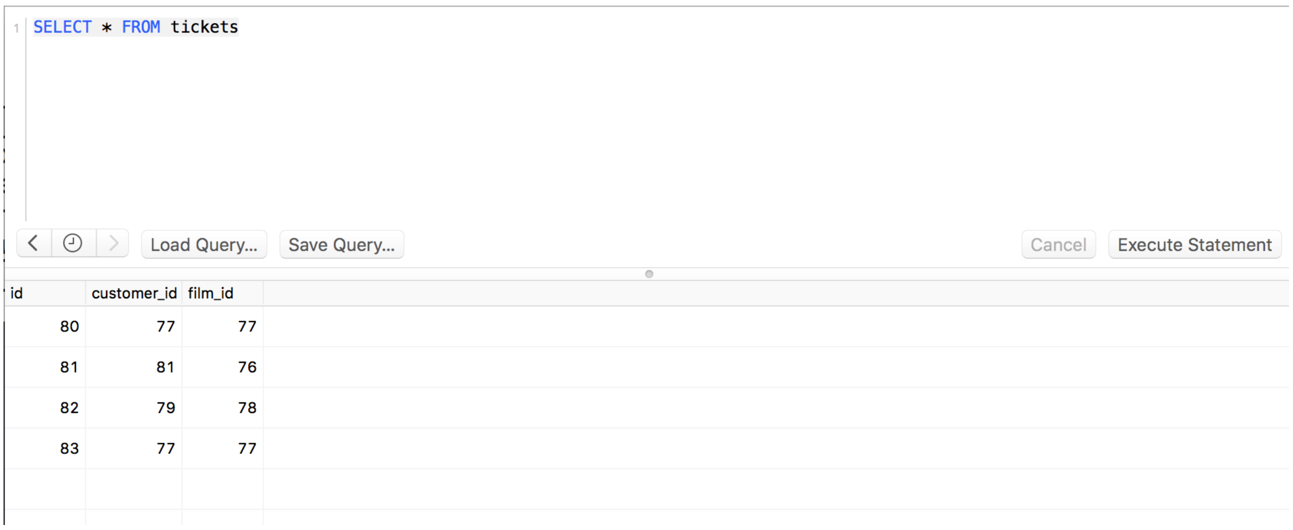
Finally, these last two screen shots show the result of the running the function via test_customer_cash. The function is being passed the argument of the first hash (@customer[0]) I've printed the result of 1000 to the terminal.

Week 3

Unit	Ref	Evidence	
I&T	I.T.3	Demonstrate searching data in a program. Take screenshots of: *Function that searches data *The result of the function running	
		Description:	



```
def self.all()
  sql = "SELECT * FROM tickets"
  values = []
  users = SqlRunner.run(sql, values)
  result = tickets.map { |ticket| Ticket.new( ticket ) }
  return result
end
```



```
SELECT * FROM tickets
```

id	customer_id	film_id
80	77	77
81	81	76
82	79	78
83	77	77

The first screenshot here displays the function "self.all" in which the sql command is selecting all the fields in the tickets table.

The second screenshot displays the result.

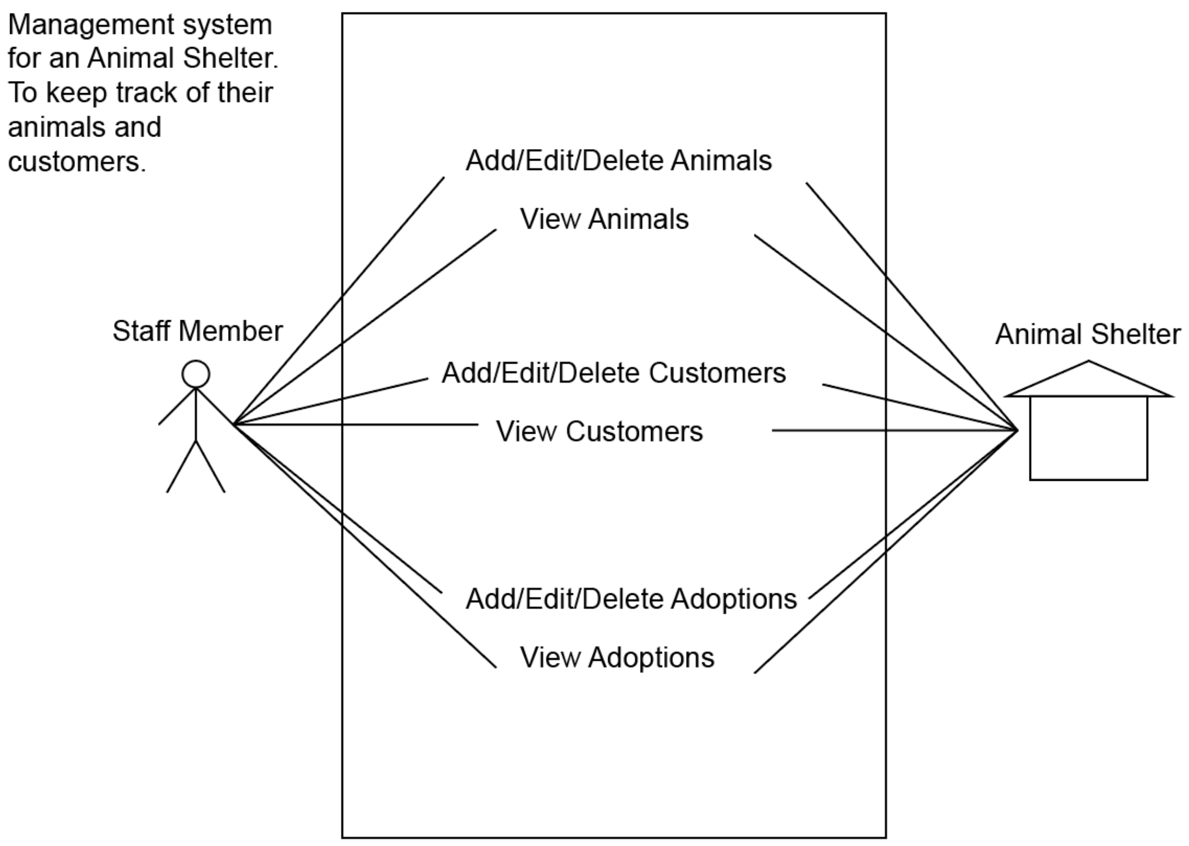
Unit	Ref	Evidence	
I&T	I.T.4	Demonstrate sorting data in a program. Take screenshots of: *Function that sorts data *The result of the function running	
		Description:	

