

I.T.5. Use of an array:

The screenshot shows a Mac OS X desktop environment. In the foreground, a Terminal window is open with the title "5-use-of-array.rb". The code inside the terminal is as follows:

```
5-use-of-array.rb
1 stops = [ "Croy", "Cumbernauld", "Falkirk High", "Linlithgow", "Livingston", "Haymarket" ]
2
3 def look_for_stop(station, array)
4   for stop in array
5     if stop == station
6       return true
7     end
8   end
9   return false
10 end
11
12 p look_for_stop('Edinburgh', stops)
13
14 p look_for_stop('Cumbernauld', stops)
15
```

Below the terminal, a Git commit history window is visible, showing two commits:

- ..z_start_point
- ..k/PDA/I-and-T
- I-and-T git:(master) ✘ ruby 5-use-of-array.rb
- false
- true
- I-and-T git:(master) ✘

The system tray at the bottom of the screen displays various application icons, including Finder, Mail, Safari, and others.

- Line 1 shows an array of strings (assigned to the variable name 'stops').
- Lines 3-10 show a function written for the purpose of determining whether a given station exists within the array 'stops'.
- It goes through each item in the array and if that item matches the station being searched for (passed into the function as a parameter on line 3), the function returns true.
- If it goes through the entire array and does not find the given station, it returns false.

I.T.6. Use of a hash:

```
6-use-of-hash.rb
1 united_kingdom = [
2   {
3     name: "Scotland",
4     population: 5295000,
5     capital: "Edinburgh"
6   },
7   {
8     name: "Wales",
9     population: 3063000,
10    capital: "Swansea"
11  },
12  {
13    name: "England",
14    population: 53010000,
15    capital: "London"
16  }
17 ]
18 def find_total_population(country_array)
19   total = 0
20   for country in country_array
21     total += country[:population]
22   end
23   return total
24 end
25 p find_total_population(united_kingdom)

6-use-of-hash.rb 25:3
```

..z_start_point ..k/PDA/I-and-T
I-and-T git:(master) ✘ ruby 6-use-of-hash.rb
61368000
I-and-T git:(master) ✘

- Lines 1-17 show an array of hashes, each hash showing a country with 3 keys (name, population and capital).
- Lines 18-24 show a function written for the purpose of calculating the total population of countries in a given array of hashes.
- It starts by setting total to 0, then it goes through each item in the array and accesses the population by the 'population' key, each time adding this number to the total.
- Once it has gone through each item it returns the total.

I.T.3. Searching for data:

```

album.rb -- ~/Documents/codeclan_work/project1
..work/project1 ruby seeds.rb -- ruby seeds.rb -- 80x24
ruby

DROP TABLE
DROP TABLE
CREATE TABLE
CREATE TABLE
↳ project1 git:(master) ✘ ruby seeds.rb

From: /Users/user/Documents/codeclan_work/project1/seeds.rb @ line 333 :
  328:   'artist_id' => artist1.id
  329: )
  330: album34.check_db()
  331:
  332: binding.pry
=> 333: nil

[1] pry(main)> Album.find(12)
=> #<Album:0x007fb34d468c10
@artist_id=8,
@buying_cost=4.0,
@id=12,
@price=8.0,
@stock=0,
@title="Pet Sounds">
[2] pry(main)>

models/album.rb* 183:1

```

- The code above shows the function I wrote for my project that was used to find a particular album from a database using only its ID, which it takes in as a parameter (line 184). The function is called on the Album class.
- I was writing in Ruby and inserted SQL to run my search (lines 186-7).
- Line 189 calls a helper method I wrote in a separate document to process the SQL correctly.
- As you can see from the console, my function returned the expected album (the album object with id=12).

I.T.4. Sorting data:

The screenshot shows a Mac OS X desktop environment. In the top-left corner is a Terminal window titled "Project — ~/Documents/codeclan_work/project1". The terminal displays Ruby code from the file "models/artist.rb". The code defines a function "albums" that generates a SQL query to find all albums for a given artist ID. It includes helper methods like "SET", "UPDATE", and "SELECT * FROM albums WHERE artist_id = \$1 ORDER BY title". A portion of the code is highlighted in yellow. In the top-right corner, there's a small floating terminal window titled "ruby" showing the output of running the script, which lists several albums by Paul Simon. The bottom of the screen features the Dock with various application icons.

```

Project
  - Gemfile
  - Projfile
  - inventory...
  - seeds.rb
  - album.rb
  - index.erb
  - sql_runner...
  - artist.ri
  - edit.erb
  - structure...
  - Gemfile.lo...
  - config

43  # ORDER BY name ASC"
44  # result = SqlRunner.run(
45  # return result.map { |a|
46  # end

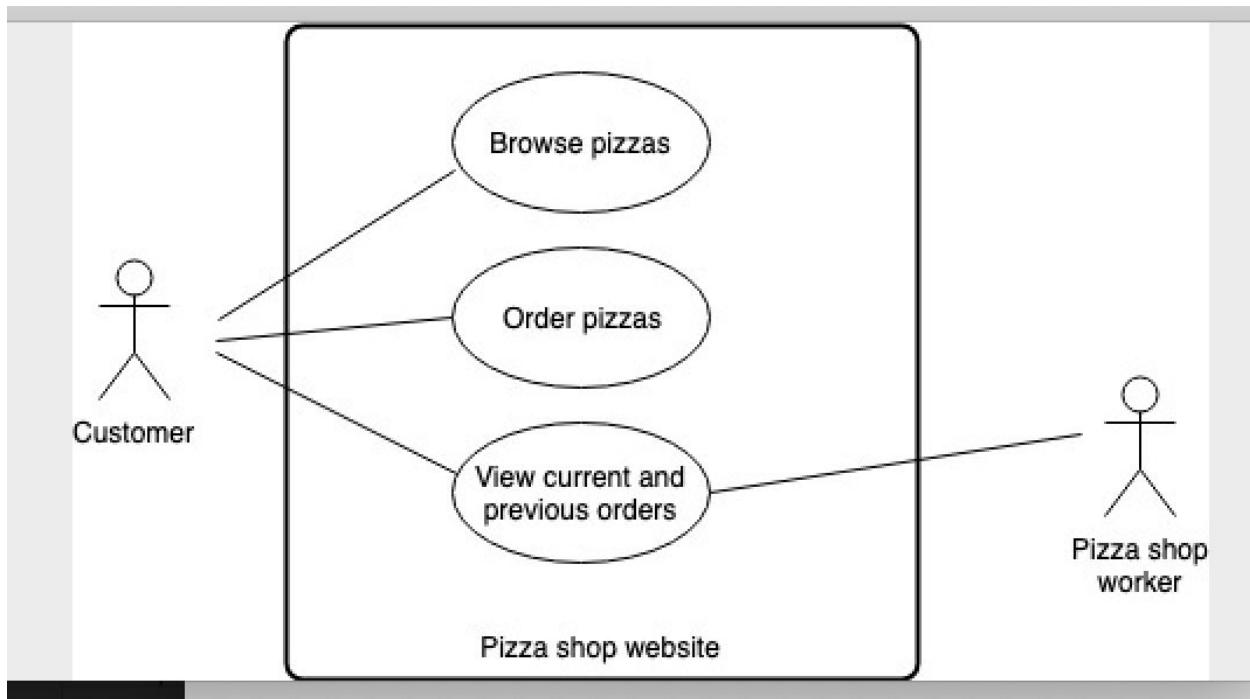
47
48  # def update()
49  #   sql =
50  #   "UPDATE artists
51  #   SET (name, genre)
52  #   = ($1, $2)
53  #   WHERE id = $3"
54  #   values = [@name, @genre]
55  #   SqlRunner.run(sql, values)
56  # end

57
58  def albums()
59    sql =
60    "SELECT * FROM albums
61    WHERE albums.artist_id = $1
62    ORDER BY title"
63    values = [@id]
64    result = SqlRunner.run(sql, values)
65    return result.map { |album| Album.new(album) }
66  end

