

Evidence Gathering Document for SQA Level 8 Professional Developer Award.

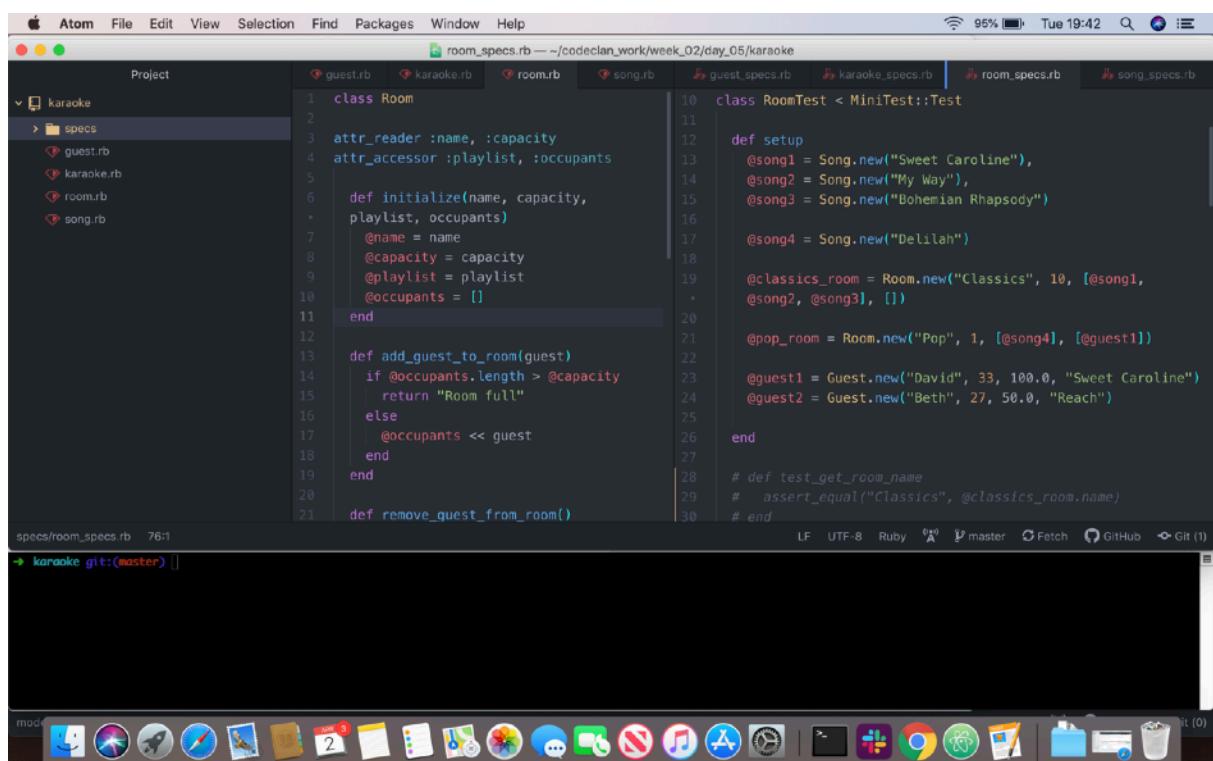
This document is designed for you to present your screenshots and diagrams relevant to the PDA and to also give a short description of what you are showing to clarify understanding for the assessor.

Each point that required details the Assessment Criteria (What you have to show) along with a brief description of the kind of things you should be showing.

Please fill in each point with screenshot or diagram and description of what you are 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: This was a program which contained a few arrays. The photos below show the test result of a function which was created to add a guest to a specific room. A guest is added to the empty 'occupants' array which is then counted to show that it now contains 1 item (or guest).



The screenshot shows the Atom code editor interface. On the left, the project structure is visible with files like guest.rb, karaoke.rb, room.rb, song.rb, guest_specs.rb, karaoke_specs.rb, room_specs.rb, and song_specs.rb. The main editor area displays two files side-by-side: room_specs.rb and RoomTest.rb. room_specs.rb contains Ruby code for a Room class with methods for initializing, adding guests, and removing guests. RoomTest.rb contains MiniTest::Test code for testing the Room class, including setup methods for songs and room instances, and test cases for adding guests and getting room names. The status bar at the bottom shows the file paths, line numbers (76:1), and various system icons.

```
room_specs.rb — ~/codeclan_work/week_02/day_05/karaoke
Atom  File  Edit  View  Selection  Find  Packages  Window  Help
95%  Tue 19:42  Q  ⌘

Project
karaoke
specs
guest.rb
karaoke.rb
room.rb
song.rb
guest_specs.rb
karaoke_specs.rb
room_specs.rb
song_specs.rb

1  class Room
2
3  attr_reader :name, :capacity
4  attr_accessor :playlist, :occupants
5
6  def initialize(name, capacity,
7                 playlist, occupants)
8    @name = name
9    @capacity = capacity
10   @playlist = playlist
11   @occupants = []
12 end
13
14 def add_guest_to_room(guest)
15   if @occupants.length > @capacity
16     return "Room full"
17   else
18     @occupants << guest
19   end
20 end
21
22 def remove_guest_from_room()
23
24
25
26
27
28
29
30

class RoomTest < MiniTest::Test
def setup
  @song1 = Song.new("Sweet Caroline")
  @song2 = Song.new("My Way")
  @song3 = Song.new("Bohemian Rhapsody")
  @song4 = Song.new("Delilah")
end
@classics_room = Room.new("Classics", 10, [@song1, @song2, @song3], [])
@pop_room = Room.new("Pop", 1, [@song4], [{}])
@guest1 = Guest.new("David", 33, 100.0, "Sweet Caroline")
@guest2 = Guest.new("Beth", 27, 50.0, "Reach")
end

# def test_get_room_name
#   assert_equal("Classics", @classics_room.name)
# end

LF  UTF-8  Ruby  ⌘A  ⌘P master  ⌘F Fetch  ⌘G GitHub  ⌘Git (1)

spec/room_specs.rb  76:1
→ karaoke git:(master) []
```

Atom File Edit View Selection Find Packages Window Help

room_specs.rb —~/codeclan_work/week_02/day_05/karaoke

```

Project guest.rb karaoke.rb room.rb song.rb guest_specs.rb karaoke_specs.rb room_specs.rb song_specs.rb
  karaoke
    > specs
      guest.rb
      karaoke.rb
      room.rb
      song.rb
  class Room
    attr_reader :name, :capacity
    attr_accessor :playlist, :occupants
    def initialize(name, capacity, playlist, occupants)
      @name = name
      @capacity = capacity
      @playlist = playlist
      @occupants = []
    end
    def add_guest_to_room(guest)
      if @occupants.length > @capacity
        return "Room full"
      else
        @occupants << guest
      end
    end
    def remove_guest_from_room()
    end
  end
  # def test_playlist_length
  #   # playlist = @classics_room.playlist.length
  #   # assert_equal(3, playlist)
  #   # end
  # def test_get_playlist
  #   # assert_equal( [@song1, @song2, @song3], @classics_room.playlist)
  #   # end
  def test_add_guest_to_room
    @classics_room.add_guest_to_room(@guest1)
    assert_equal(1, @classics_room.occupants.length)
  end
  # def test_remove_guest_from_room
  #   # add 2 guests, then remove 1
  #   # @classics_room.add_guest_to_room(@guest1)
  #   # @classics_room.add_guest_to_room(@guest2)
  #   # @classics_room.remove_guest_from_room()
  #   #
  #   # assert_equal(1, @classics_room.occupants.length)
  # end

```

Run options: --seed 45481

Running:

.

Finished in 0.000969s, 1031.9917 runs/s, 1031.9917 assertions/s.

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

→ karaoke git:(master) x

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: This program shows an array of hashes. The test shows the result of adding a pet to the customer pets array contained in a hash.

Terminal Shell Edit View Window Help

pet_shop.rb — codeclan_work/week_01/day_05/start_point —~/codeclan_work/week_01/day_05/start_point

```

pet_shop.rb — codeclan_wor... pet_shop.rb — Downloads/so...
53 # shop[:pets].push(new_pet)
54 # end
55 #
56 # def customer_cash(customer)
57 #   return customer[:cash]
58 # end
59 #
60 # def remove_customer_cash(customer, cash)
61 #   customer[:cash] -= cash
62 # end
63 #
64 def customer_pet_count(customer)
65 |   return customer[:pets].length
66 end
67 #
68 def add_pet_to_customer(customer, pet)
69 |   customer[:pets].push(pet)
70 end
71 #
72 #
73 # def customer_can_afford_pet(customer,
74 |   pet)
75 #   return customer[:cash] > pet[:price]
76 # end
77 #
78 #
79 # def sell_pet_to_customer(shop, pet,
80 |   customer)

```

pet_shop_spec.rb

```

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
      }
    ]
    @new_pet = {
      name: "Bors the Younger",
      pet_type: :cat,
      breed: "Cornish Rex",
      price: 100
    }
    @pet_shop = {
      pets: []
    }
  end

```

start_point git:(master) x ruby specs/pet_shop_spec.rb

Run options: --seed 8095

Running:

.

Finished in 0.000868s, 1152.0737 runs/s, 1152.0737 assertions/s.

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

→ start_point git:(master) x

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: Below we have created a function that searches for all the customers at a specific screening. The terminal output at the bottom of the screenshot shows the two customers who were at screening 1.

The screenshot shows a Mac OS X desktop with several windows open:

- Terminal:** The title bar says "Terminal". The menu bar includes "File", "Edit", "View", "Window", and "Help". The status bar shows "100% battery" and "Mon 10:14".
- Code Editor:** The title bar says "screening.rb — ~/codeclan_work/week_03/homework/cinema". The sidebar shows a "Project" tree with "cinema", "db", "models", and "console.rb". The main editor area contains Ruby code for a "Customer" class.
- Terminal Console:** The title bar says "cinema — ruby console.rb — ruby console.rb — 179x12". It shows a psql prompt and some command-line history.

```
Project cinema.sql sql_runner.rb customer.rb screening.rb film.rb ticket.rb console.rb

cinema
db
models
console.rb

32: Sq lRunner.run(sql)
33: end
34:
35: # lists customers at a specific screening
36: def customers() # eg screening1.customers
37:   sql = "SELECT customers.* FROM customers INNER JOIN tickets ON customers.id = tickets.customer_id WHERE
38:     tickets.screening_id = $1"
39:   values = [@id]
40:   results = SqlRunner.run(sql, values)
41:   return results.map { |customer| Customer.new(customer) }
42:
43:
44: # number of customers at a specific screening
45:
46: def number_of_customers
47:   return customers.length
48: end

models/screening.rb 33:6 LF UTF-8 Ruby ⌂ master ⌂ Fetch ⌂ GitHub ⌂ Git (0)

From: /Users/user/codeclan_work/week_03/homework/cinema/console.rb @ line 59 :
54: ticket5.save()
55:
56:
57:
58: binding.pry
=> 59: nil

[1] pry(main)> screening1.customers
=> [<Customer:0x007fb607a6b4d0 @funds=40, @id=1, @name="David">, <Customer:0x007fb607a6b408 @funds=50, @id=2, @name="Beth">]
[2] pry(main)>
```

Unit	Ref	Evidence
I&T	I.T.4	<p>Demonstrate sorting data in a program. Take screenshots of:</p> <ul style="list-style-type: none"> *Function that sorts data *The result of the function running
		<p>Description: The program below has a function to find the dinosaur that attracts the most visitors. To do this I sorted the dinosaurs in the array by the value of <code>guestsAttractedPerDay</code> and then returned the first result. The result in the test spec shows that <code>dinosaur1</code> was the expected outcome as he had the highest value for <code>guestsAttractedPerDay</code>.</p>