Paste Screenshot here

Description here

Week 5 and 6

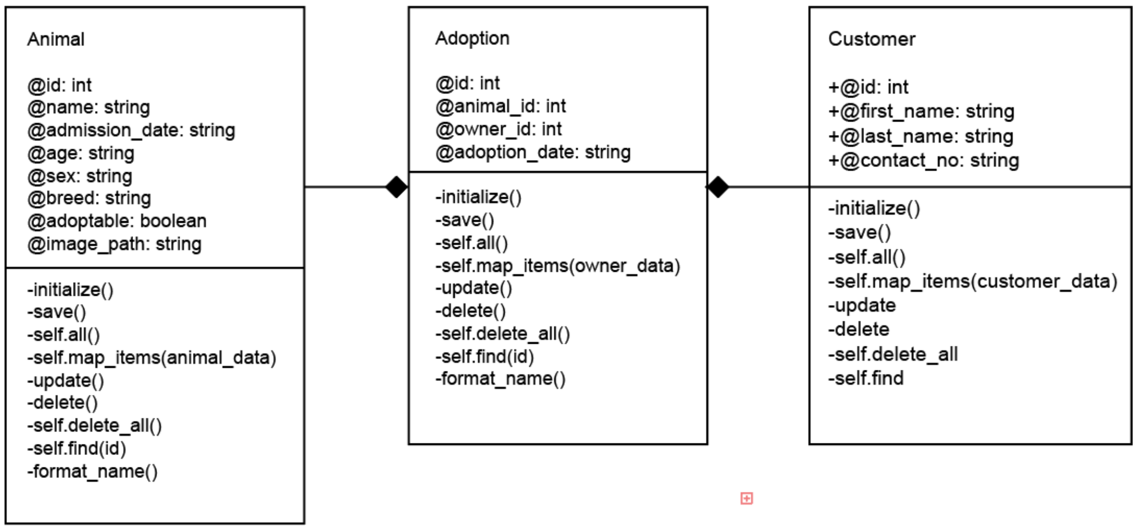
Unit	Ref	Evidence	
A&D	A.D.1	A Use Case Diagram	
		Description:	


Management system for an Animal Shelter. To keep track of their animals and customers.
Add/Edit/Delete Animals
View Animals
Staff Member
Add/Edit/Delete Customers
View Customers
Animal Shelter
Add/Edit/Delete Adoptions
View Adoptions

This is a Use Case diagram of my Animal Shelter project. I had to design a management system for an Animal Shelter. They should be able to keep track of their animals and customers. Here we have the Staff Member, they should be able to add/edit/delete the animals, customers and adoptions and also view them.

Unit	**Ref**	**Evidence**	
A&D	A.D.2	A Class Diagram	
		Description:	

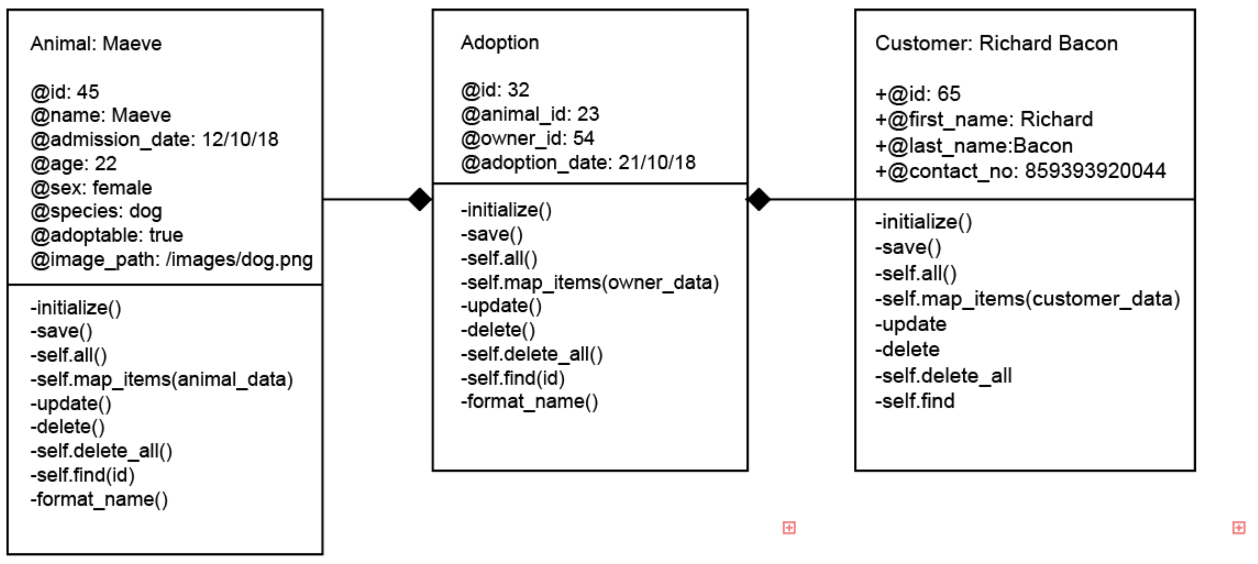
Animal Shelter



<u>Description here</u>

Unit	Ref	Evidence	
A&D	A.D.3	An Object Diagram	
		Description:	