```

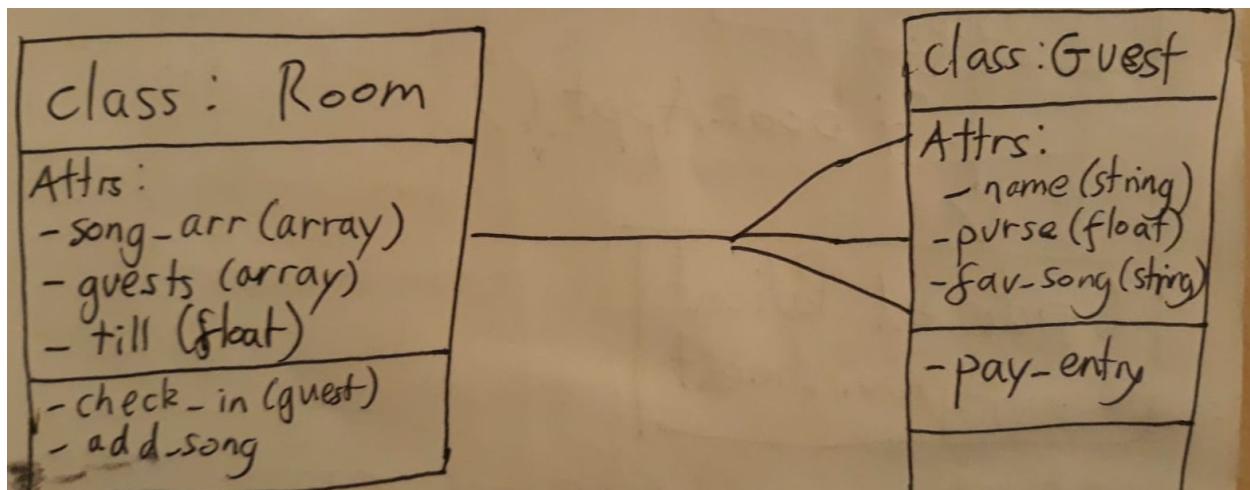
- The code above shows the function I wrote for my project that was used to find all the albums in a database that are listed under a particular artist. The function is called on an Artist object.
- The artist is identified by the artist's id, which I joined to the album table via the column 'artist_id'.
- As stated above, I was writing in Ruby and inserted SQL to run my search (lines 60-62). Line 62 sorts the selected data by the 'title' column, returning the data in alphabetical order.
- Line 64 calls the same helper method I referred to in I.T.3.
- As you can see from the console, my function returns all albums by Paul Simon.

A.D.1 Use case diagram:

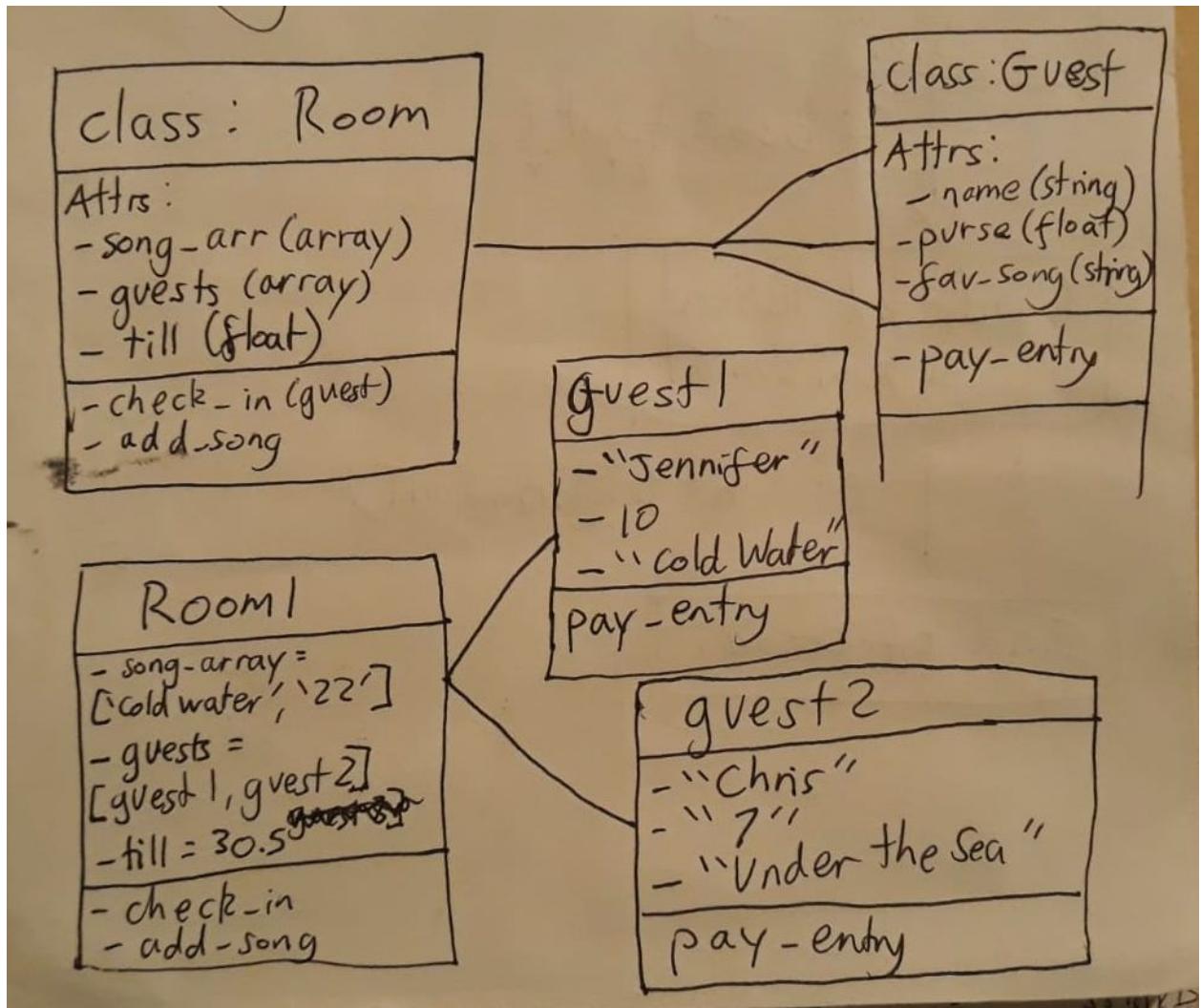


Above is a use case diagram I drew to help me visualise how a pizza shop website might be used by both customers and staff to process pizza orders. A customer can browse the menu and place orders, as well as review their order and previous orders they have made. The staff can obviously also view current orders, as well as previous orders (possibly for marketing purposes).

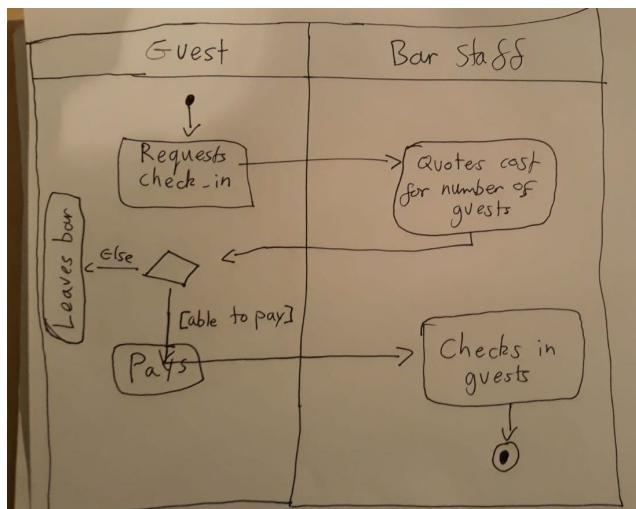
A.D.2. Class diagram:



A.D.3. Object diagram:



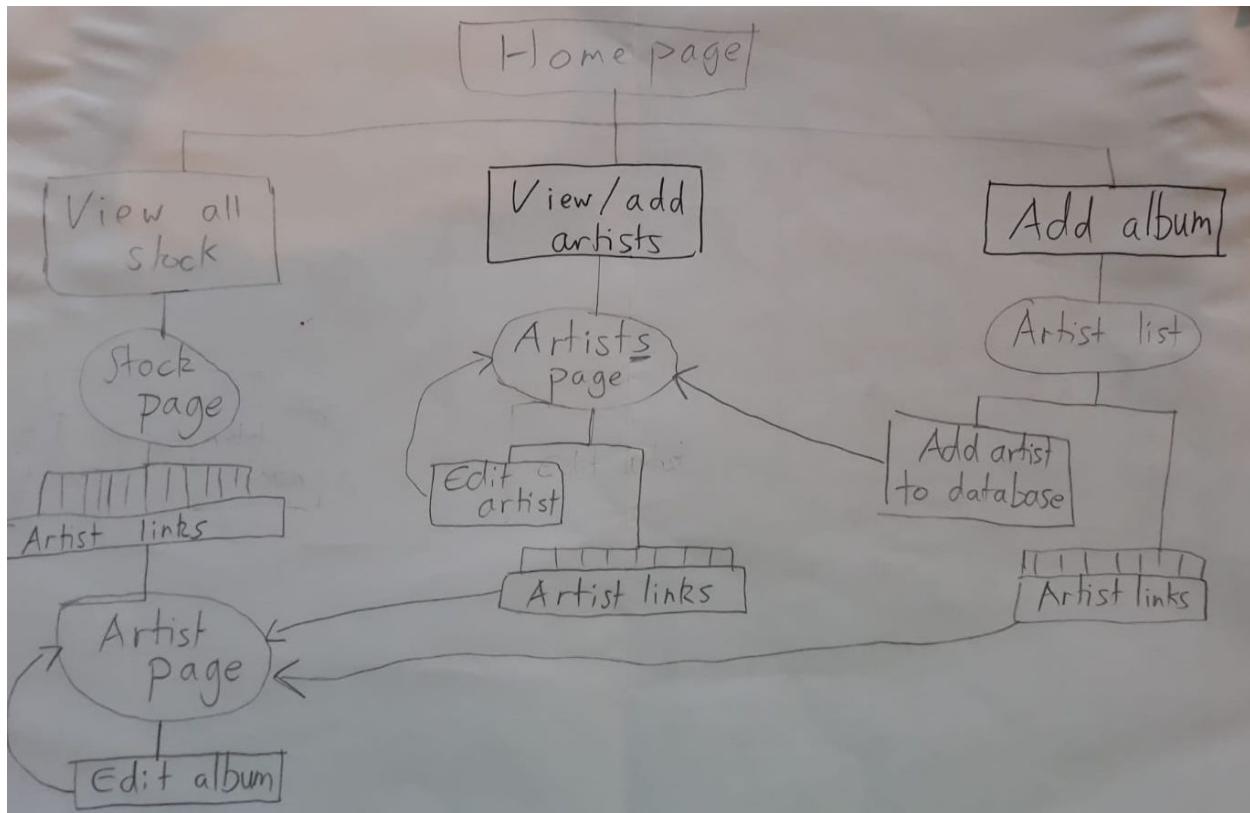
A.D.4. Activity diagram:



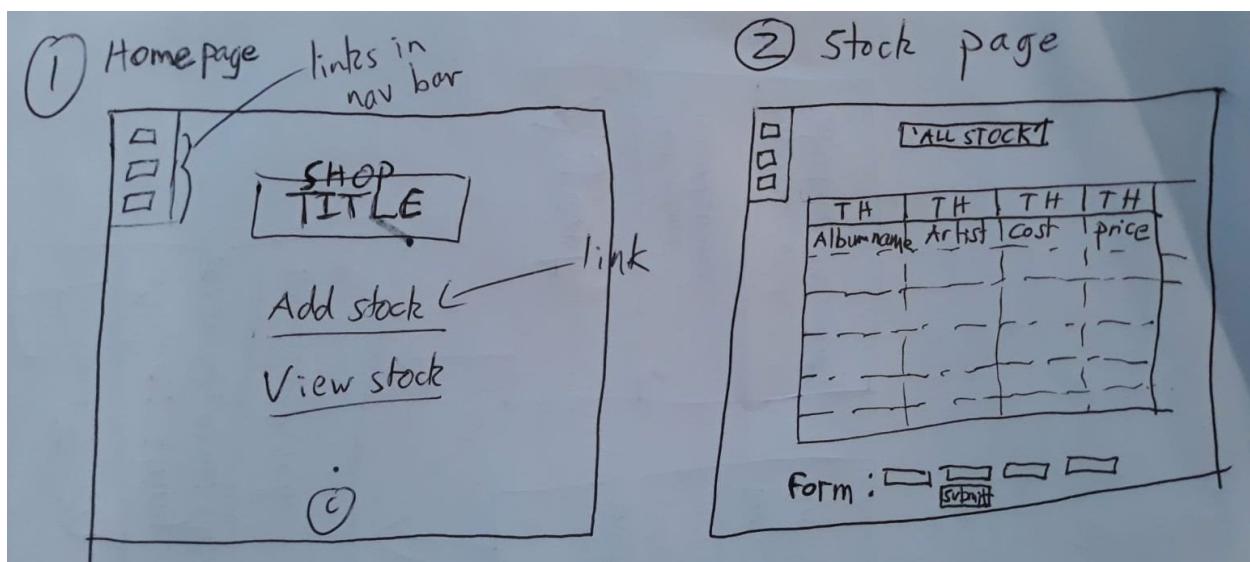
A.D.6. Implementations constraints plan:

Constraint category	Implementation constraint	Solution
Hardware and software platforms	For this project I was only allowed to use Ruby, which meant certain dynamic features were unavailable to me. I wanted to include a drop-down list of artists for the user to select from when adding an album to the database, but this was not possible.	I had to re-plan my interface, and instead list all my artists statically on the page and make each artist a link to another page. This alternative interface was clear and usable enough for my purposes.
Performance requirements	The intended purpose of the app is for store managers/staff to be able to update stock as and when it is bought/sold, so efficiency/speed is important.	I ensured only necessary data was stored in the database. Data that could be calculated based on existing data was not included in the back-end, so as to not use unnecessary storage.
Persistent storage and transactions	My program had to allow shop staff to keep closing the program and coming back to it as it was designed to be long-term storage for data.	I ensured my data was stored in a database in a separate back-end with a router to retrieve and save information, so that data persisted after the program was closed.
Usability	As already stated, the intended purpose of the app is for store managers/staff to be able to update stock as and when it is bought/sold, which means it has to be easy to navigate and quick to learn how to use.	There are no unnecessary frills in the UI, and it is easy to navigate back and forth between tables and the homepage, with obvious and clear links.
Budgets	I had no money to complete this project. I was therefore unable to hire staff to complete it for me.	I overcame this constraint by doing all the work myself.
Time limitations	The program had to be designed and created within one week.	I dealt with this strict time constraint by constantly being aware that I had to prioritise which features to ensure were complete and working, and decide which ones were not part of the MVP so had to be abandoned.

P.5. User site map:



P.6. Wireframe diagrams:



P.10. Pseudocode:

```

def check_in(guest)
    if (guest has enough money) AND
        if guest(existing guests
            have not reached capacity)
            → push guest to guest array
    else
        → "guest not checked in"
    then
        for loop:
            - for each song in room's song-array
                if song is guest's favourite song
                    → guest. celebrate
            else
                → guest. commiserate
end

```

P.13. User input being processed:

Chrome File Edit View History Bookmarks People Window Help

My Drive - Google Dr | PDA evidence - Goog | PDA evidence - Goog | PDA evidence - Goog | Your Repositories | inventory +

https://badbass.herokuapp.com/artists/74/albums

All Stock Artists Home

Paul Simon

Album code	Album title	Buying cost	Selling price	Mark-up	Stock	Stock status	Edit
120	Graceland	7.0	10.0	3.0	6 Add to stock Reduce stock	In stock	Edit Album Delete album from database
121	Rhythm of the Saints	4.0	8.0	4.0	7 Add to stock Reduce stock	In stock	Edit Album Delete album from database
151	Stranger to Stranger	7.0	11.0	4.0	0 Add to stock Reduce stock	Out of stock	Edit Album Delete album from database

Add album:

Title: One Trick Pony Buying cost: 3 Price: 6 Stock: 1 SUBMIT

All Stock Artists Home

Paul Simon

Album code	Album title	Buying cost	Selling price	Mark-up	Stock	Stock status	Edit
120	Graceland	7.0	10.0	3.0	6 Add to stock Reduce stock	In stock	Edit Album Delete album from database
152	One Trick Pony	3.0	6.0	3.0	1 Add to stock Reduce stock	Low stock	Edit Album Delete album from database
121	Rhythm of the Saints	4.0	8.0	4.0	7 Add to stock Reduce stock	In stock	Edit Album Delete album from database
151	Stranger to Stranger	7.0	11.0	4.0	0 Add to stock Reduce stock	Out of stock	Edit Album Delete album from database

Add album:

Title: Buying cost: Price: Stock:

The above screenshots show the Paul Simon album page from my project. The first screenshot shows the user having inputted details of a new album they wish to add to the database. The second screenshot shows the new rendering of the page after the form had been submitted. These details have been processed and are now included in the list of albums.