The screenshot shows a Mac desktop environment. In the foreground, the Atom code editor is open with several files visible in the sidebar: `hw_tdd_jurassic_park_start`, `models` (containing `dinosaur.js` and `park.js`), and `specs` (containing `dinosaur_spec.js` and `park_spec.js`). The `park_spec.js` file is currently selected and shown in the main editor area. The code defines a `Park` prototype with methods `addDino`, `removeDino`, and `mostPopular`. The `mostPopular` method uses `slice` and `sort` to sort the array of dinosaurs by `guestsAttractedPerDay` and returns the first one. A test suite in `dinosaur_spec.js` uses Mocha to verify these functions. In the background, a terminal window shows the command `mocha specs` being run, and the output shows test results for `Dinosaur` and `Park` models.

```

 park_spec.js -- ~/codeclan_work/week_06/day_02/hw_tdd_jurassic_park_start
Project
  hw_tdd_jurassic_park_start
    models
      dinosaur.js
      park.js
    specs
      dinosaur_spec.js
      park_spec.js
      .gitignore
      package-lock.json
      package.json

park_spec.js
  6 | Park.prototype.addDino = function(dinosaur){
  7 |   this.dinosaurs.push(dinosaur);
  8 | };
  9 |
 10 |
 11 | Park.prototype.removeDino = function(){
 12 |   this.dinosaurs.pop();
 13 | };
 14 |
 15 |
 16 | Park.prototype.mostPopular = function(){
 17 |   let dino_array = this.dinosaurs.slice()
 18 |   dino_array.sort(function (a, b){
 19 |     return b.guestsAttractedPerDay -
 20 |       a.guestsAttractedPerDay;
 21 |   });
 22 |   return dino_array[0];
 23 |
 24 |

dinosaur_spec.js
  47 | it('should be able to find the dinosaur that
  48 | attracts the most visitors', function(){
  49 |   park.addDino(dinosaur1);
  50 |   park.addDino(dinosaur2);
  51 |   park.addDino(dinosaur3);
  52 |   const actual = park.mostPopular();
  53 |   assert.strictEqual(actual, dinosaur1)
  54 | });
  55 |
  56 | it('should be able to find all dinosaurs of a
  57 | particular species', function(){
  58 |   park.addDino(dinosaur1);
  59 |   park.addDino(dinosaur2);
  60 |   park.addDino(dinosaur1);
  61 |   const actual =
  62 |     park.findBySpecies(dinosaur1.species);
  63 |   assert.deepStrictEqual(actual, [dinosaur1,
  64 |     dinosaur1]);
  65 | });

specs/park_spec.js 29:31 (1, 10) Highlighted: 3
  LF  UTF-8  JavaScript  master  Fetch  GitHub  Git (1)

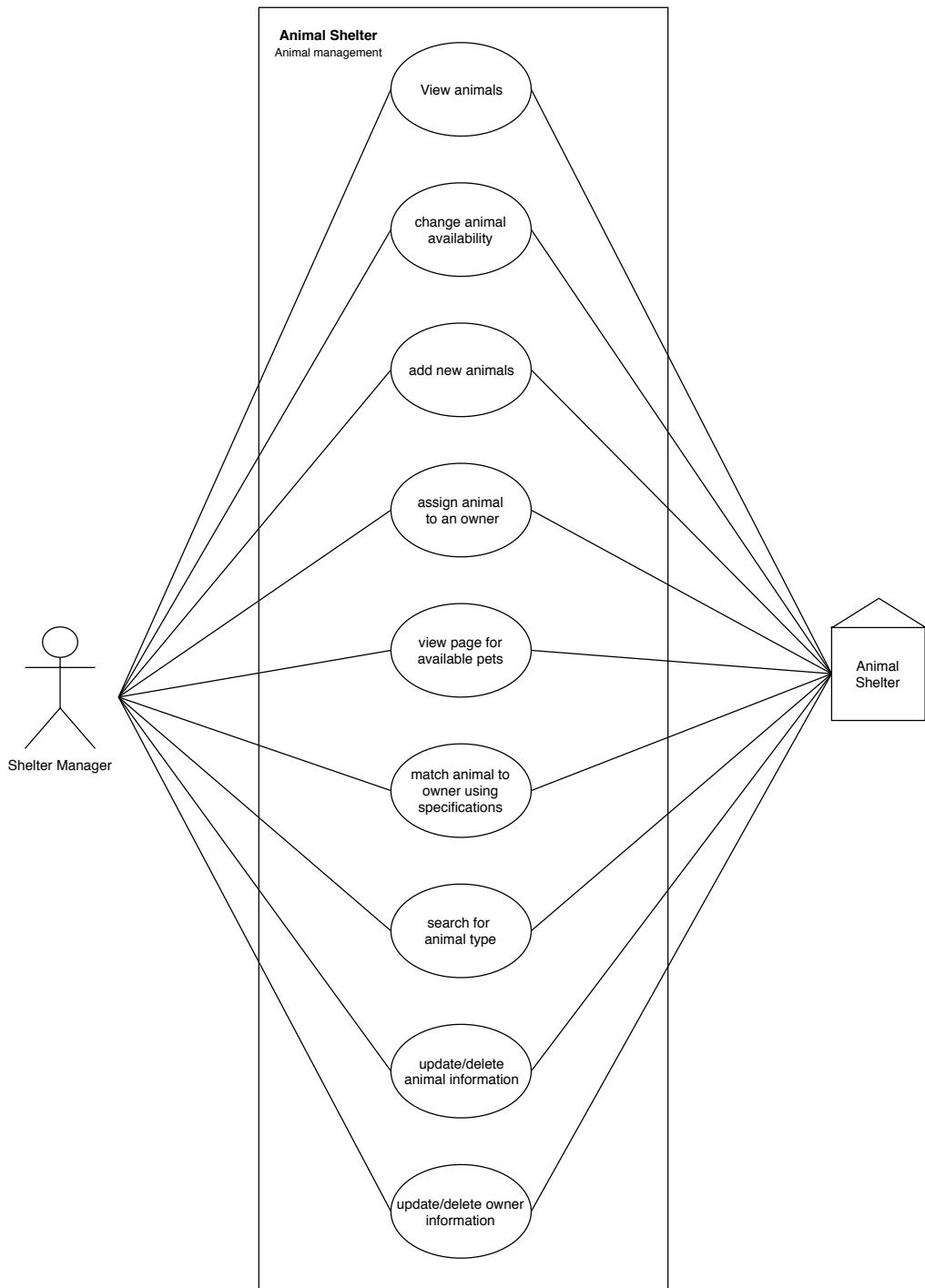
> mocha specs
Dinosaur
  ✓ should have a species
  ✓ should have a diet
  ✓ should have an average number of visitors it attracts per day

Park
  ✓ should have a name
  ✓ should have a ticket price

```

Week 5 and 6

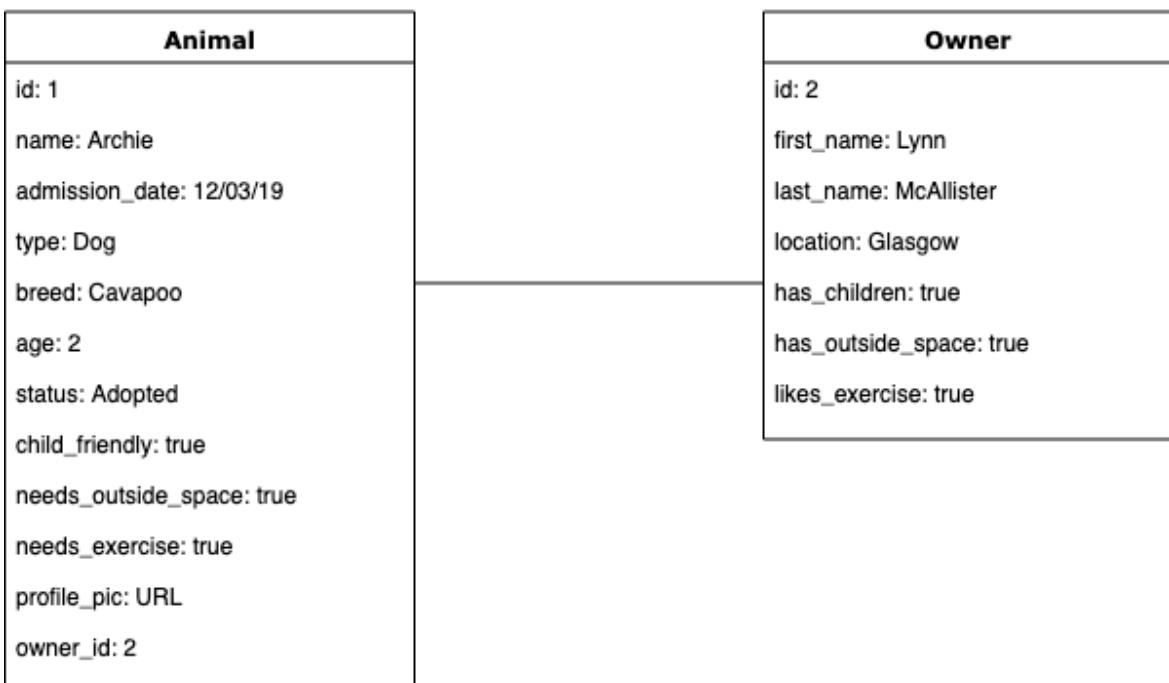
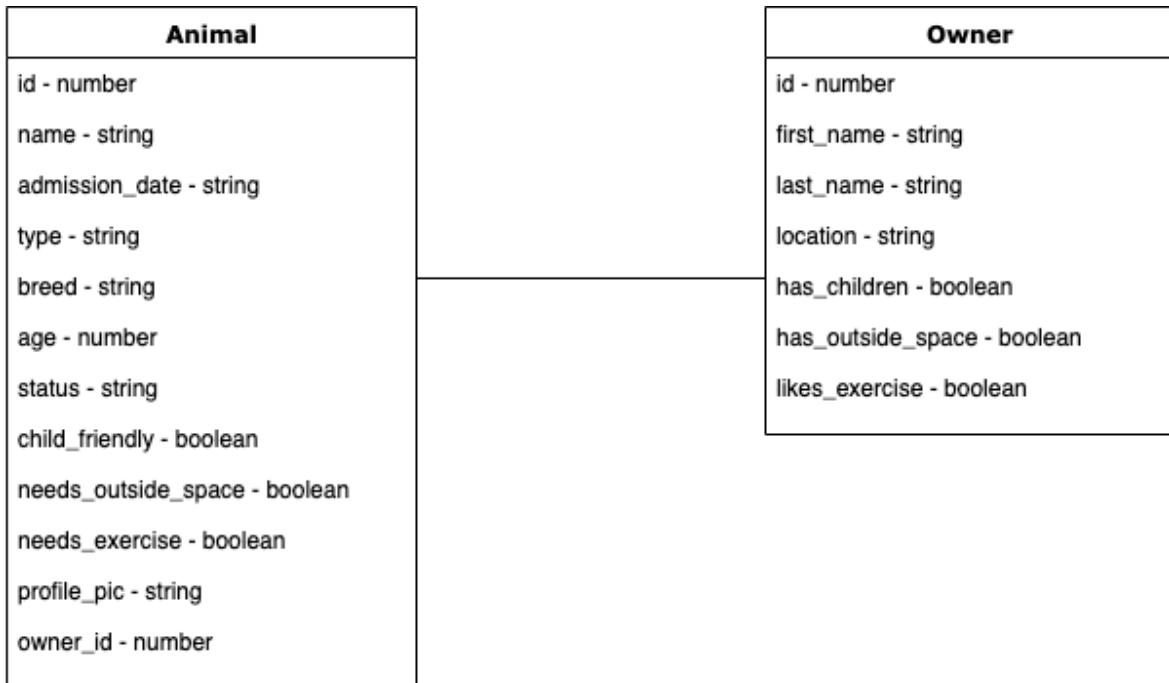
Unit	Ref	Evidence
A&D	A.D.1	A Use Case Diagram
		<p>Description: the use case diagram below shows us all the tasks the shelter manager would need to do on any given day. As the program is not intended to be used by the general public, the shelter manager will be the only user.</p>



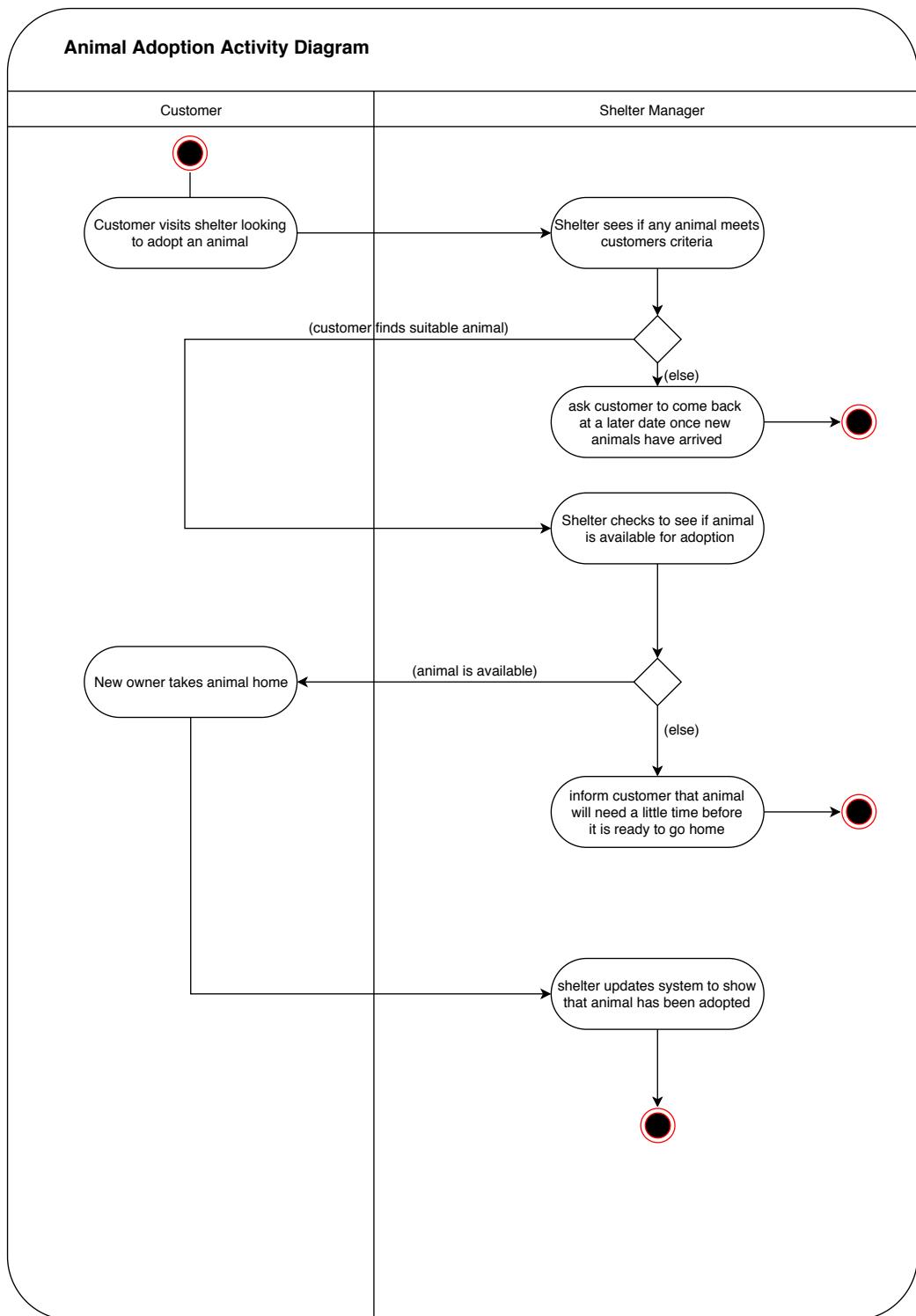
Unit	Ref	Evidence
A&D	A.D.2	A Class Diagram
		Description: The diagram below shows the attributes for each class as well as the methods that will be required. The type of each attribute is also shown.