Animal Shelter



Unit	**Ref**	**Evidence**	
A&D	A.D.4	An Activity Diagram	
		Description:	

Description here

Unit	**Ref**	**Evidence**	
A&D	A.D.6	Produce an Implementations Constraints plan detailing the following factors: *Hardware and software platforms *Performance requirements *Persistent storage and transactions *Usability *Budgets *Time	
		Description:	

Paste Screenshot here

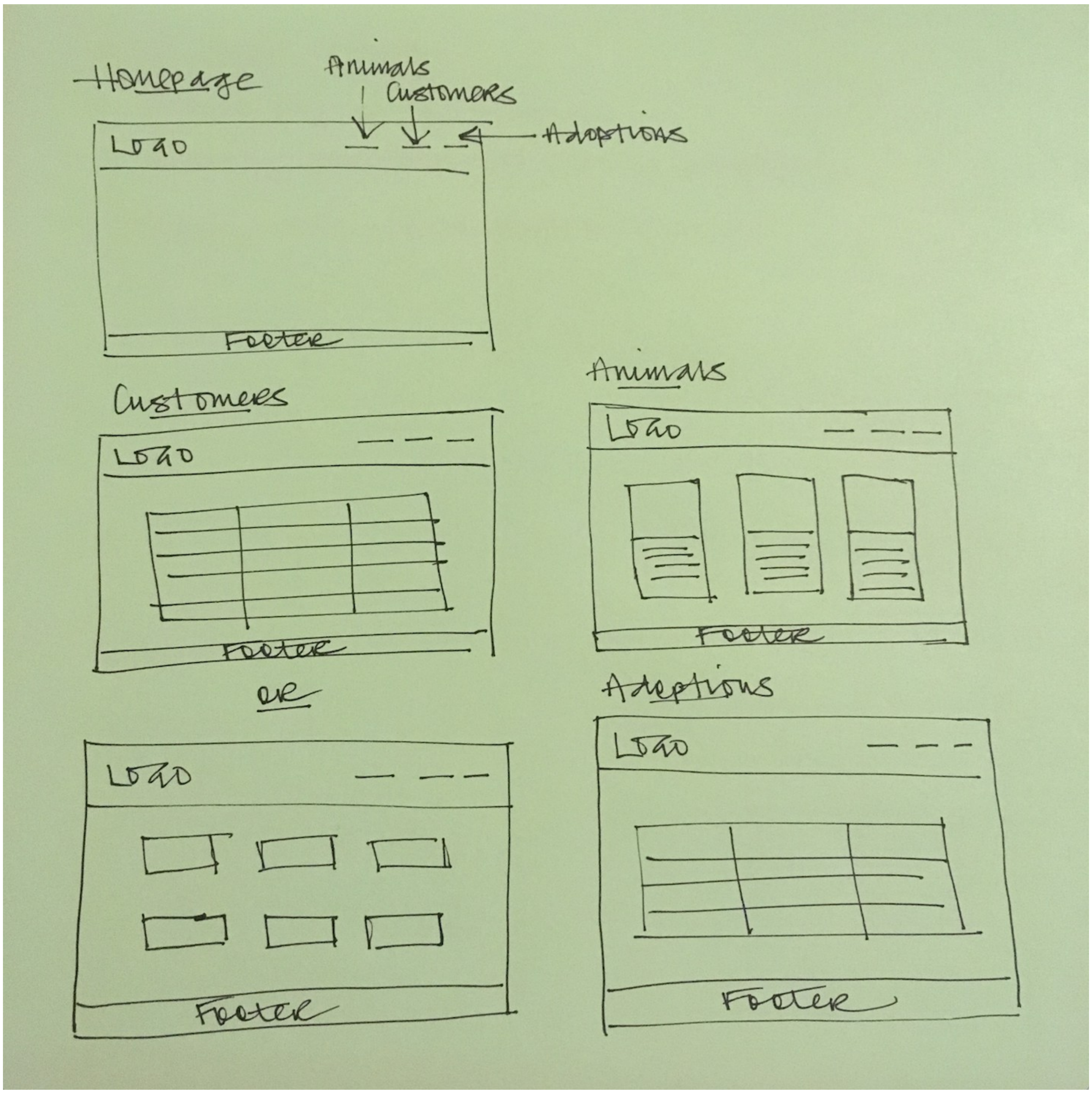
Description here

Unit	Ref	Evidence	
P	P.5	User Site Map	
		Description:	

Paste Screenshot here

Description here

Unit	Ref	Evidence	
P	P.6	2 Wireframe Diagrams	
		Description:	



Description here

Unit	Ref	Evidence	
P	P.10	Example of Pseudocode used for a method	
		Description:	

Paste Screenshot here

Description here

Unit	Ref	Evidence	
P	P.13	Show user input being processed according to design requirements. Take a screenshot of: * The user inputting something into your program * The user input being saved or used in some way	
		Description:	

<u>Paste Screenshot here</u>

<u>Description here</u>

Unit	Ref	Evidence	
P	P.14	Show an interaction with data persistence. Take a screenshot of: * Data being inputted into your program * Confirmation of the data being saved	
		Description:	