P.14. Interaction with data persistence:

localhost / inventory / artists Connected. PostgreSQL 10.5 LOCAL

id	name	genre
14	Abba	Pop
6	Backstreet Boys	Pop
7	Bob Marley	Other
3	Cosmo Jarvis	Other
11	Ella Fitzgerald	Jazz
4	Fleetwood Mac	Other
10	Frank Sinatra	Jazz
16	Maximum the Hormone	Rock
12	Muse	Rock
2	Paul Simon	Other
15	Porcupine Tree	Other
5	Taylor Swift	Pop
8	The Beach Boys	Other
17	The Proclaimers	Other
13	The Rolling Stones	Rock
1	Various	Other
9	Weird Al Yankovic	Other

Content Structure DDL + Row 17 rows Filter < Page 1 of 1

Chrome File Edit View History Bookmarks People Window Help
My Drive - Google Dr × PDA evidence - Google Dr × PDA evidence - Google Dr × PDA evidence - Google Dr × Your Repositories × inventory × +
localhost:4567/artists

← → ⌂ ⓘ localhost:4567/artists

	Name	Genre	Action	Action
10	Frank Sinatra	Jazz	Delete artist from database	Edit artist
16	Maximum the Hormone	Rock	Delete artist from database	Edit artist
12	Muse	Rock	Delete artist from database	Edit artist
2	Paul Simon	Other	Delete artist from database	Edit artist
15	Porcupine Tree	Other	Delete artist from database	Edit artist
5	Taylor Swift	Pop	Delete artist from database	Edit artist
8	The Beach Boys	Other	Delete artist from database	Edit artist
17	The Proclaimers	Other	Delete artist from database	Edit artist
13	The Rolling Stones	Rock	Delete artist from database	Edit artist
1	Various	Other	Delete artist from database	Edit artist
9	Weird Al Yankovic	Other	Delete artist from database	Edit artist

Add artist:

Name: Genre:



The screenshot shows a database table named 'artists' in Postico. The table has three columns: 'id', 'name', and 'genre'. The 'name' column is sorted in ascending order. The data consists of 18 rows, each representing an artist and their genre. The genres listed are Pop, Other, Jazz, and Rock.

id	name	genre
14	Abba	Pop
6	Backstreet Boys	Pop
7	Bob Marley	Other
18	Britney Spears	Pop
3	Cosmo Jarvis	Other
11	Ella Fitzgerald	Jazz
4	Fleetwood Mac	Other
10	Frank Sinatra	Jazz
16	Maximum the Hormone	Rock
12	Muse	Rock
2	Paul Simon	Other
15	Porcupine Tree	Other
5	Taylor Swift	Pop
8	The Beach Boys	Other
17	The Proclaimers	Other
13	The Rolling Stones	Rock
1	Various	Other
9	Weird Al Yankovic	Other

The above screenshots show the artists in my database, both from the back-end and front-end. The first screenshot shows the artists listed in my database (on Postico). The second screenshot shows the user having inputted details of a new artist that they wish to add to the database, via the front-end artists page form. The third screenshot shows the new artist list in Postico after the form had been submitted. These form details have been processed and have persisted in the back-end.

P.15. Output of results and feedback to user:

localhost:4567/albums

		Legend						
Bob Marley	11 Delete album from database	4.0	8.0	4.0	1 Add to stock Reduce stock	Low stock		
Various	Now! That's what I call coding 28 Delete album from database	4.0	8.0	4.0	2 Add to stock Reduce stock	Low stock		
Paul Simon	One Trick Pony 34 Delete album from database	6.0	8.0	2.0	6 Add to stock Reduce stock	In stock		
Muse	Origin of Symmetry 19 Delete album from database	4.0	8.0	4.0	0 Add to stock Reduce stock	Out of stock		
The Beach Boys	Pet Sounds 12 Delete album from database	4.0	8.0	4.0	4 Add to stock Reduce stock	In stock		
Taylor Swift	Reputation 9 Delete album from database	4.0	8.0	4.0	8 Add to stock Reduce stock	In stock		
Paul Simon	Rhythm of the Saints 2 Delete album from database	4.0	8.0	4.0	4 Add to stock Reduce stock	In stock		
Fleetwood Mac	Rumours 5 Delete album from database	5.0	6.0	1.0	0 Add to stock Reduce stock	Out of stock		

localhost:4567/artists/1/albums

All Stock Artists Home

Various

Album code	Album title	Buying cost	Selling price	Mark-up	Stock	Stock status	Edit
29	CodeClan	4.0	8.0	4.0	7 Add to stock Reduce stock	In stock	Edit Album Delete album from database
30	Disney Hits	4.0	8.0	4.0	0 Add to stock Reduce stock	Out of stock	Edit Album Delete album from database
28	Now! That's what I call coding	4.0	8.0	4.0	2 Add to stock Reduce stock	Low stock	Edit Album Delete album from database