Animal	Owner
<p>Attributes:</p> <ul style="list-style-type: none"> id - number name - string admission_date - string type - string breed - string age - number status - string child_friendly - boolean needs_outside_space - boolean needs_exercise - boolean profile_pic - string owner_id - number <p>Methods:</p> <ul style="list-style-type: none"> CRUD kid_friendly() outside_space() exercise() owner() self.all_available() self.training() self.vet() self.adopted() self.search(params) 	<p>Attributes:</p> <ul style="list-style-type: none"> id - number first_name - string last_name - string location - string has_children - boolean has_outside_space - boolean likes_exercise - boolean <p>Methods:</p> <ul style="list-style-type: none"> full_name() CRUD needs_children_friendly() outdoor_space () owner_exercise() animals()

Unit	Ref	Evidence
A&D	A.D.3	An Object Diagram
		Description: The object diagram shows an example of the data structure for each class.



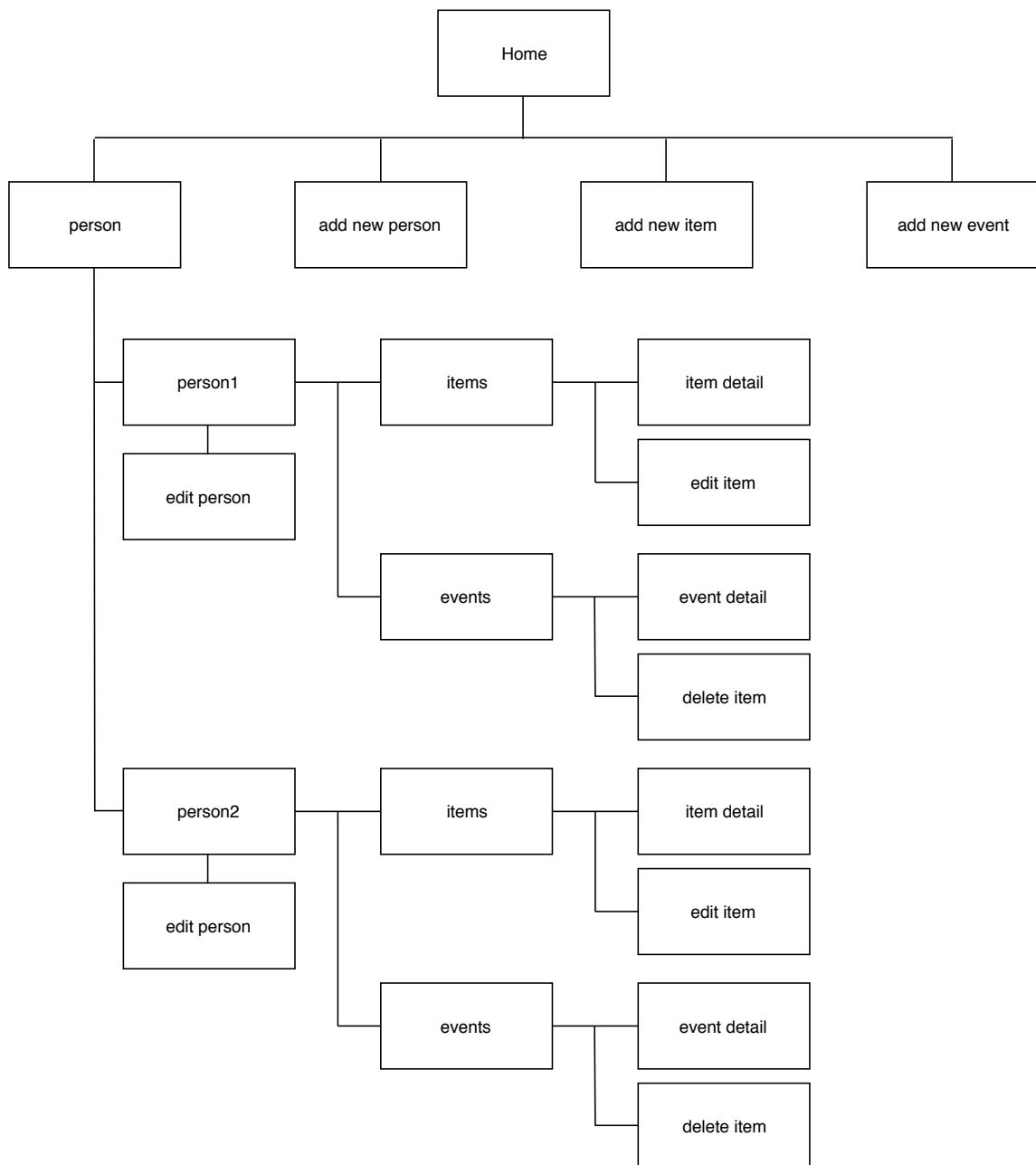
Unit	Ref	Evidence
A&D	A.D.4	An Activity Diagram
		<p>Description: The activity diagram below shows the main journey, from start to finish, for an adoption at the shelter. The diagram shows the outcome from each decision/choice and how that will affect the final outcome.</p>



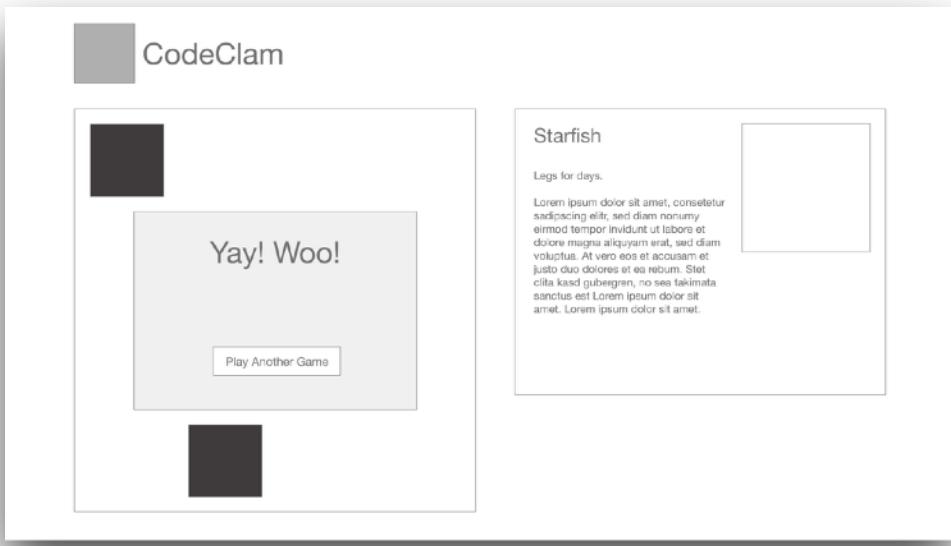
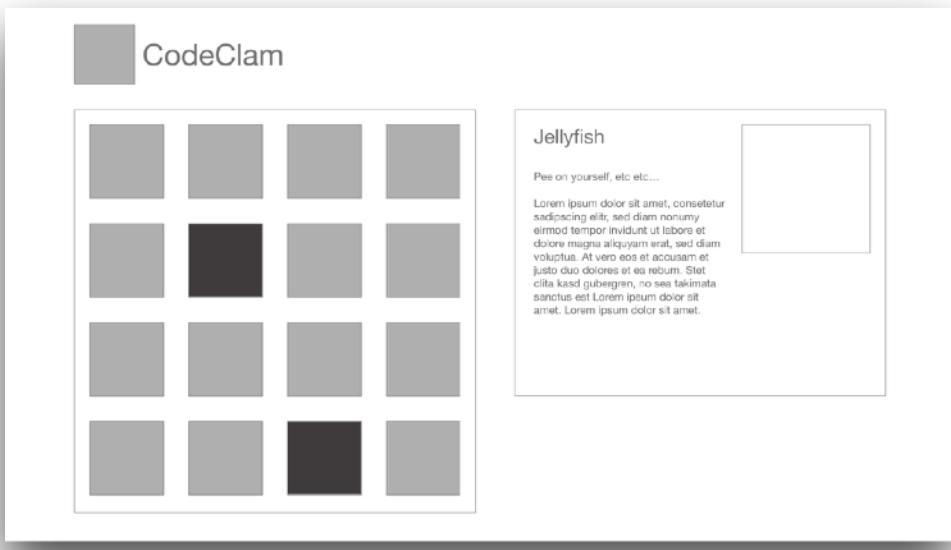
Unit	Ref	Evidence
A&D	A.D.6	<p>Produce an Implementations Constraints plan detailing the following factors:</p> <ul style="list-style-type: none"> *Hardware and software platforms *Performance requirements *Persistent storage and transactions *Usability *Budgets *Time
		<p>Description: Below is an example of an implementations constraints plan. As the website was designed for an imaginary Animal Shelter with no plans to upscale it, many of these potential issues were void.</p>

Constraint Strategy	Implementation constraints	Solution
Hardware and software platforms	No issue with hardware, laptop will be sufficient to build app. By using software we know how to use, will ensure there should be minimal disruption in the design process.	Used Ruby, Sinatra and PostgreSQL which will be easy to set up and maintain.
Performance requirements	App must update instantly to keep records up to date.	Ensured all routes and CRUD methods are working to ensure there will be no issues once the app is launched.
Persistent storage and transactions	Small scale app with no plans to upscale so a small server is more than sufficient.	Storage on Heroku will be sufficient for the purposes of this app.
Usability	App must be simple and clear how to use. It is being asked for to simplify and manage the pets and owners for the animal shelter.	Clear distinction between sections. All routes are clearly labelled.
Budgets	No budget. Need to work on project without any hired support.	Use free storage facilities.
Time	Only had a week to plan and build the application.	Keep the design simple. Done is better than perfect.

Unit	Ref	Evidence
P	P.5	User Site Map
		Description: This site map is for the present planner wish list website we created in the last project. It shows the different pages a user could navigate to when using the website.



Unit	Ref	Evidence
P	P.6	2 Wireframe Diagrams
		<p>Description: The diagrams below show examples of wireframe diagrams for the Codeclam matching pairs game project. The diagram show an early prototype for what the main page was going to be like as well as the final page on game completion. As the project grew the design did change but the use of the wireframes early on, certainly had a major impact on how the final page looked.</p>



Unit	Ref	Evidence
P	P.10	<p>Example of Pseudocode used for a method</p> <p>Description: The pseudocode here just describes exactly what the method is going to do. It will work through each if statement and if the change is greater it will add a coin to the returned coin array and move on to the next coin. It loops round the 20p coin twice as the change could be 40 as it reaches this point and as a result two 20p coins will be required.</p>

```

// this method is used to calculate what coins would be returned
// for a given amount of change

public void workOutChange(int change){
    ArrayList<Coin> returnedCoins = new ArrayList<Coin>();
    if(change >= 200){ // if change is more than 200 a two pound coin can be issued
        this.returnedCoins.add(twoPound);
        change -= 200;
    } if(change >= 100){ // if change is more than 100 a pound coin can be issued
        this.returnedCoins.add(pound);
        change -= 100;
    } if (change >= 50){ // if change is more than 50 a 50p coin can be issued
        this.returnedCoins.add(fifty);
        change -= 50;
    } if (change >= 20){ // if change is more than 20 a 20p coin can be issued
        this.returnedCoins.add(twenty);
        change -= 20;
        if (change >= 20){ // if change is still more than 20 a 20p coin can be issued ie 40p change
            this.returnedCoins.add(twenty);
            change -= 20;
        }
    } if (change >= 10){ // if change is more than 10 a 10p coin can be issued
        this.returnedCoins.add(ten);
        change -= 10;
    } if (change >= 5){ // if change is more than 5 a 5p coin can be issued
        this.returnedCoins.add(five);
        change -= 5;
    }
}

```

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: The brief asked us to be able to add new animals to the shelter, Below shows the user form inputting an entry for a new dinosaur to the shelter. The bottom image shows the profile of Terry the T Rex after it has been saved to the database.