<u>Paste Screenshot here</u>

<u>Description here</u>

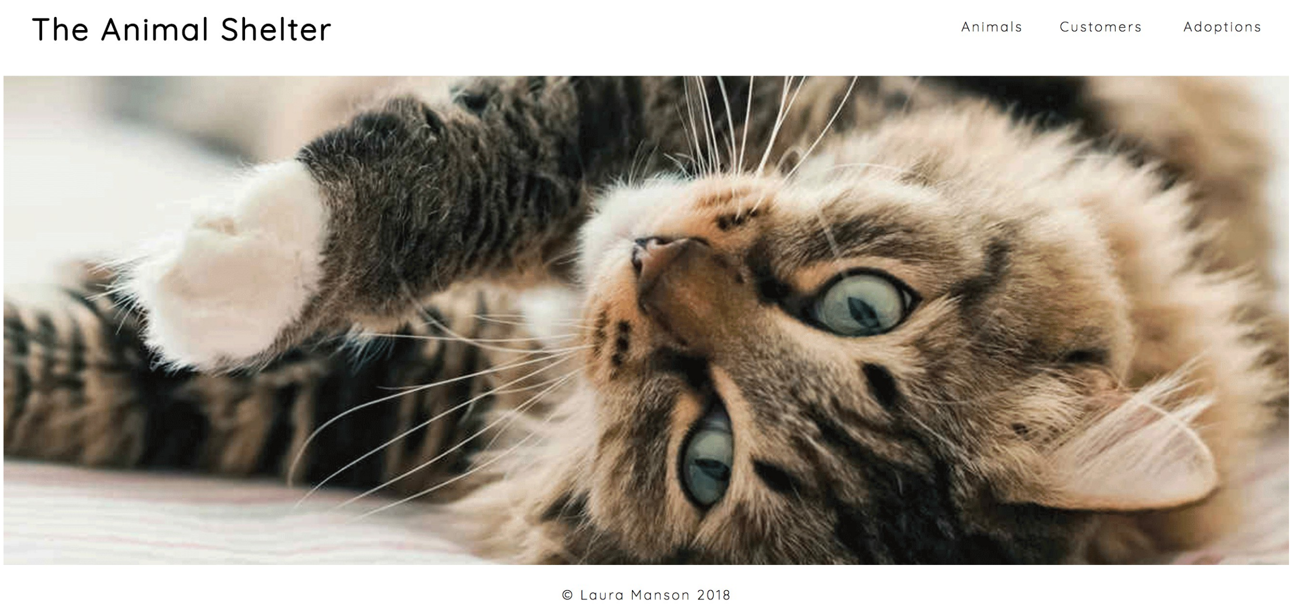
Unit	Ref	Evidence	
P	P.15	Show the correct output of results and feedback to user. Take a screenshot of: * The user requesting information or an action to be performed * The user request being processed correctly and demonstrated in the program	
		Description:	

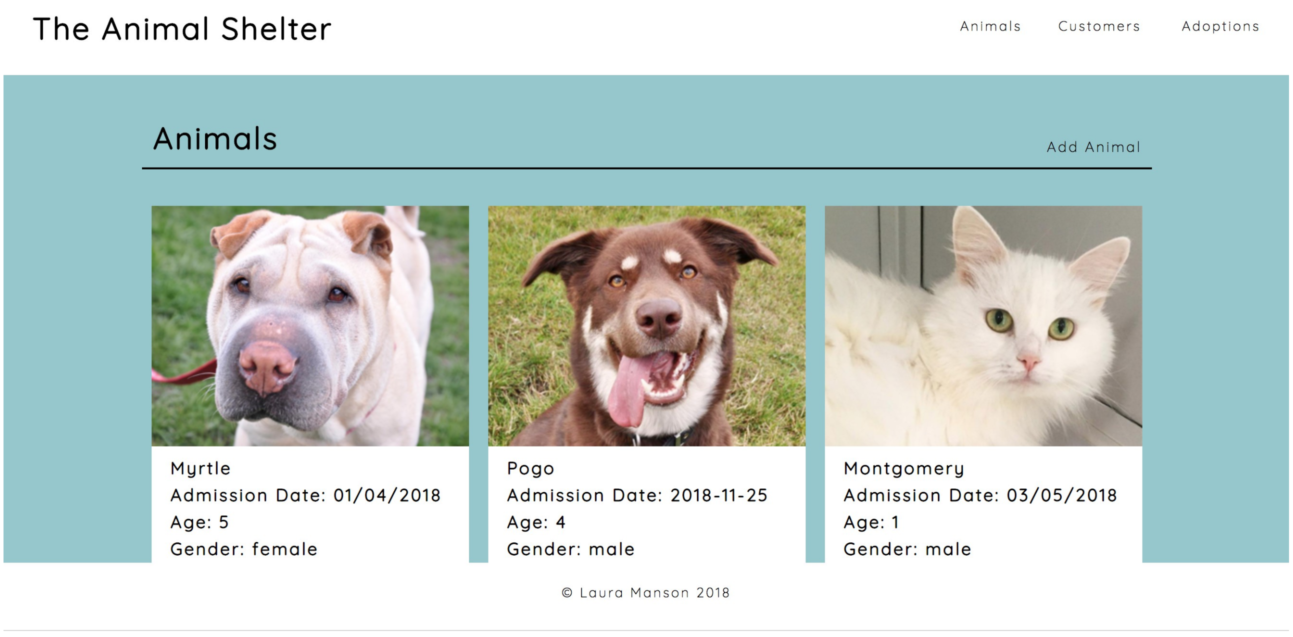
<u>Paste Screenshot here</u>

<u>Description here</u>

Unit	Ref	Evidence	
P	P.11	Take a screenshot of one of your projects where you have worked alone and attach the Github link.	
		Description:	