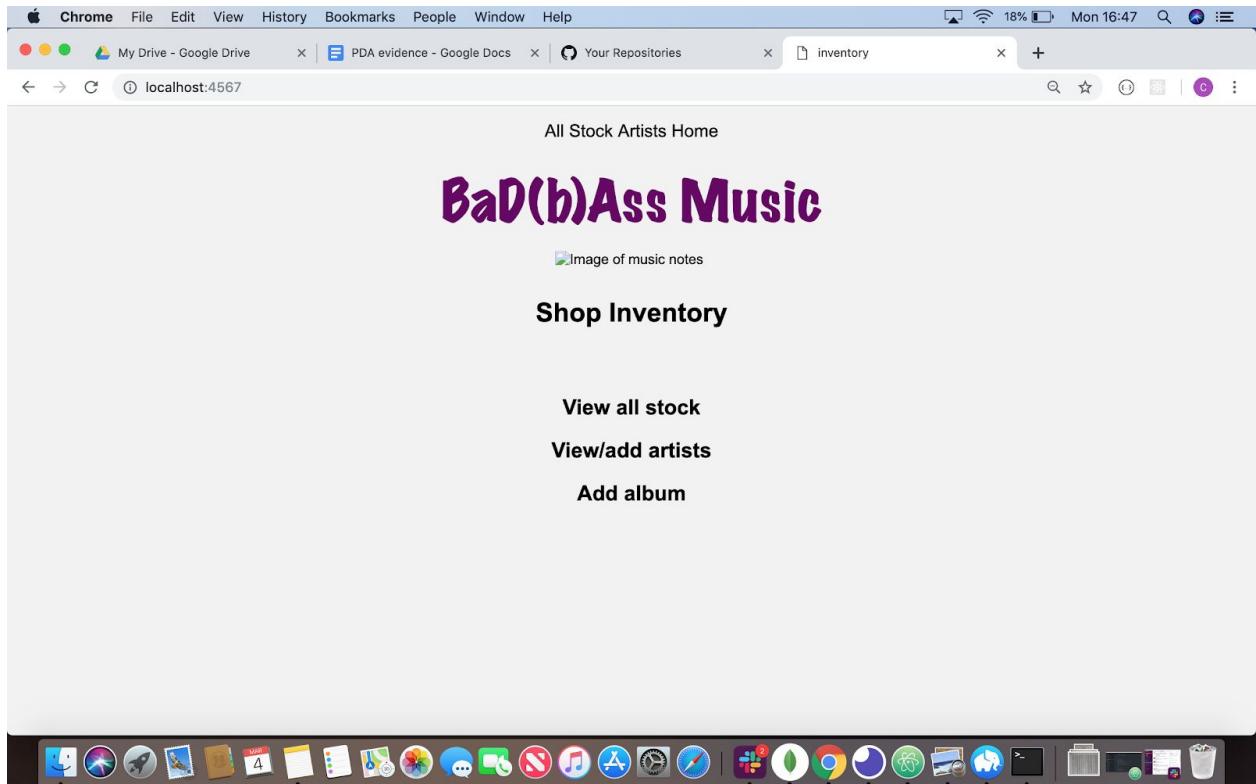
Add album:

Title: Buying cost: Price: Stock:

The above screenshots demonstrate my program providing expected feedback to the user. The first screenshot shows my table of all stock, and the user clicking on a link to the artist belonging

to a particular album. The second screenshot shows what is returned to the user after clicking on that link: the user sees all albums listed under that artist.

P.11. Individual project and Github repo:



https://github.com/cass3836/Project1_shop_inventory

Above is a screenshot of the Homepage of the shop inventory tracking program I created for my individual project, displayed locally. I have also included the Github link to my repo with all the code.

P.12. Planning/stages of development:

Wk 4/5 Project - Shop Inventory

{ - Artists - name
((- Albums - name, des, stock, cost, price, | decade ?
| artist ID?
→ CRUD

Inventor:

- list all albums (and their artists)
- highlight low stock or out of stock (in select function?)

Filters:

- artist
- date.
- (- song?)
- + ~~add~~ album fields ^{name}
- price (> or < something)

✓ add artist
✓ add albums - edits stock count accordingly.
✓ delete .. - if artist = deleted, so
are all albums but
not vice versa.
✓ .. - edits stock count accordingly.
✓ edit .. - if an edit ~~or~~ cause to be a duplicate,
~~delete~~ and update stock count of duplicate
✓ filter / find partial or a + all (and all)
- highlight low + out of stock. [not doing]

- ✓ extra column: in / low / out soft stock
 - ✓ 'mark-up' column
 - filter by price
 - ✓ automatically fill in artist ID
 - ✓ delete button
 - ✓ edit button
 - To do on - All Albums page:
 - mark-up? (calculate?)
 - ✓ Add + Debt stock buttons
 - ✓ Stock status part of MVP
 - ✓ Artist name
 - filters (by genre + by price) & by album
 - To do on 'Artist' page
 - ✓ genre
 - project charge edit thing so can't repeat actually not sure that's right.
 - Other AFE:
 - ✓ auto fill edit form
 - are you sure? no yes
 - ✓ homepage redirection.
 - ✓ stop order flicking. (- say page)
- If fine:
- ✓ add mark-up to db
 - ✓ sent out edit can't gonna happen
 - ✓ search bar in All Stock page
 - fix artist edit thing against errors

P.16. Use of an API:

Atom File Edit View Selection Find Packages Window Help

request_helper.js — ~/Documents/codeclan_work/PDA/I-and-T/wk7-d3-countries-lab

Project

- wk7-d3-countries-lab
 - public
 - src
 - helpers
 - pub_sub.js
 - request_helper.js
 - models
 - country.js
 - views
 - app.js
 - .gitignore
 - package-lock.json
 - package.json
 - webpack.config.js

```
const RequestHelper = function(url){  
    this.url = url  
}  
  
RequestHelper.prototype.get = function(onComplete){  
    const xhr = new XMLHttpRequest();  
  
    xhr.addEventListener('load', () =>{  
        if (xhr.status !== 200) return;  
        const data = JSON.parse(xhr.responseText);  
        onComplete(data);  
    });  
  
    xhr.open('GET', this.url);  
    xhr.setRequestHeader('Accept', 'application/json');  
    xhr.send();  
}  
  
module.exports = RequestHelper;
```