Chrome File Edit View History Bookmarks People Window Help 92% Mon 10:11 Not Secure wannapetanimalshelter.herokuapp.com/animals/new

Add a new animal

Name: Terry

Admission Date: 06/04/2019

Type: Dinosaur

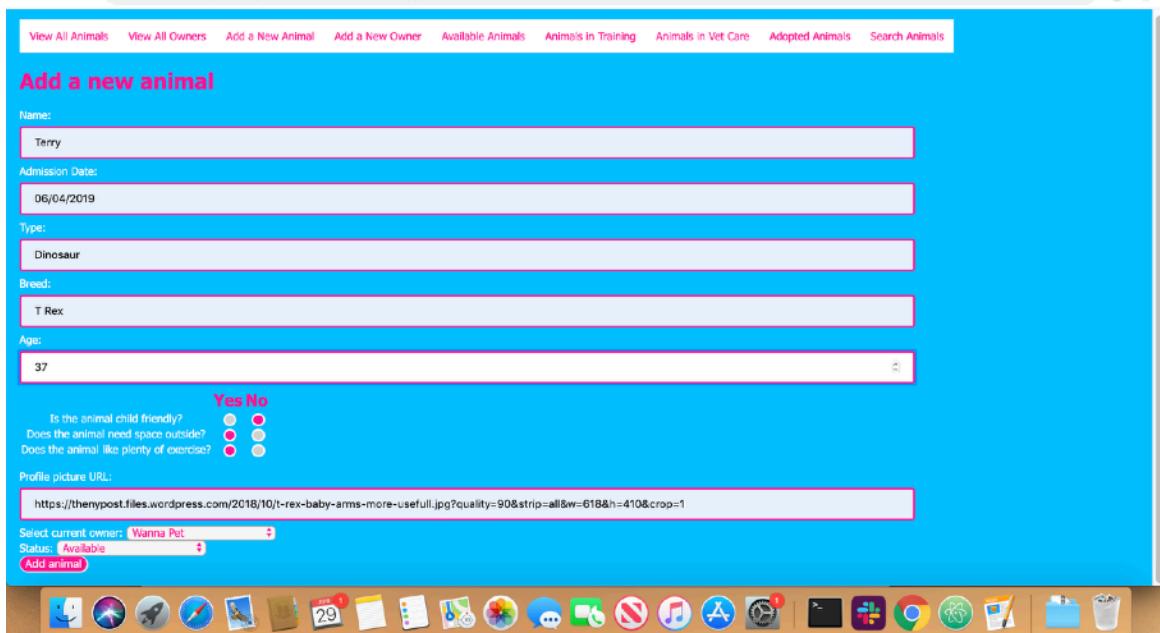
Breed: T Rex

Age: 37

Is the animal child friendly? Yes No
 Does the animal need space outside? Yes No
 Does the animal like plenty of exercise? Yes No

Profile picture URL: <https://thenypost.files.wordpress.com/2018/10/t-rex-baby-arms-more-usefull.jpg?quality=90&strip=all&w=618&h=410&crop=1>

Select current owner: Wanna Pet Status: Available Add animal



Chrome File Edit View History Bookmarks People Window Help 93% Mon 10:11 Not Secure wannapetanimalshelter.herokuapp.com/animals/12?

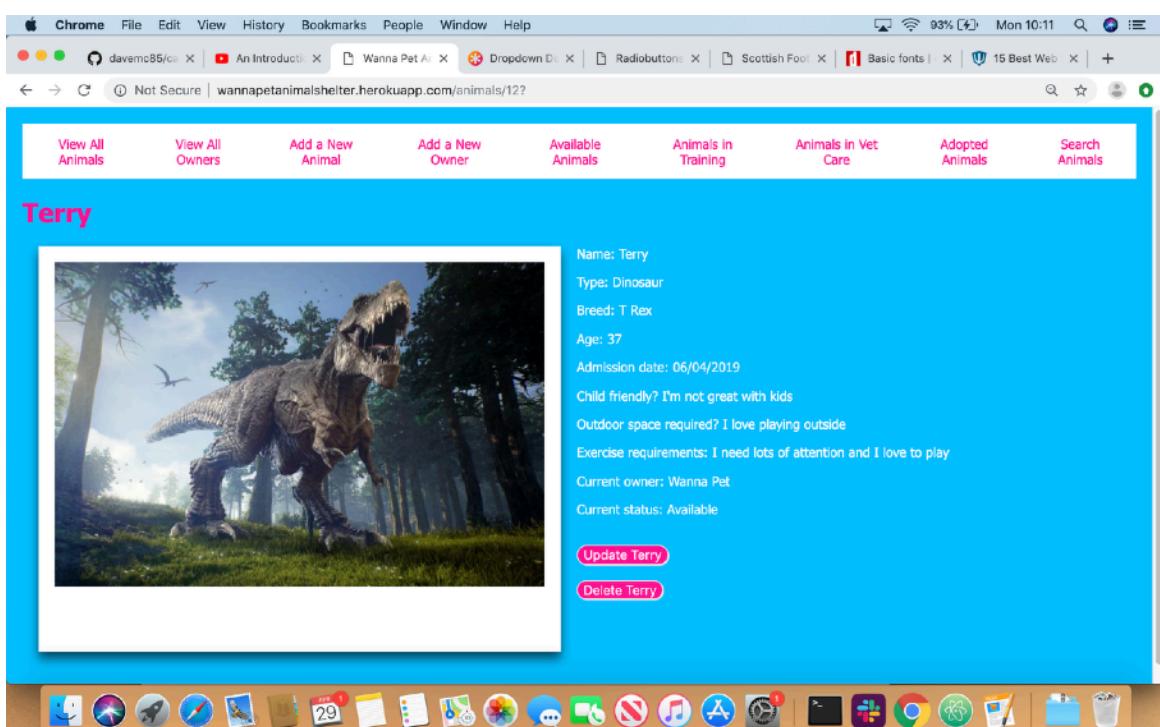
View All Animals View All Owners Add a New Animal Add a New Owner Available Animals Animals in Training Animals in Vet Care Adopted Animals Search Animals

Terry



Name: Terry
 Type: Dinosaur
 Breed: T Rex
 Age: 37
 Admission date: 06/04/2019
 Child friendly? I'm not great with kids
 Outdoor space required? I love playing outside
 Exercise requirements: I need lots of attention and I love to play
 Current owner: Wanna Pet
 Current status: Available

Update Terry Delete Terry



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: Below shows a user updating the details of one of the animals at the shelter. The dog has had its name changed from Archie to Douglas. The result of this can be seen in the bottom image.

Chrome File Edit View History Bookmarks People Window Help

Contributors to ljizzard/code... | Wanna Pet Animal Shelter

Not Secure wannapetanimalshelter.herokuapp.com/animals/1/edit?

View All Animals View All Owners Add a New Animal Add a New Owner Available Animals Animals in Training Animals in Vet Care Adopted Animals Search Animals

Update Archie the Dog

Name:

Admission Date:

Type:

Breed:

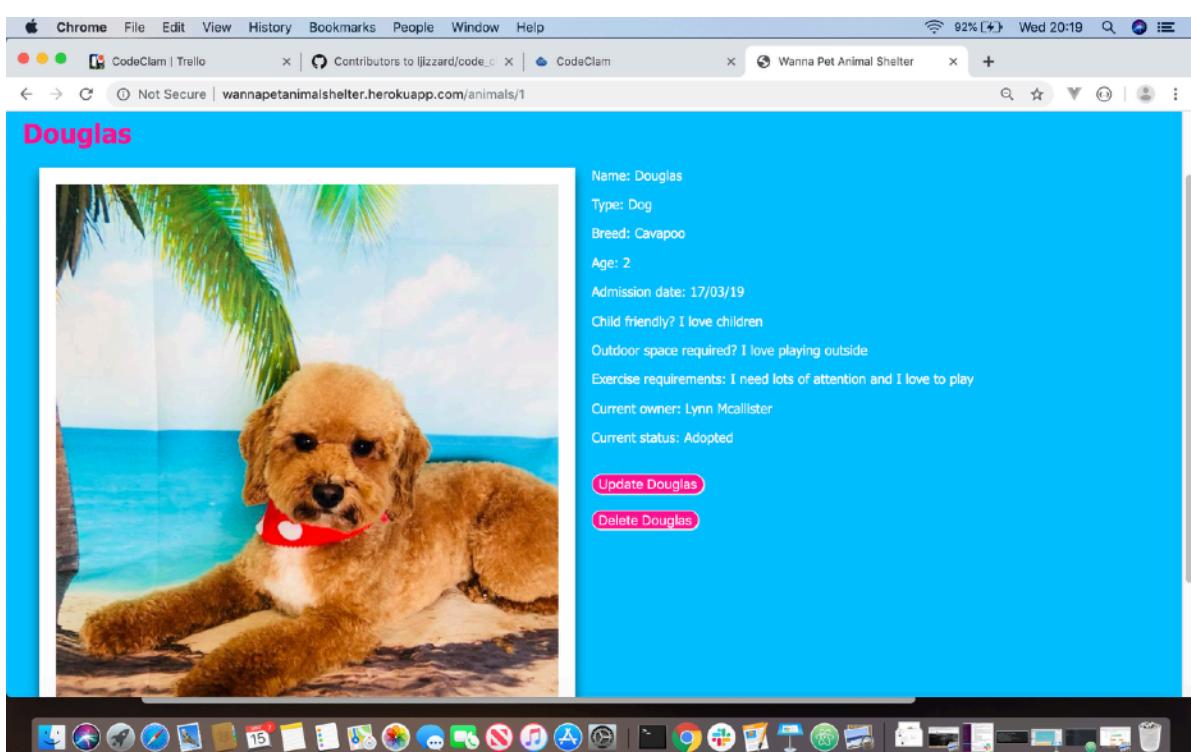
Age:

Is the animal child friendly? Yes No

Does the animal need space outside? Yes No

Does the animal require plenty of exercise? Yes No

Profile picture URL:



Unit	Ref	Evidence
P	P.15	<p>Show the correct output of results and feedback to user. Take a screenshot of:</p> <ul style="list-style-type: none"> * The user requesting information or an action to be performed * The user request being processed correctly and demonstrated in the program
		<p>Description: The images below show how the search functionality worked on the website. The user has searched for dog and the results page has returned the details of the four dogs the shelter has on its books.</p>

Chrome File Edit View History Bookmarks People Window Help

CodeClam | Trelio | Contributors to lizard/code_c | CodeClam | Wanna Pet Animal Shelter | +

Not Secure wannapetanimalshelter.herokuapp.com/animals/search

View All Animals View All Owners Add a New Animal Add a New Owner Available Animals Animals in Training Animals in Vet Care Adopted Animals Search Animals

Search for an animal

What kind of animal are you looking for?

search

Chrome File Edit View History Bookmarks People Window Help

CodeClam | Trelio | Contributors to lizard/code_c | CodeClam | Wanna Pet Animal Shelter | +

Not Secure wannapetanimalshelter.herokuapp.com/animals/search

View All Animals View All Owners Add a New Animal Add a New Owner Available Animals Animals in Training Animals in Vet Care Adopted Animals Search Animals

Animals that meet search criteria

	Name: Archie
Type: Dog	Breed: Cavapoo
Current owner: Lynn McAllister	Current status: Adopted
View Archie's Profile	
	Name: Spot
Type: Dog	Breed: Dalmatian

View All Animals View All Owners Add a New Animal Add a New Owner Available Animals Animals in Training Animals in Vet Care Adopted Animals Search Animals

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: https://github.com/davemc85/animal_shelter

View All Animals View All Owners Add a New Animal Add a New Owner Available Animals Animals in Training Animals in Vet Care Adopted Animals Search Animals