Github for Animal Shelter Project: https://github.com/ilikebees/animal_shelter

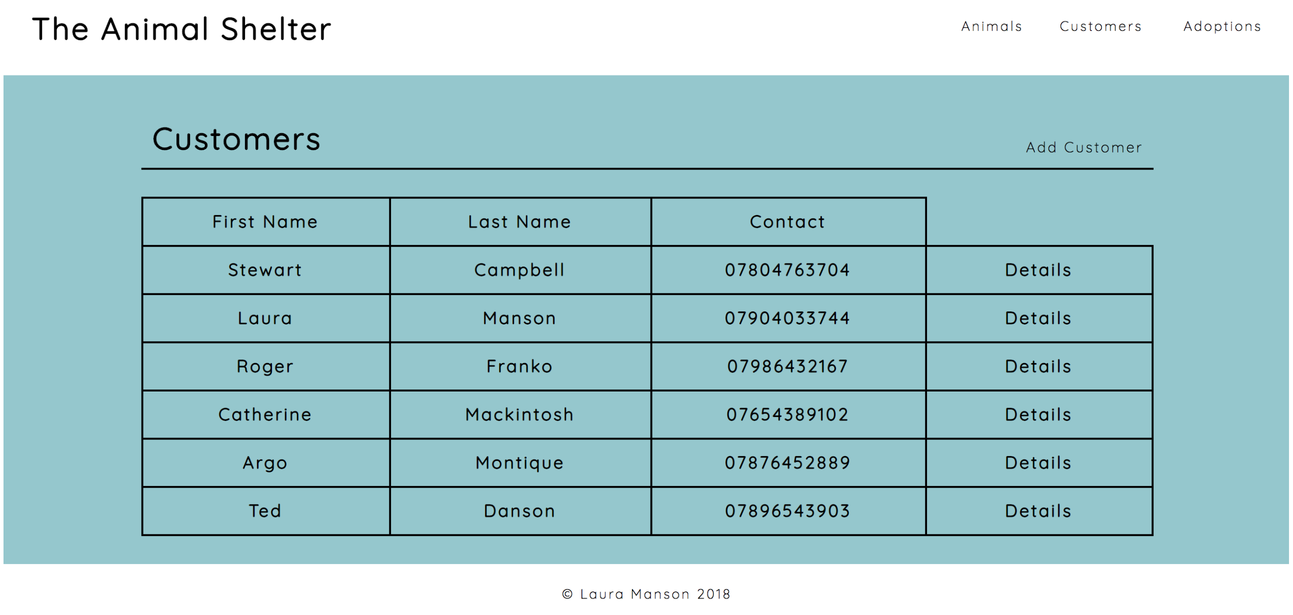




Myrtle
Admission Date: 01/04/2018
Age: 5
Gender: female

Pogo
Admission Date: 2018-11-25
Age: 4
Gender: male

Montgomery
Admission Date: 03/05/2018
Age: 1
Gender: male

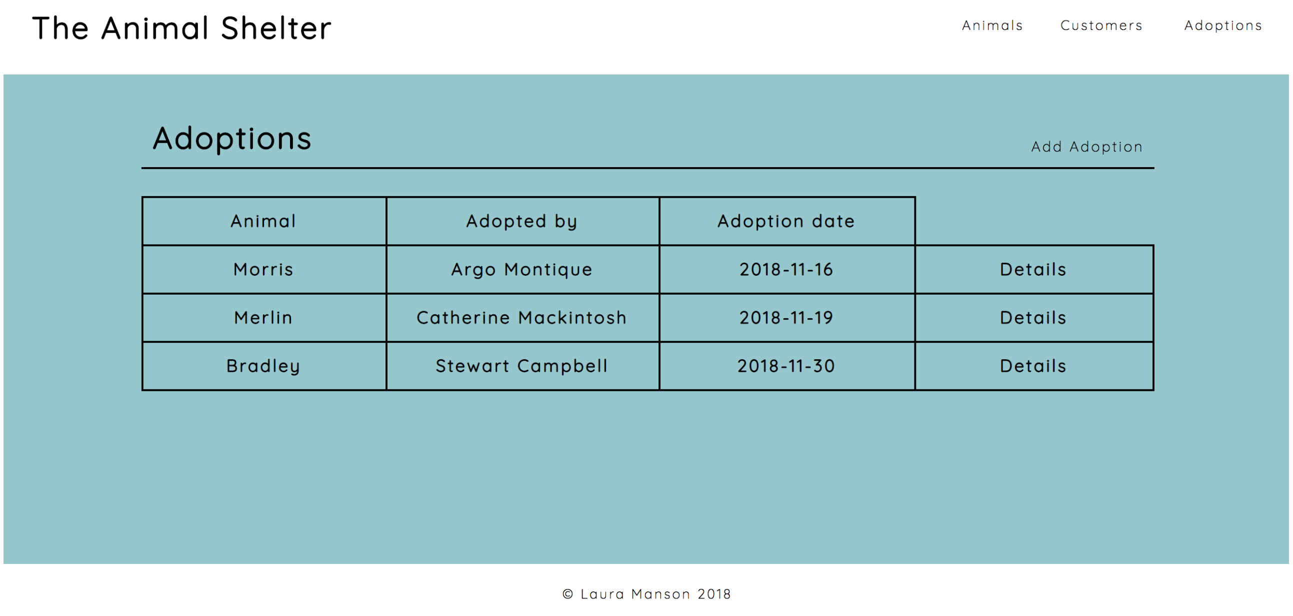


The Animal Shelter

Customers

First Name	Last Name	Contact	
Stewart	Campbell	07804763704	Details
Laura	Manson	07904033744	Details
Roger	Franko	07986432167	Details
Catherine	Mackintosh	07654389102	Details
Argo	Montique	07876452889	Details
Ted	Danson	07896543903	Details

© Laura Manson 2018



The Animal Shelter

Adoptions

Animal	Adopted by	Adoption date	
Morris	Argo Montique	2018-11-16	Details
Merlin	Catherine Mackintosh	2018-11-19	Details
Bradley	Stewart Campbell	2018-11-30	Details