src/helpers/request_helper.js 18:1

File Test

LF UTF-8 JavaScript master Fetch 0 files

Atom File Edit View Selection Find Packages Window Help

country.js — ~/Documents/codeclan_work/PDA/I-and-T/wk7-d3-countries-lab

Project

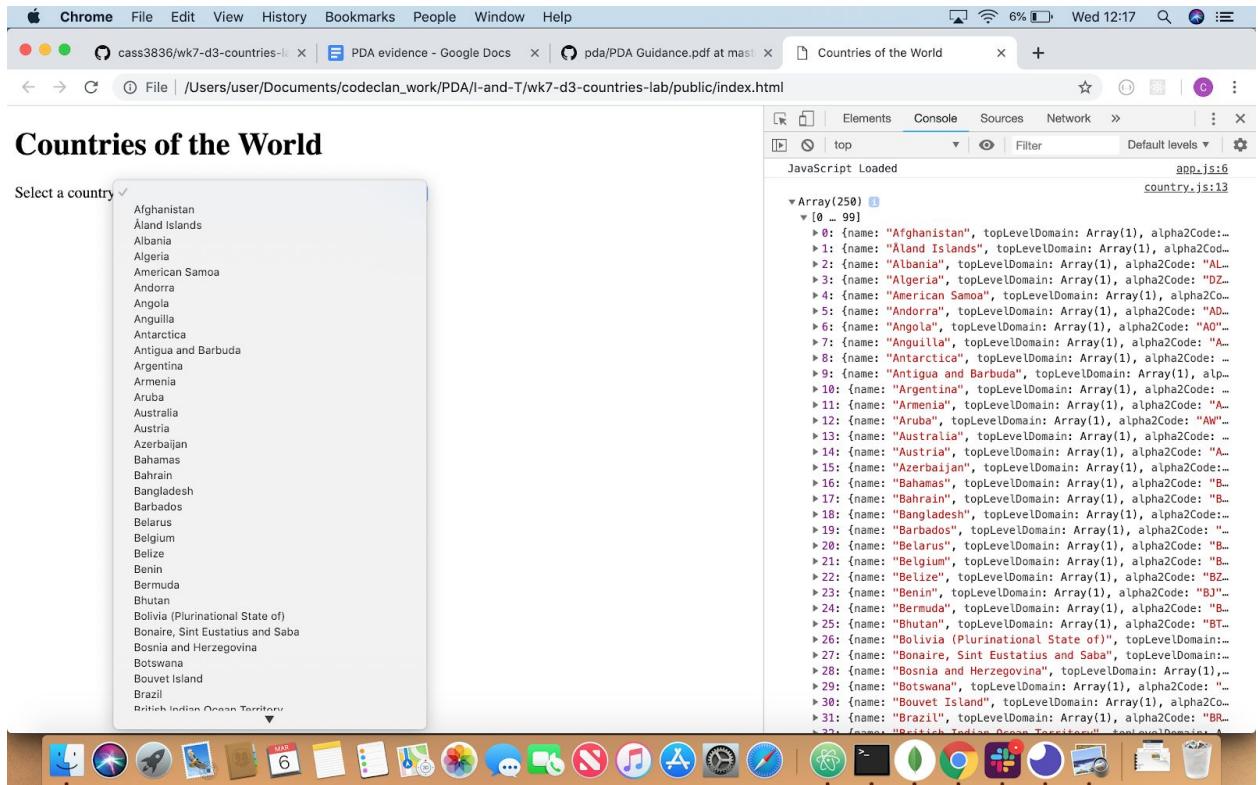
- wk7-d3-countries-lab
 - public
 - css
 - js
 - index.html
 - src
 - helpers
 - pub_sub.js
 - request_helper.js
 - models
 - country.js
 - views
 - app.js
 - .gitignore
 - package-lock.json
 - package.json
 - webpack.config.js

```
const PubSub = require('../helpers/pub_sub.js');  
const RequestHelper = require('../helpers/request_helper.js');  
  
const Country = function() {  
    this.text = null;  
}  
  
Country.prototype.getData = function(){  
    const request = new RequestHelper('https://restcountries.eu/rest/v2/all');  
    request.get((data) => {  
        this.text = data;  
        PubSub.publish('Country:allData', this.text);  
        console.log(this.text);  
    });  
    PubSub.subscribe('SelectView:selected-index', (event) => {  
        const selectedIndex = event.detail;  
        console.log(this);  
        this.getCountry(selectedIndex);  
        // console.log(selectedIndex);  
    });  
    PubSub.subscribe('CountryInfoView:selectedBorder', (event) => {  
        const selectedBorder = event.detail;  
        this.getIndex(selectedBorder);  
    });  
}
```

src/models/country.js 13:23

File Test

LF UTF-8 JavaScript master Fetch 1 file



- The above screenshots show my use of a public API containing country data.
- The first screenshot shows the function I wrote to retrieve the API data. It is written as a helper method in a separate document to make the code more readable.
- The second screenshot shows this function being used in the 'getData' function that I wrote for a Country model. In line 9, the URL for the API is given as an argument to the RequestHelper.
- The console log on line 13 is shown in the third screenshot, where all the data from the API has been successfully retrieved and logged. (The drop-down list also shows the API being used and displayed in the UI.)

P.18. Testing:

Atom File Edit View Selection Find Packages Window Help

Project — ~/Documents/codeclan_work/PDA_Static_and_Dynamic_Task_A

testing_task_2.rb testing_task_spec.rb testing_task_1.md

```

Project
PDA_Static_and_Dynamic_Task_A
  specs
    testing_task_spec.rb
      card.rb
      Static_&_Dynamic_Testing.md
      testing_task_1.md
      testing_task_2.rb

testing_task_spec.rb
11 @card1 = Card.new("diamonds", 5)
12 @card2 = Card.new('spades', 5)
13 end
14
15 def test_check_for_ace_true
16   assert_equal(true, @cardgame1.check_for_ace(@card1))
17 end
18
19 def test_check_for_ace_false
20   assert_equal(false, @cardgame1.check_for_ace(@card2))
21 end
22
23 def test_check_highest_card
24   assert_equal("5 of diamonds", @cardgame1.highest_card(@card1, @card2))
25 end
26
27 def test_check_highest_card_equal
28   assert_equal("Cards have equal value", @cardgame1.highest_card(@card3, @card2))
29 end
30
31 def test_cards_total
32   assert_equal("You have a total of 6", @cardgame1.cards_total([@card1, @card2]))
33 end
34
35 end
36