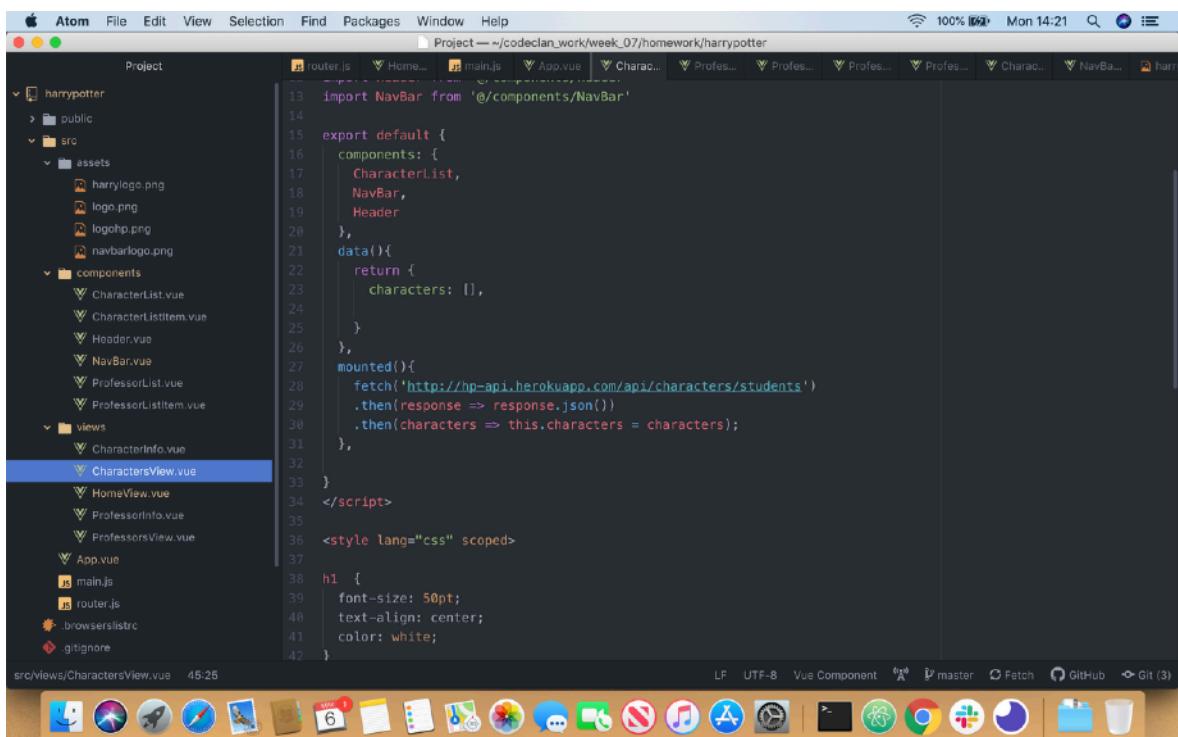
WELCOME TO WANNA PET ANIMAL SHELTER

wannapetanimalshelter.herokuapp.com/animals/new

View All Animals	View All Owners	Add a New Animal	Add a New Owner	Available Animals	Animals in Training	Animals in Vet Care	Adopted Animals	Search Animals
All Our Animals								
	Name: Archie Type: Dog Admission date: 17/03/19 Status: Adopted Current owner: Lynn Mcallister View Archie's Profile							
	Name: Tony Type: Tiger Admission date: 26/01/19 Status: Available Current owner: Wanna Pet View Tony's Profile							

Week 7

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: The API below has details of characters from the Harry Potter franchise. When a character name is selected, more information about that character is displayed.



The screenshot shows the Atom code editor interface. The left sidebar displays a project structure for a Vue.js application named 'harrypotter'. The 'src' folder contains components like 'CharacterList', 'NavBar', 'Header', and 'CharacterInfo'. The 'views' folder contains 'HomeView', 'ProfessorInfo', 'ProfessorsView', and 'CharactersView'. The 'CharactersView.vue' file is currently open in the main editor area. It imports 'NavBar' and defines a component with a 'data()' method that fetches character data from 'http://hp-api.herokuapp.com/api/characters/students'. The code includes a CSS style block for an h1 element. The status bar at the bottom shows the file is 45-26 lines long and is a Vue Component.

```

Project — ~/codeclan_work/week_07/homework/harrypotter
  harrypotter
    public
    src
      assets
        harrylogo.png
        logo.png
        logohp.png
        navbarlogo.png
      components
        CharacterList.vue
        NavBar.vue
        Header.vue
        CharacterListItem.vue
        ProfessorList.vue
        ProfessorListItem.vue
      views
        HomeView.vue
        CharacterInfo.vue
        CharactersView.vue
        ProfessorInfo.vue
        ProfessorsView.vue
    App.vue
    main.js
    router.js
    .browserslistrc
    .gitignore
src/views/CharactersView.vue 45-26
  LF  UTF-8  Vue Component  ↻ master  ⌂ Fetch  GitHub  ⌂ Git (3)

```





Unit	Ref	Evidence
P	P.18	<p>Demonstrate testing in your program. Take screenshots of:</p> <ul style="list-style-type: none"> * Example of test code * The test code failing to pass * Example of the test code once errors have been corrected * The test code passing
		<p>Description: the below example shows a newly written test failing as the function which would allow the test to pass has not been written yet. Once the function has been written so that item C can be purchased, the test now can pass.</p>

```

Terminal Shell Edit View Window Help
test.js README.md
85     machine.coinReturn();
86     assert.strictEqual(0.00, machine.getBalance())
87   });
88 });
89 describe('Balance of money after purchase A', function () {
90   it('Is 0.35 when item a is selected', function () {
91     machine = new VendingMachine();
92     machine.addDollar();
93     machine.itemA();
94     assert.strictEqual(0.35, machine.getBalance())
95   });
96 });
97 describe('Balance of money after purchase B', function () {
98   it('Is 0.00 when item B is selected', function () {
99     machine = new VendingMachine();
100    machine.addDollar();
101    machine.itemB();
102    assert.strictEqual(0.00, machine.getBalance())
103  });
104 });
105 describe('Balance of money after purchase C', function () {
106   it('Is 0.50 when item c is bought', function () {
107     machine = new VendingMachine();
108     machine.addDollar();
109     machine.addDollar();
110     machine.itemC();
111     assert.strictEqual(0.50, machine.getBalance())
112   });
113 });
114 });

test.js 37:28 LF UTF-8 JavaScript master Fetch GitHub Git (1) 83% Fri 20:36

```

```

coding-test.js git:(master)* npm test
> vending-machine@0.0.1 test /Users/user/codeclan_work/week_10/day_03/coding-test.js
> mocha

Vending Machine
  Balance of no money inserted
    ✓ Is zero when initially powered up
  Balance of money after nickel inserted
    ✓ Is 0.05 when a nickel is added
  Balance of money after dime inserted
    ✓ Is 0.10 when a dime is added
  Balance of money after quarter inserted
    ✓ Is 0.25 when a quarter is added
  Balance of money after dollar inserted
    ✓ Is 1.00 when a dollar is added
  Balance of money when coins returned
    ✓ Is 0.00 when coin return is selected
  Balance of money after purchase A
    ✓ Is 0.35 when item a is selected
  Balance of money after purchase B
    ✓ Is 0.00 when item B is selected
  Balance of money after purchase C
    ✓ Is 0.50 when item c is bought

  8 passing (14ms)
  1 failing

  1) Vending Machine
    Balance of money after purchase C
      Is 0.50 when item c is bought:
        TypeError: machine.itemC is not a function
          at Context.<anonymous> (test.js:110:17)

npm ERR! Test failed. See above for more details.

```

```

Terminal Shell Edit View Window Help
test.js README.md
22   return this.balance += dime;
23 };
24 VendingMachine.prototype.addQuarter = function () {
25   return this.balance += quarter;
26 };
27 VendingMachine.prototype.addDollar = function () {
28   return this.balance += dollar;
29 };
30 VendingMachine.prototype.coinReturn = function () {
31   return this.balance = 0;
32 };
33 VendingMachine.prototype.itemA = function () {
34   return this.balance -= a;
35 };
36 VendingMachine.prototype.itemB = function () {
37   return this.balance -= b;
38 };
39 VendingMachine.prototype.itemC = function () {
40   return this.balance -= c;
41 };
42
43
44 let assert = require('assert');
45
46 describe('Vending Machine', function () {
47
48   describe('Balance of no money inserted', function () {
49     it('Is zero when initially powered up', function () {
50       machine = new VendingMachine();
51       assert.strictEqual(0, machine.getBalance())
52     });
53   });
54 });
55 });

test.js 35:3 LF UTF-8 JavaScript master Fetch GitHub Git (1) 83% Fri 20:36

```

```

coding-test.js git:(master)* npm test
> vending-machine@0.0.1 test /Users/user/codeclan_work/week_10/day_03/coding-test.js
> mocha

Vending Machine
  Balance of no money inserted
    ✓ Is zero when initially powered up
  Balance of money after nickel inserted
    ✓ Is 0.05 when a nickel is added
  Balance of money after dime inserted
    ✓ Is 0.10 when a dime is added
  Balance of money after quarter inserted
    ✓ Is 0.25 when a quarter is added
  Balance of money after dollar inserted
    ✓ Is 1.00 when a dollar is added
  Balance of money when coins returned
    ✓ Is 0.00 when coin return is selected
  Balance of money after purchase A
    ✓ Is 0.35 when item a is selected
  Balance of money after purchase B
    ✓ Is 0.00 when item B is selected
  Balance of money after purchase C
    ✓ Is 0.50 when item c is bought

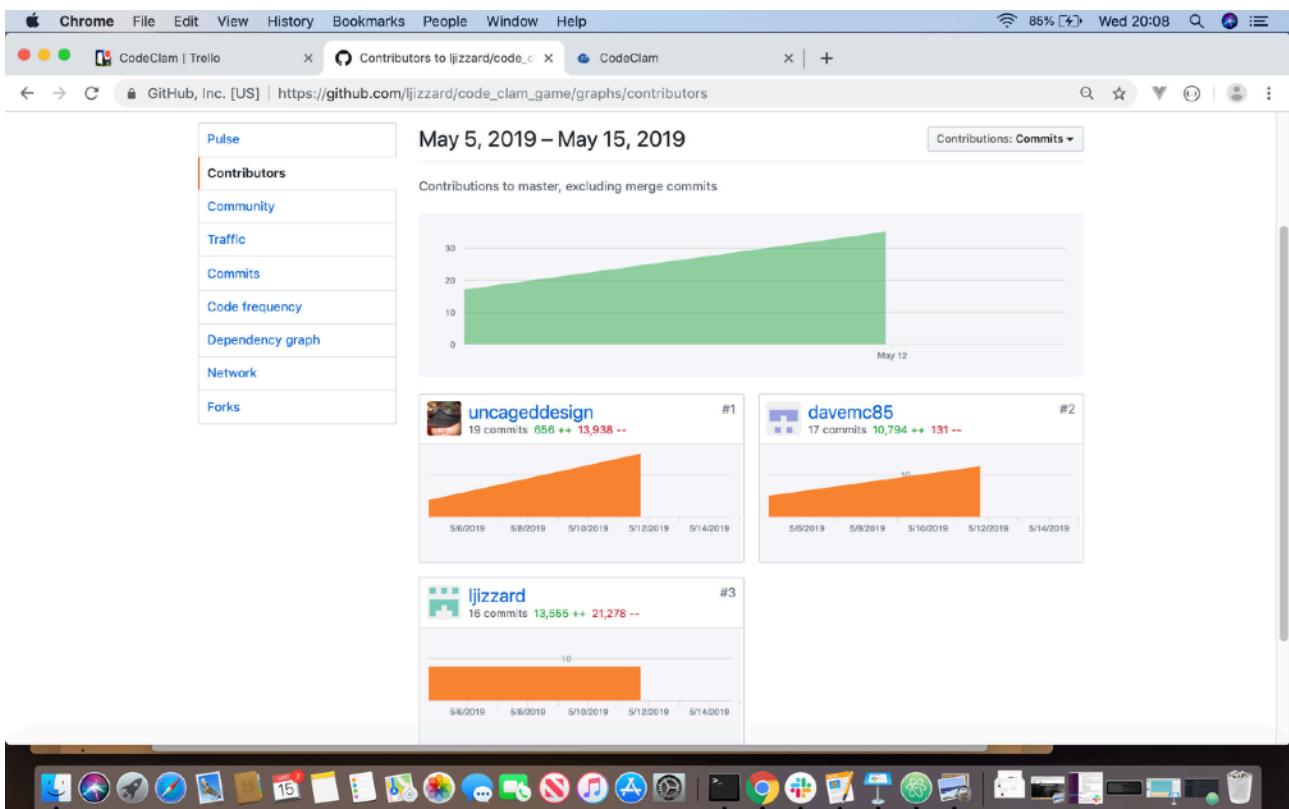
  9 passing (13ms)

npm update check failed
Try running with sudo or get access to the local update config store via
sudo chown -R $USER:$CDN -n $USER /Users/$USER/.config

```

Week 9

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: This screenshot below shows my group for the project. uncageddesign is Mhairi, Ijizzard is Lindsey and davemc85 is myself



Unit	Ref	Evidence
P	P.12	Take screenshots or photos of your planning and the different stages of development to show changes.
		Description: Here are some of the images from our planning for the Codeclam project. The initial wireframe diagrams show our initial layout idea and the bottom image shows how the final page looked. The user needs shows a sample of our users and what they would expect from the game.

CodeClam

Jellyfish

Pee on yourself, etc etc...

CodeClam

Yay! Woo!

Play Another Game

Starfish

Legs for days.

User needs

As a...	I want to...	So that...
Parent with a young child	keep them entertained with something educational	I can get some of my own tasks done.
Young human	do something that holds my attention	my parent doesn't go insane
a young child with learning difficulties	play a game on my own/ independently	my confidence can grow
Early learning teacher	have a game that is educational fun and helps with memory skills	my pupils are engaged using a safe site.
33 year old man	find satisfaction of matching creatures in a fun manner	I can catch them all. (and satisfy my kleptomania)
Carer for the elderly	keep those with memory issues engaged with simple, fun activities	We can help them regenerate memory pathways.