© Laura Manson 2018

Description here

Unit	Ref	Evidence	
P	P.12	Take screenshots or photos of your planning and the different stages of development to show changes.	
		Description:	

Paste Screenshot here

Description here

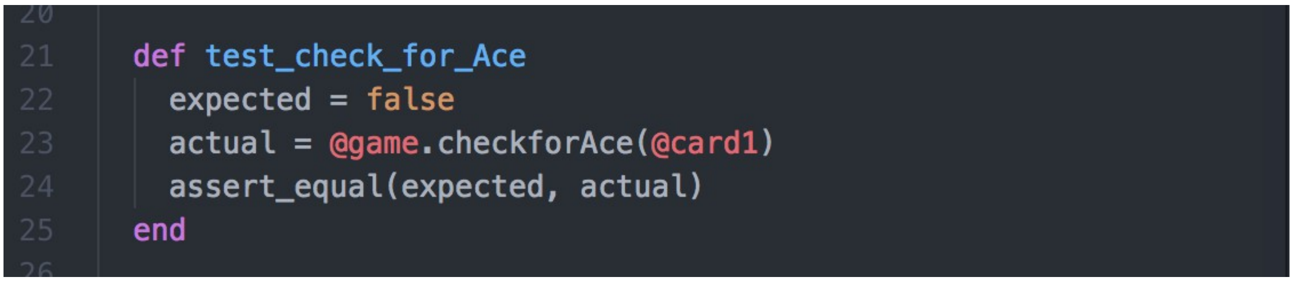
<u>Week 7</u>

Unit	Ref	Evidence	
P	P.16	Show an API being used within your program. Take a screenshot of: * The code that uses or implements the API * The API being used by the program whilst running	
		Description:	

<u>Paste Screenshot here</u>

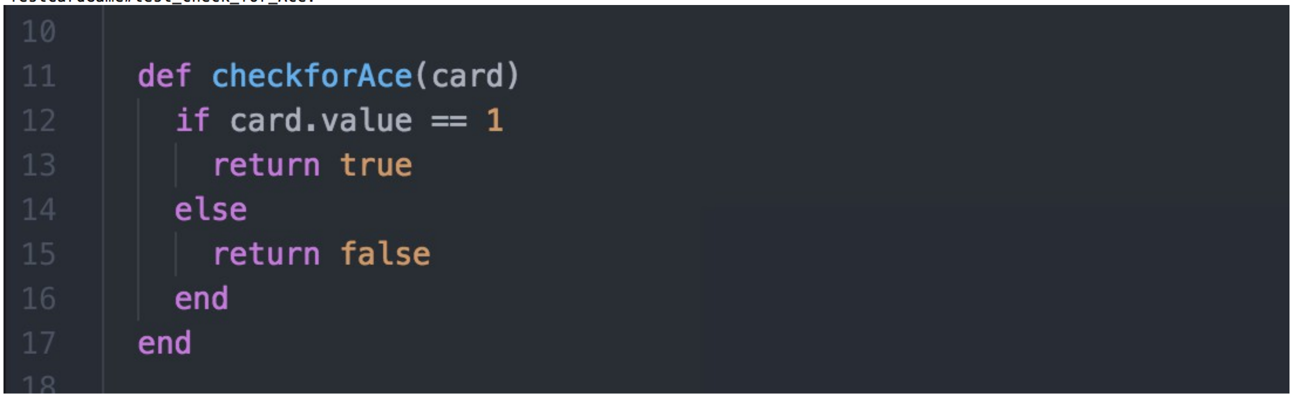
<u>Description here</u>

Unit	Ref	Evidence	
P	P.18	Demonstrate testing in your program. Take screenshots of: * Example of test code * The test code failing to pass * Example of the test code once errors have been corrected * The test code passing	
		Description:	

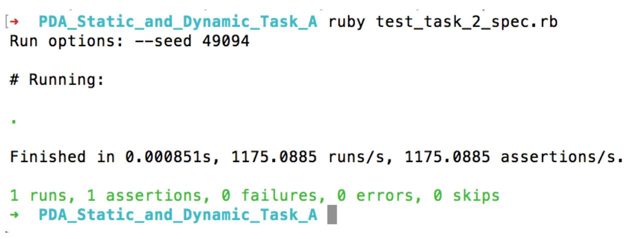


```ruby
def test_check_for_Ace
  expected = false
  actual = @game.checkforAce(@card1)
  assert_equal(expected, actual)
end
```

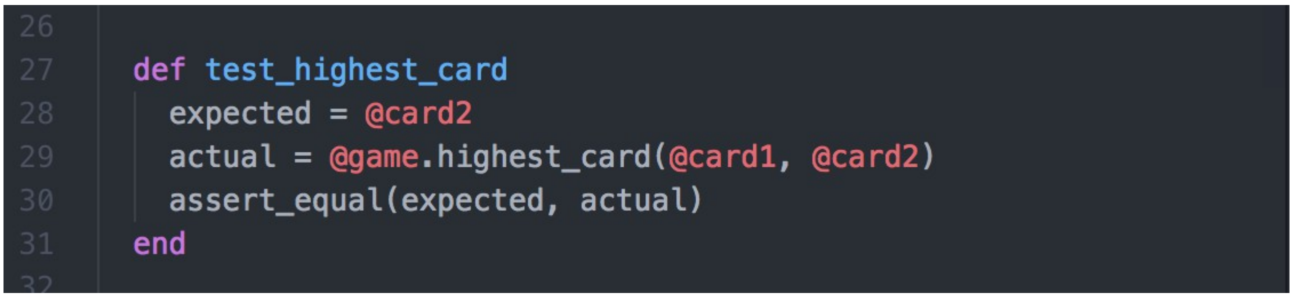
```
Run options: --seed 22933

# Running:

E

Finished in 0.001240s, 806.4515 runs/s, 0.0000 assertions/s.

  1) Error:
TestCardGame#test_check_for_Ace:
```