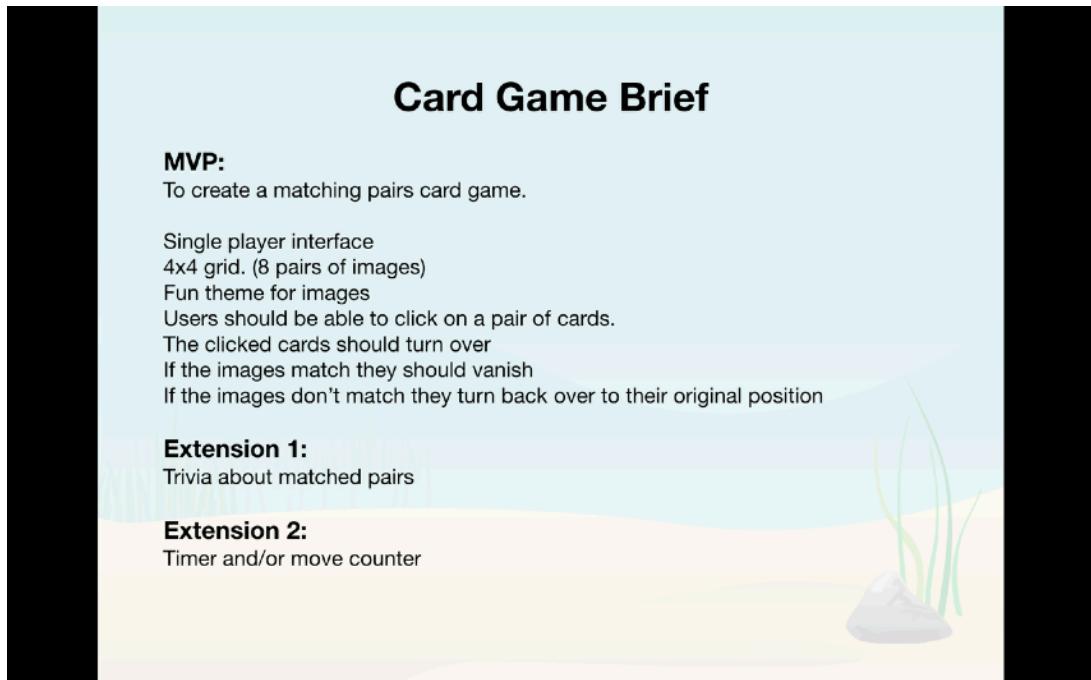
CodeClam

Instructions

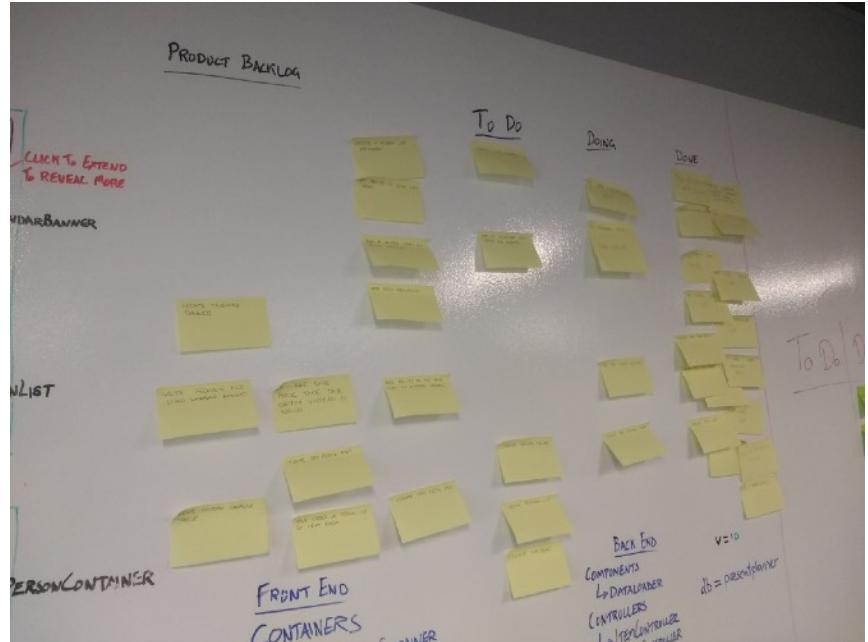
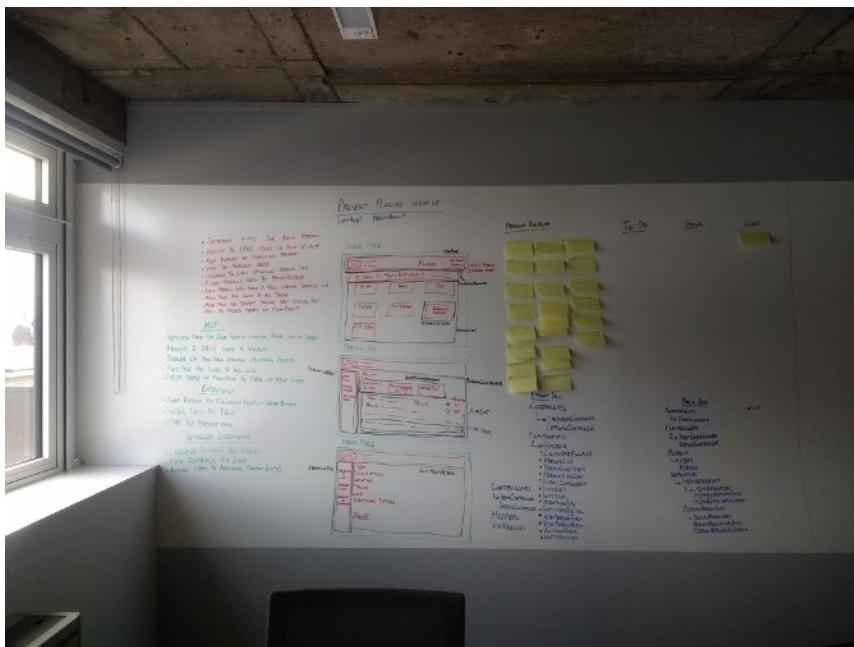
Spin the clams two cards at a time. If they match we'll add them to your catch of the day. See if you can catch them all!

DID YOU KNOW?... Jellyfish were around before the Dinosaurs.

Unit	Ref	Evidence
P	P.2	Take a screenshot of the project brief from your group project.
		Description: The initial brief was to create a card or dice game so we had to create our own MVP and extensions. Below shows our final decision on what we should try and achieve in the week.



Unit	Ref	Evidence
P	P.3	Provide a screenshot of the planning you completed during your group project, e.g. Trello MOSCOW board.
		Description: Below are photos of our agile board. You can see our initial ideas, our brief with MVP and extensions, wireframe diagrams, file structure, project backlog and job list. This was updated throughout the process to keep up to date with any changes.



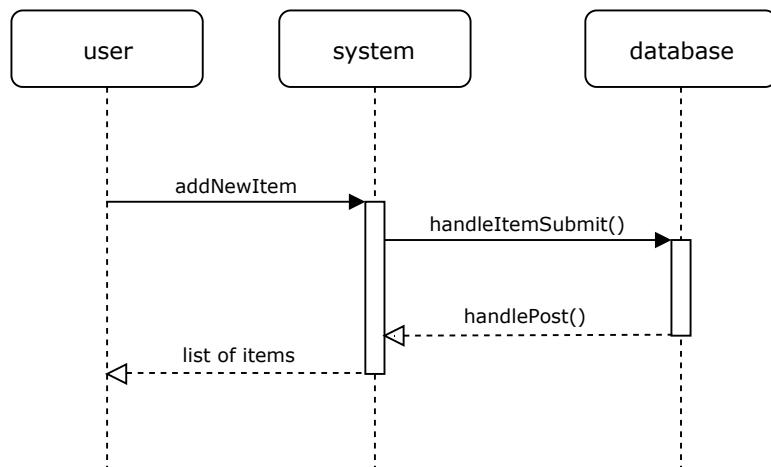
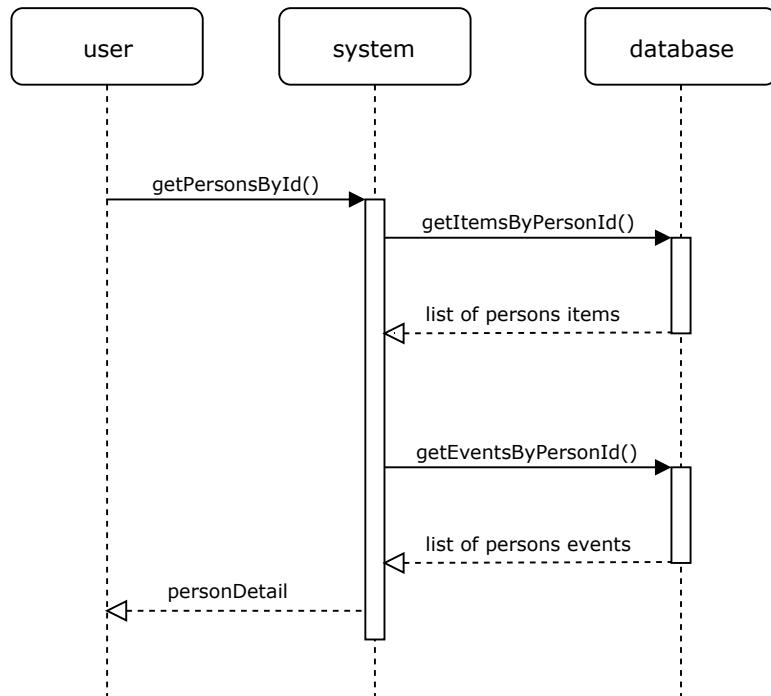
Unit	Ref	Evidence
P	P.4	Write an acceptance criteria and test plan.
		Description: Here is our group acceptance criteria and test plan. Each row has an element that fulfilled part of our Codeclam brief. As each part passed it meant our game was a step closer to completion.

Acceptance Criteria

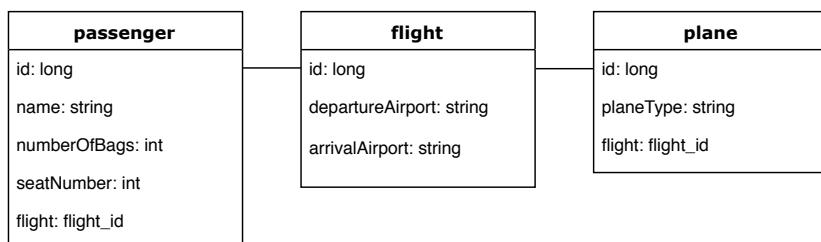
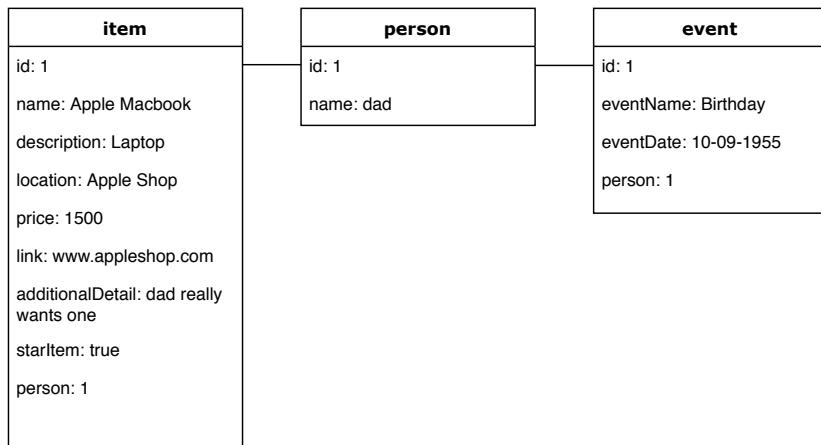
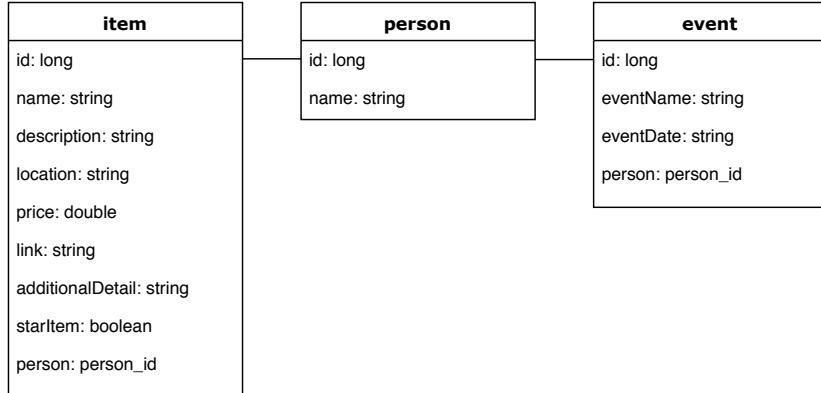
Stage: Initial Planning

Acceptance Criteria	Expected Result/Output	Pass / Fail
Cards can be displayed in grid	Grid of 4 x 4	Pass
Card can be selected	Card changes class on click	Pass
Only two cards can be selected	Third click does nothing	Pass
If cards match they disappear	Set opacity to 0	Pass
Cards reset if no match	Cards turn back over	Pass
Random fact generated	On page refresh	Pass
Game resets on completion	Auto refresh after timeout	Pass

Unit	Ref	Evidence
P	P.7	<p>Produce two system interaction diagrams (sequence and/or collaboration diagrams).</p> <p>Description: diagram1: this shows the interaction that occurs when the user clicks on a specific person. Behind the scenes the system pulls all the items and all the events assigned to that person and displays them in lists.</p> <p>Diagram 2: this is a similar sequence diagram. It shows the interaction that would occur if the user wanted to add a new item to the list.</p>



Unit	Ref	Evidence
P	P.8	Produce two object diagrams.
		<p>Description: The object diagrams below show examples of the data structures for each class. The top example is for a present planner wish list where a person is assigned to an item and an event in a one to many relationship (as in one person can have many items/events).</p> <p>The bottom example is of a similar set up but uses the airport example where a flight has a plane and many passengers.</p>

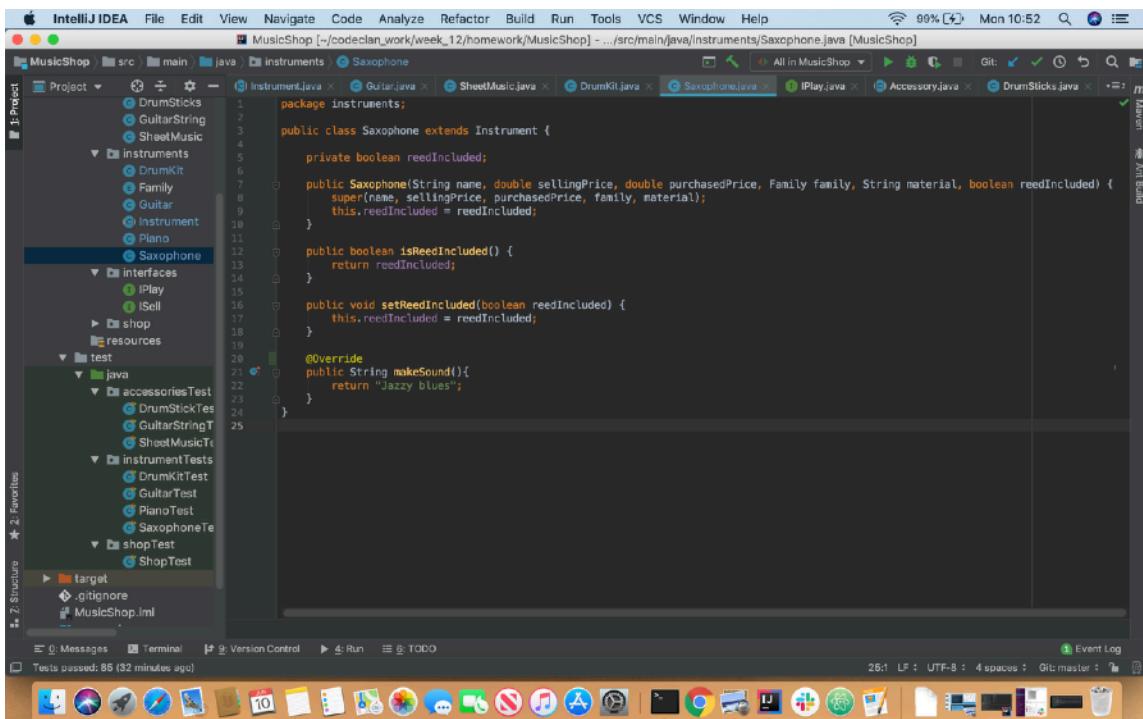


Unit	Ref	Evidence
P	P.17	Produce a bug tracking report
		Description: Here is sample of the bugs we encountered during the last group project. The vast majority of errors were syntax and spelling mistakes. The selection below shows a few of the more challenging issues.

Bug/Error	Solution	Date
java.time.format.DateTimeParseException: Text could not be parsed at index 3	Changed the date format to match the input style used in db.	26-06-2019
Parsing error: Identifier 'event' has already been declared	We had declared 'event' twice. One was removed	25-06-2019
Uncaught TypeError: Cannot read property 'persons' of undefined	Had to use projections to assign an id to a person in the backend when you were looking at the "/events/1/persons"	26-06-2019
Star item function not working when new item is added	Changed event.target.value to event.target.checked as it has a checked function	26-06-2019

Week 12

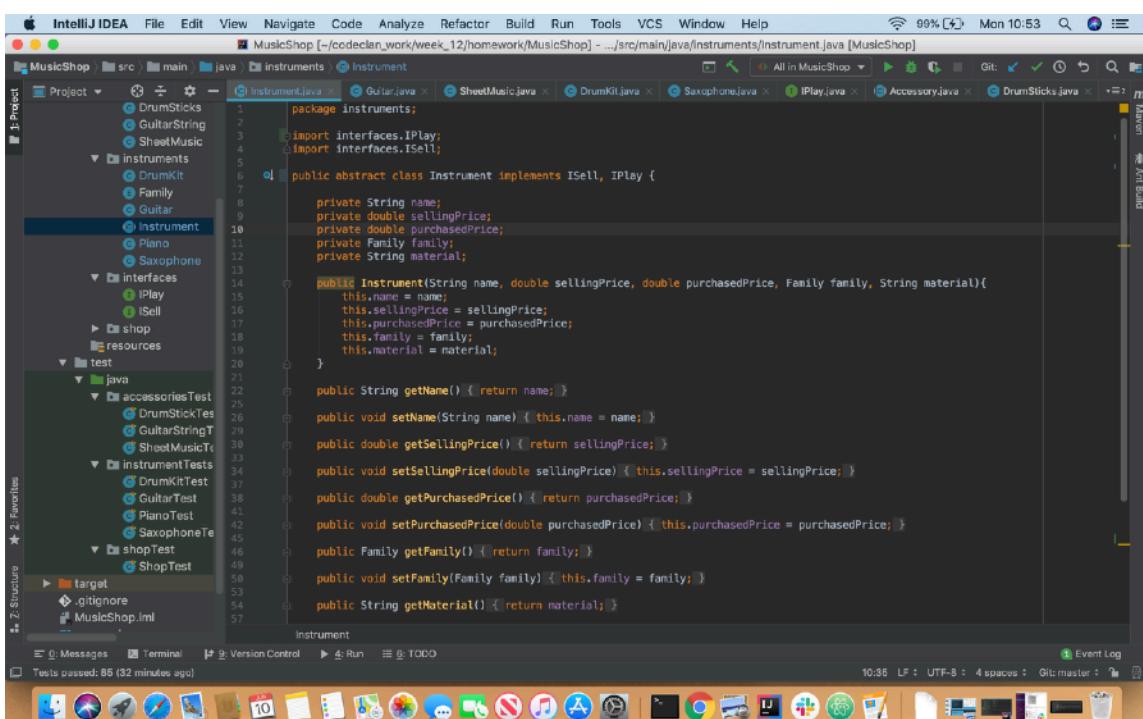
Unit	Ref	Evidence
I&T	I.T.7	The use of Polymorphism in a program and what it is doing.
		<p>Description: Inheritance lets us inherit attributes and methods from another class. Polymorphism uses those methods to perform different tasks. The superclass Instrument here has a makeSound() method. The subclasses of Instrument (guitar, piano etc) all have their own implementation of makeSound() when it is called.</p>



```

package instruments;
public class Saxophone extends Instrument {
    private boolean reedIncluded;
    public Saxophone(String name, double sellingPrice, double purchasedPrice, Family family, String material, boolean reedIncluded) {
        super(name, sellingPrice, purchasedPrice, family, material);
        this.reedIncluded = reedIncluded;
    }
    public boolean isReedIncluded() {
        return reedIncluded;
    }
    public void setReedIncluded(boolean reedIncluded) {
        this.reedIncluded = reedIncluded;
    }
    @Override
    public String makeSound(){
        return "Jazzy blues";
    }
}

```



```

package instruments;
import interfaces.IPlay;
import interfaces.ISell;
public abstract class Instrument implements ISell, IPlay {
    private String name;
    private double sellingPrice;
    private double purchasedPrice;
    private Family family;
    private String material;
    public Instrument(String name, double sellingPrice, double purchasedPrice, Family family, String material){
        this.name = name;
        this.sellingPrice = sellingPrice;
        this.purchasedPrice = purchasedPrice;
        this.family = family;
        this.material = material;
    }
    public String getName() { return name; }
    public void setName(String name) { this.name = name; }
    public double getSellngPrice() { return sellingPrice; }
    public void setSellngPrice(double sellingPrice) { this.sellingPrice = sellingPrice; }
    public double getPurchasedPrice() { return purchasedPrice; }
    public void setPurchasedPrice(double purchasedPrice) { this.purchasedPrice = purchasedPrice; }
    public Family getFamily() { return family; }
    public void setFamily(Family family) { this.family = family; }
    public String getMaterial() { return material; }
}

```

IntelliJ IDEA File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

MusicShop [-codeclan_work/week_12/homework/MusicShop] - ...src/main/java/instruments/Instrument.java [MusicShop]

Project

src main java Instruments Instrument

Instrument.java

```
public class Instrument {
    private String name;
    private double sellingPrice;
    private double purchasedPrice;
    private Family family;
    private String material;

    public Instrument(String name, double sellingPrice, double purchasedPrice, Family family, String material) {
        this.name = name;
        this.sellingPrice = sellingPrice;
        this.purchasedPrice = purchasedPrice;
        this.family = family;
        this.material = material;
    }

    public String getName() { return name; }

    public void setName(String name) { this.name = name; }

    public double getSellngPrice() { return sellingPrice; }

    public void setSellngPrice(double sellingPrice) { this.sellingPrice = sellingPrice; }

    public double getPurchasedPrice() { return purchasedPrice; }

    public void setPurchasedPrice(double purchasedPrice) { this.purchasedPrice = purchasedPrice; }

    public Family getFamily() { return family; }

    public void setFamily(Family family) { this.family = family; }

    public String getMaterial() { return material; }

    public void setMaterial(String material) { this.material = material; }

    public double calculateMarkup() { return this.sellingPrice - this.purchasedPrice; }

    public String makeSound(){
        return "Play that funky music";
    }
}
```

Instrument

Messages Terminal Version Control Run TODO

Tests passed: 85 (32 minutes ago)