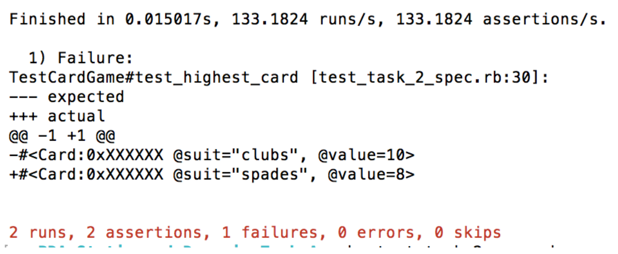
```ruby
def checkforAce(card)
  if card.value == 1
    return true
  else
    return false
  end
end
```



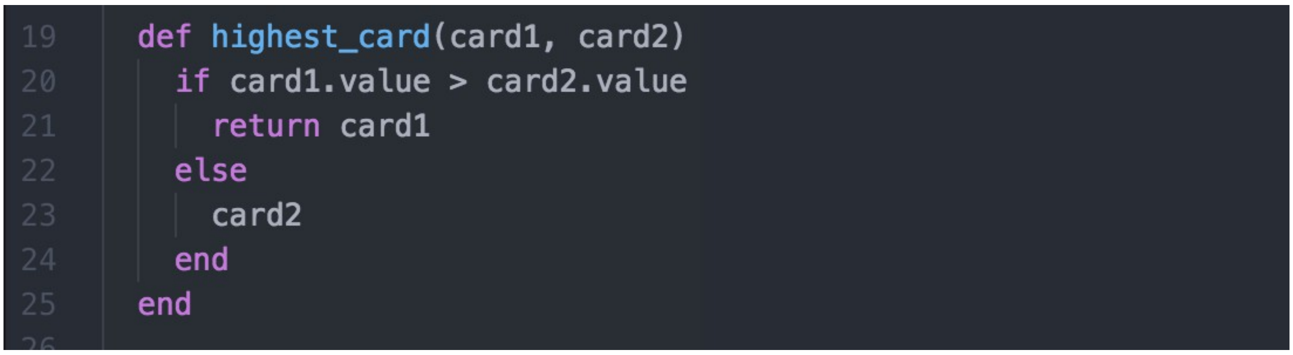
```
→  PDA_Static_and_Dynamic_Task_A ruby test_task_2_spec.rb
Run options: --seed 49094

# Running:

.

Finished in 0.000851s, 1175.0885 runs/s, 1175.0885 assertions/s.

1 runs, 1 assertions, 0 failures, 0 errors, 0 skips
→  PDA_Static_and_Dynamic_Task_A
```



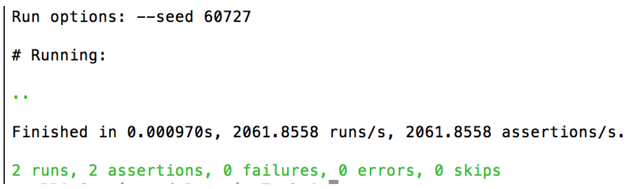
```ruby
26
27    def test_highest_card
28      expected = @card2
29      actual = @game.highest_card(@card1, @card2)
30      assert_equal(expected, actual)
31    end
```



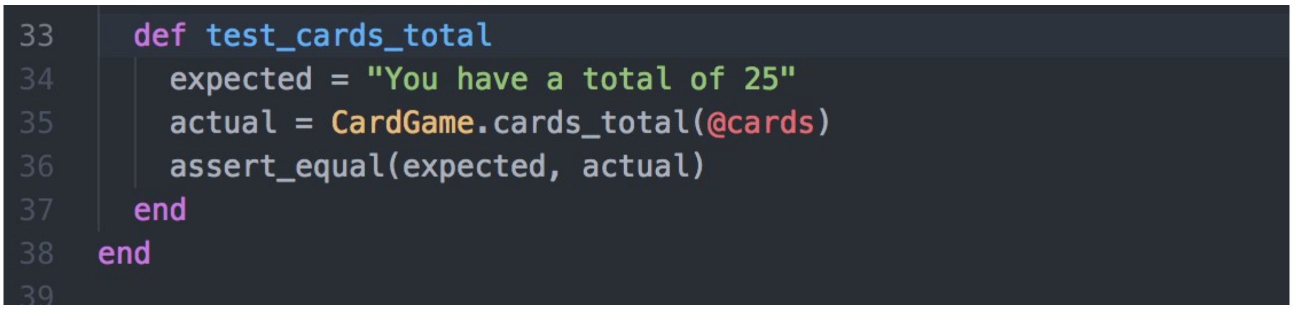
```
Finished in 0.015017s, 133.1824 runs/s, 133.1824 assertions/s.

  1) Failure:
TestCardGame#test_highest_card [test_task_2_spec.rb:30]:
--- expected
+++ actual
@@ -1 +1 @@
-#<Card:0xXXXXXX @suit="clubs", @value=10>
+#<Card:0xXXXXXX @suit="spades", @value=8>


2 runs, 2 assertions, 1 failures, 0 errors, 0 skips
```



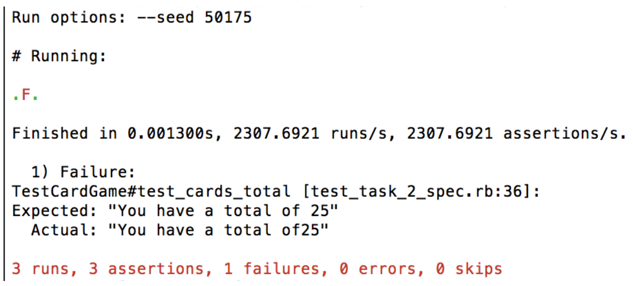
```ruby
def highest_card(card1, card2)
  if card1.value > card2.value
    return card1
  else
    card2
  end
end
```