target .gitignore MusicShop.iml

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

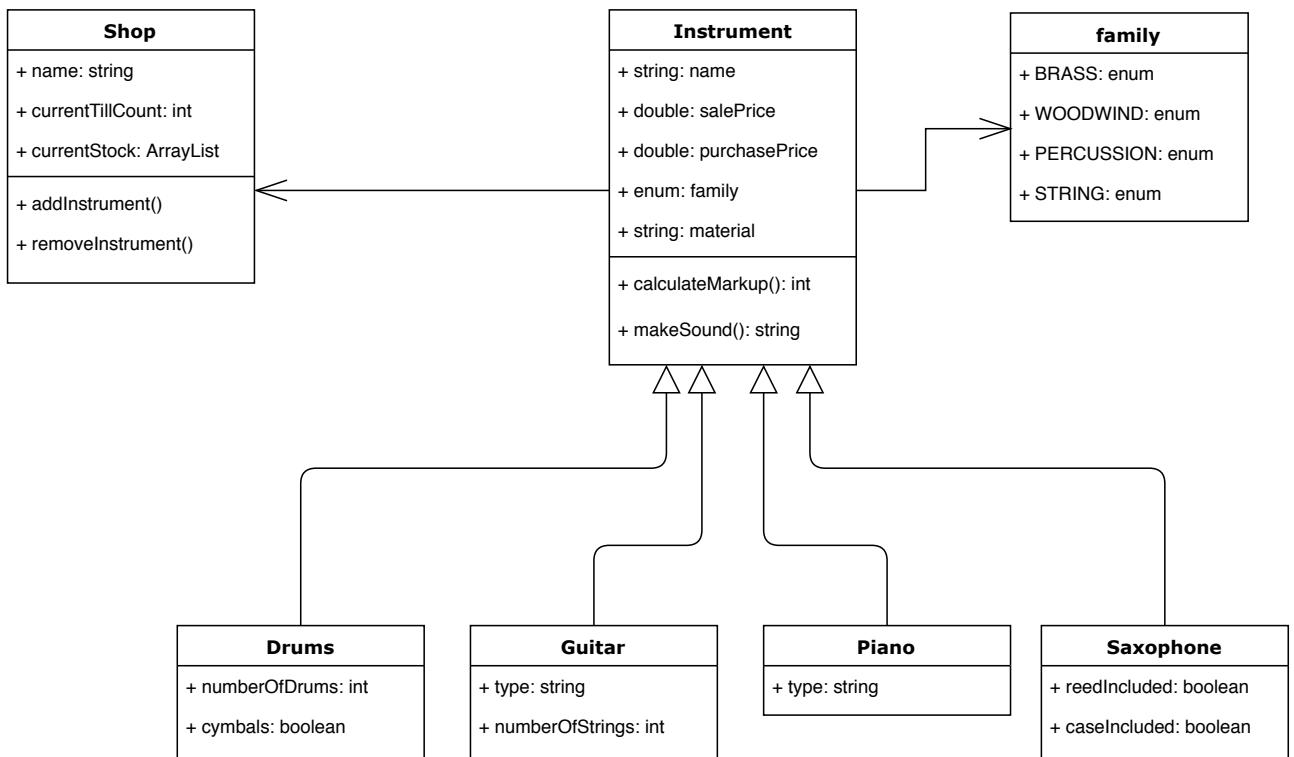
All In MusicShop Git Art Build

100% 10:53

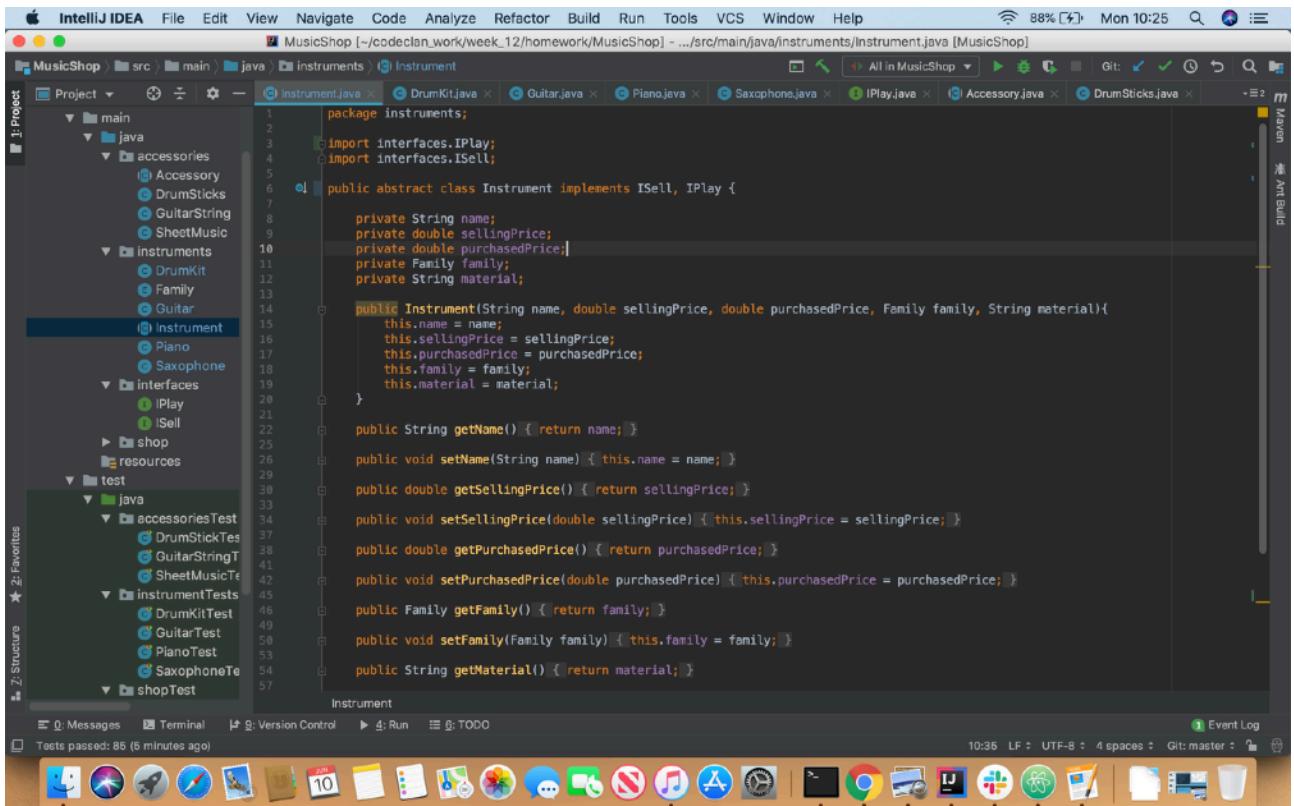
1 Project 2 Favorites 2 Structure

52:6 LF UTF-8 4 spaces Git: master

Unit	Ref	Evidence
A&D	A.D.5	An Inheritance Diagram
		<p>Description: This is an inheritance diagram for a music shop. The 4 different instruments (Drums, guitar, piano and saxophone) all inherit properties and methods from the instrument class which is abstract. The Shop class has the functions auto add and remove instruments in an ArrayList for the current stock.</p>



Unit	Ref	Evidence
I&T	I.T.1	The use of Encapsulation in a program and what it is doing.
		<p>Description: The code below is for an instrument abstract class. The instance variables are private so that they cannot be accessed directly from outside the class. You can only set and get values of these variables through the methods of the class using getter and setters.</p>



```

1 package instruments;
2
3 import interfaces.IPlay;
4 import interfaces.ISell;
5
6 public abstract class Instrument implements ISell, IPlay {
7
8     private String name;
9     private double sellingPrice;
10    private double purchasedPrice;
11    private Family family;
12    private String material;
13
14    public Instrument(String name, double sellingPrice, double purchasedPrice, Family family, String material){
15        this.name = name;
16        this.sellingPrice = sellingPrice;
17        this.purchasedPrice = purchasedPrice;
18        this.family = family;
19        this.material = material;
20    }
21
22    public String getName() { return name; }
23
24    public void setName(String name) { this.name = name; }
25
26    public double getSellPrice() { return sellingPrice; }
27
28    public void setSellPrice(double sellingPrice) { this.sellingPrice = sellingPrice; }
29
30    public double getPurchasedPrice() { return purchasedPrice; }
31
32    public void setPurchasedPrice(double purchasedPrice) { this.purchasedPrice = purchasedPrice; }
33
34    public Family getFamily() { return family; }
35
36    public void setFamily(Family family) { this.family = family; }
37
38    public String getMaterial() { return material; }
39
40}

```

Unit	Ref	Evidence
I&T	I.T.2	<p>Take a screenshot of the use of Inheritance in a program. Take screenshots of:</p> <ul style="list-style-type: none"> *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.
		<p>Description: This example uses the music shop. The DrumSticks class inherits from the Accessory class. The calculateMarkup() method is shown calculating the markup of the DrumSticks with the result shown for the drum sticks in the last image.</p>

```

package accessories;
import interfaces.ISell;

public abstract class Accessory implements ISell {
    private String name;
    private double sellingPrice;
    private double purchasedPrice;

    public Accessory(String name, double sellingPrice, double purchasedPrice) {
        this.name = name;
        this.sellingPrice = sellingPrice;
        this.purchasedPrice = purchasedPrice;
    }

    public String getName() { return name; }

    public void setName(String name) { this.name = name; }

    public double getSellPrice() { return sellingPrice; }

    public void setSellPrice(double sellingPrice) { this.sellingPrice = sellingPrice; }

    public double getPurPrice() { return purchasedPrice; }

    public void setPurPrice(double purchasedPrice) { this.purchasedPrice = purchasedPrice; }

    public double calculateMarkup() { return sellingPrice - purchasedPrice; }
}

```

```

package accessories;
public class DrumSticks extends Accessory {
    public DrumSticks(String name, double sellingPrice, double purchasedPrice) {
        super(name, sellingPrice, purchasedPrice);
    }
}

```

IntelliJ IDEA 2022.3

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

MusicShop [~/codeclan_work/week_12/homework/MusicShop] - .../src/test/java/accessoriesTest/DrumStickTest.java [MusicShop]

Project

main

- java
- accessories
 - Accessory
 - DrumSticks
 - GuitarString
 - SheetMusic
- instruments
 - DrumKit
 - Family
 - Guitar
 - Instrument
 - Piano
 - Saxophone
- interfaces
 - IPlay
 - ISell
- shop
- resources

test

- java
 - accessoriesTest
 - DrumStickTest
 - GuitarStringTest
 - SheetMusicTest
 - instrumentTests
 - DrumKitTest
 - GuitarTest
 - PianoTest
 - SaxophoneTest
 - shopTest

DrumStickTest.java

```
package accessoriesTest;

import accessories.DrumSticks;
import org.junit.Before;
import org.junit.Test;

import static org.junit.Assert.assertEquals;

public class DrumStickTest {

    private DrumSticks drumSticks;

    @Before
    public void setUp(){
        drumSticks = new DrumSticks("Basic", 10, 5);
    }

    @Test
    public void canGetName(){
        assertEquals("Basic", drumSticks.getName());
    }

    @Test
    public void canSetName(){
        drumSticks.setName("Fancy");
        assertEquals("Fancy", drumSticks.getName());
    }

    @Test
    public void canGetSellingPrice() {
        assertEquals(10, drumSticks.getSellingPrice(), delta: 0.00);
    }

    @Test
    public void canSetSellingPrice(){
        drumSticks.setSellingPrice(15);
        assertEquals(15, drumSticks.getSellingPrice(), delta: 0.00);
    }

    @Test
    public void canGetPurchasedPrice() {
        assertEquals(5, drumSticks.getPurchasePrice(), delta: 0.00);
    }

    @Test
    public void canGetName() {
        drumStickTest.canGetName();
    }
}
```

Messages Terminal Version Control Run TODO Event Log

Tests passed: 85 (2 minutes ago)

19:30 LF: UTF-8 4 spaces Git: master

IntelliJ IDEA 2022.3

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

MusicShop [~/codeclan_work/week_12/homework/MusicShop] - .../src/test/java/accessoriesTest/DrumStickTest.java [MusicShop]

Project

main

- java
- accessories
 - Accessory
 - DrumSticks
 - GuitarString
 - SheetMusic
- instruments
 - DrumKit
 - Family
 - Guitar
 - Instrument
 - Piano
 - Saxophone
- interfaces
 - IPlay
 - ISell
- shop
- resources

test

- java
 - accessoriesTest
 - DrumStickTest
 - GuitarStringTest
 - SheetMusicTest
 - instrumentTests
 - DrumKitTest
 - GuitarTest
 - PianoTest
 - SaxophoneTest
 - shopTest
- target
- .gitignore
- MusicShop.iml

DrumStickTest.java

```
package accessoriesTest;

import accessories.DrumSticks;
import org.junit.Before;
import org.junit.Test;

import static org.junit.Assert.assertEquals;

public class DrumStickTest {

    @Test
    public void canSetName(){
        drumSticks.setName("Fancy");
        assertEquals("Fancy", drumSticks.getName());
    }

    @Test
    public void canGetSellingPrice(){
        assertEquals(10, drumSticks.getSellingPrice(), delta: 0.00);
    }

    @Test
    public void canSetSellingPrice(){
        drumSticks.setSellingPrice(15);
        assertEquals(15, drumSticks.getSellingPrice(), delta: 0.00);
    }

    @Test
    public void canGetPurchasedPrice(){
        assertEquals(5, drumSticks.getPurchasePrice(), delta: 0.00);
    }

    @Test
    public void canSetPurchasedPrice(){
        drumSticks.setPurchasePrice(10);
        assertEquals(10, drumSticks.getPurchasePrice(), delta: 0.00);
    }

    @Test
    public void canGetCalculateMarkup(){
        assertEquals(5, drumSticks.calculateMarkup(), delta: 0.00);
    }
}
```

Messages Terminal Version Control Run TODO Event Log

Tests passed: 85 (2 minutes ago)

19:30 LF: UTF-8 4 spaces Git: master

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.
		<p>Description: The first image shows a change calculator for a vending machine. The function takes in a number input (change) and works out the best way to issue that amount of change using the least amount of coins.</p> <p>The second image is of a bubble sort used to order the passengers by the seat number they were randomly assigned when a ticket was purchased. Both of these algorithms, although complex to look at, provide a very clear process of working out the correct result.</p>

```
// this method is used to calculate what coins would be returned
// for a given amount of change

public void workOutChange(int change){
    ArrayList<Coin> returnedCoins = new ArrayList<Coin>();
    if(change >= 200){ // if change is more than 200 a two pound coin can be issued
        this.returnedCoins.add(twoPound);
        change -= 200;
    } if(change >= 100){ // if change is more than 100 a pound coin can be issued
        this.returnedCoins.add(pound);
        change -= 100;
    } if (change >= 50){ // if change is more than 50 a 50p coin can be issued
        this.returnedCoins.add(fifty);
        change -= 50;
    } if (change >= 20){// if change is more than 20 a 20p coin can be issued
        this.returnedCoins.add(twenty);
        change -= 20;
        if (change >= 20){ // if change is still more than 20 a 20p coin can be issued ie 40p change
            this.returnedCoins.add(twenty);
            change -= 20;
        }
    } if (change >= 10){ // if change is more than 10 a 10p coin can be issued
        this.returnedCoins.add(ten);
        change -= 10;
    } if (change >= 5){ //// if change is more than 5 a 5p coin can be issued
        this.returnedCoins.add(five);
        change -= 5;
    }
}
```

```
public ArrayList<Passenger> sortPassengersBySeatNumber(Flight flight) {
    ArrayList<Passenger> sortedPassengers = flight.getPassengers();
    Passenger temp;
    for (int i = sortedPassengers.size() -1; i > 0; i--){
        boolean isSorted = true;

        for (int j = 0; j < i; j++){
            if (sortedPassengers.get(j + 1).getSeatNumber() < sortedPassengers.get(j).getSeatNumber()){
                isSorted = false;
                temp = sortedPassengers.get(j + 1);
                sortedPassengers.set(j + 1, sortedPassengers.get(j));
                sortedPassengers.set(j, temp);
            }
        }
        if (isSorted){
            break;
        }
    }
    return sortedPassengers;
}
```