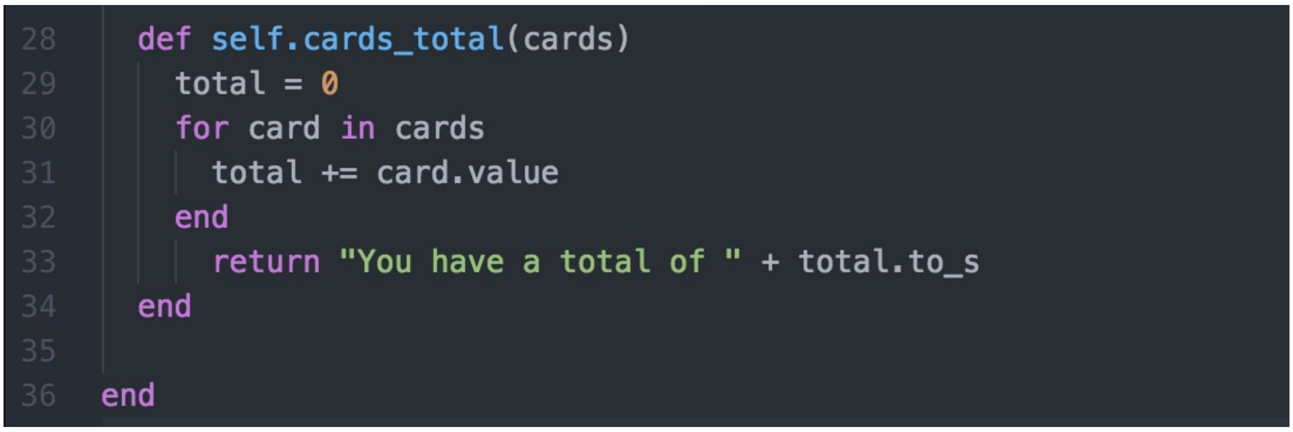
```
Run options: --seed 60727

# Running:

..

Finished in 0.000970s, 2061.8558 runs/s, 2061.8558 assertions/s.

2 runs, 2 assertions, 0 failures, 0 errors, 0 skips
```



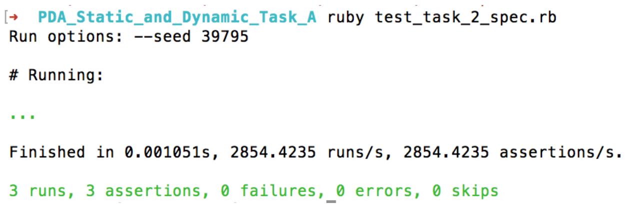
```
  def test_cards_total
    expected = "You have a total of 25"
    actual = CardGame.cards_total(@cards)
    assert_equal(expected, actual)
  end
end
```



```
Run options: --seed 50175

# Running:

.F.

Finished in 0.001300s, 2307.6921 runs/s, 2307.6921 assertions/s.

  1) Failure:
TestCardGame#test_cards_total [test_task_2_spec.rb:36]:
Expected: "You have a total of 25"
  Actual: "You have a total of25"

3 runs, 3 assertions, 1 failures, 0 errors, 0 skips
```



```
  def self.cards_total(cards)
    total = 0
    for card in cards
      total += card.value
    end
      return "You have a total of " + total.to_s
  end

end
```



```
➜  PDA_Static_and_Dynamic_Task_A ruby test_task_2_spec.rb
Run options: --seed 39795

# Running:

...

Finished in 0.001051s, 2854.4235 runs/s, 2854.4235 assertions/s.

3 runs, 3 assertions, 0 failures, 0 errors, 0 skips
```

Unit	Ref	Evidence	
P	P.1	Take a screenshot of the contributor's page on Github from your group project to show the team you worked with.	
		Description:	

Paste Screenshot here

Description here

Unit	Ref	Evidence	
P	P.2	Take a screenshot of the project brief from your group project.	
		Description:	

Paste Screenshot here

Description here

Unit	Ref	Evidence	
P	P.3	Provide a screenshot of the planning you completed during your group project, e.g. Trello MOSCOW board.	
		Description:	

Paste Screenshot here

Description here

Unit	Ref	Evidence	
P	P.4	Write an acceptance criteria and test plan.	

Paste Screenshot here

Description here

Unit	Ref	Evidence	
P	P.7	Produce two system interaction diagrams (sequence and/or collaboration diagrams).	
		Description:	

Paste Screenshot here

Description here

Unit	Ref	Evidence	
P	P.8	Produce two object diagrams.	
		Description:	

Paste Screenshot here

Description here

Unit	Ref	Evidence	
P	P.17	Produce a bug tracking report	
		Description:	

Paste Screenshot here

Description here

Week 12

Unit	Ref	Evidence	
I&T	I.T.7	The use of Polymorphism in a program and what it is doing.	
		Description:	

Paste Screenshot here

Description here

Unit	Ref	Evidence	
A&D	A.D.5	An Inheritance Diagram	
		Description:	

Paste Screenshot here

Description here

Unit	Ref	Evidence	
I&T	I.T.1	The use of Encapsulation in a program and what it is doing.	
		Description:	

Paste Screenshot here

Description here

Unit	Ref	Evidence	
I&T	I.T.2	Take a screenshot of the use of Inheritance in a program. Take screenshots of: *A Class *A Class that inherits from the previous class *An Object in the inherited class *A Method that uses the information inherited from another class.	
		Description:	

Paste Screenshot here

Description here

Unit	Ref	Evidence	
P	P.9	Select two algorithms you have written (NOT the group project). Take a screenshot of each and write a short statement on why you have chosen to use those algorithms.	
		Description:	

Paste Screenshot here

Description here