```

specs/testing_task_spec.rb 4:22

LF UTF-8 Ruby master Fetch 0 files

..java/students

```

PDA_Static_and_Dynamic_Task_A git:(master) ✘ ruby specs/testing_task_spec.rb
Run options: --seed 8767

# Running:

F.F..

Finished in 0.001263s, 3958.8282 runs/s, 3958.8282 assertions/s.

1) Failure:
TestCardGame#test_check_for_ace_true [specs/testing_task_spec.rb:16]:
Expected: true
Actual: false

2) Failure:
TestCardGame#test_cards_total [specs/testing_task_spec.rb:32]:
Expected: "You have a total of 6"
Actual: "You have a total of 15"

5 runs, 5 assertions, 2 failures, 0 errors, 0 skips
→ PDA_Static_and_Dynamic_Task_A git:(master) ✘

```

Project

```

PDA_Static_and_Dynamic_Task_A
  specs
    testing_task_spec.rb
    card.rb
    Static_&_Dynamic_Testing.md
    testing_task_1.rb
    testing_task_2.rb

```

```

testing_task_2.rb
require_relative('card.rb')
class CardGame
  def check_for_ace(card)
    if card.value == 1
      return true
    else
      return false
    end
  end

  def highest_card(card1, card2)
    if card1.value > card2.value
      return "#{card1.value} of #{card1.suit}"
    elsif
      card1.value < card2.value
      return "#{card2.value} of #{card2.suit}"
    else
      return "Cards have equal value"
    end
  end

  def cards_total(cards)
    total = 0
    for card in cards
      total += card.value
    end
    return "You have a total of #{total}"
  end
end

```

testing_task_2.rb 11:18

LF UTF-8 Ruby master Fetch 2 files

```

..java/students ..$ 
[→ PDA_Static_and_Dynamic_Task_A git:(master) ✘ ruby specs/testing_task_spec.rb
Run options: --seed 64775

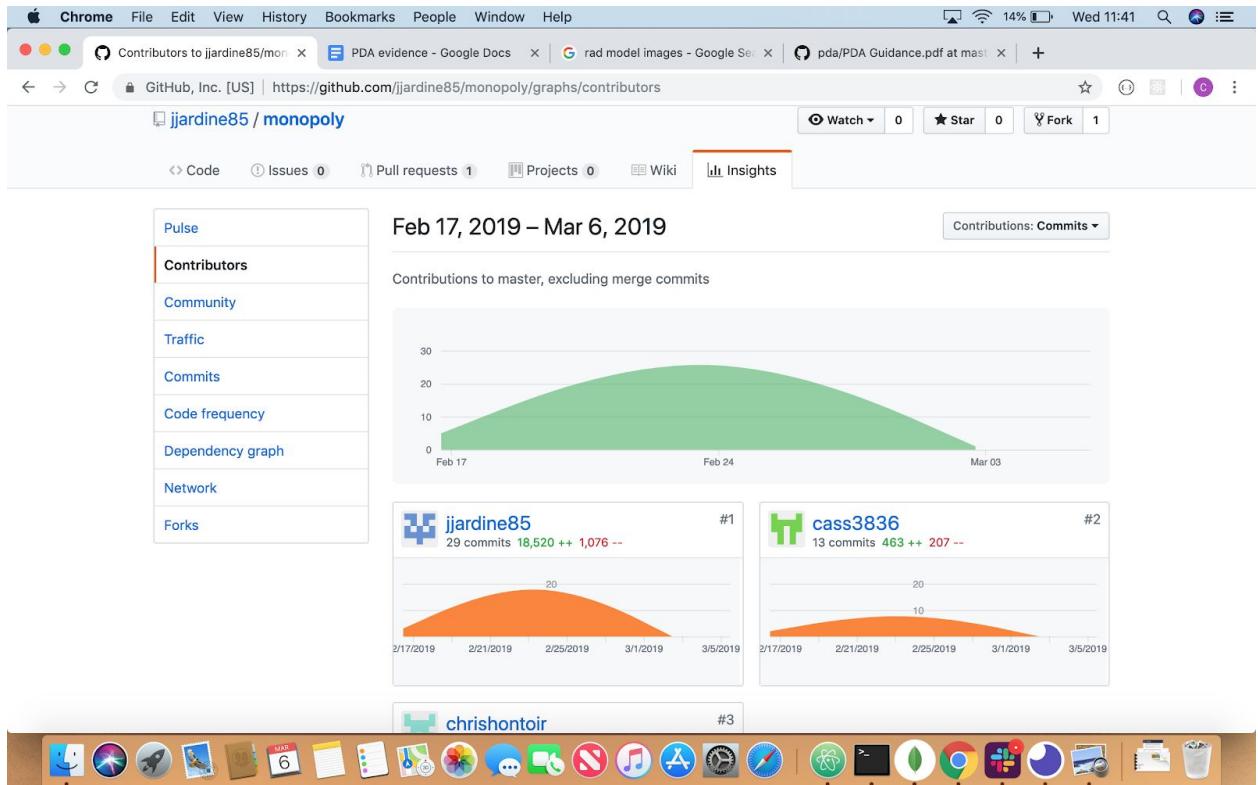
# Running:

*****
Finished in 0.001168s, 4280.8219 runs/s, 4280.8219 assertions/s.

5 runs, 5 assertions, 0 failures, 0 errors, 0 skips
→ PDA_Static_and_Dynamic_Task_A git:(master) ✘

```

P.1. Github contributor's page:



The above screenshot shows the contributor page for the Github repo from my group project. For the project, a group of 4 of us worked together to create an online version of the Monopoly board game. We used 3 of our accounts to push commits to the master branch, (but primarily we used jjardine85).

P.2. Group project brief:

Branch: master sw1_classnotes / week_09 / projects / briefs / browser_game.md

cifarquhar add group project materials 28f1dfb 13 days ago

1 contributor

8 lines (4 sloc) | 419 Bytes

Raw Blame History

Browser Game

Create a browser game based on an existing card or dice game. Model and test the game logic and then display it in the browser for a user to interact with.

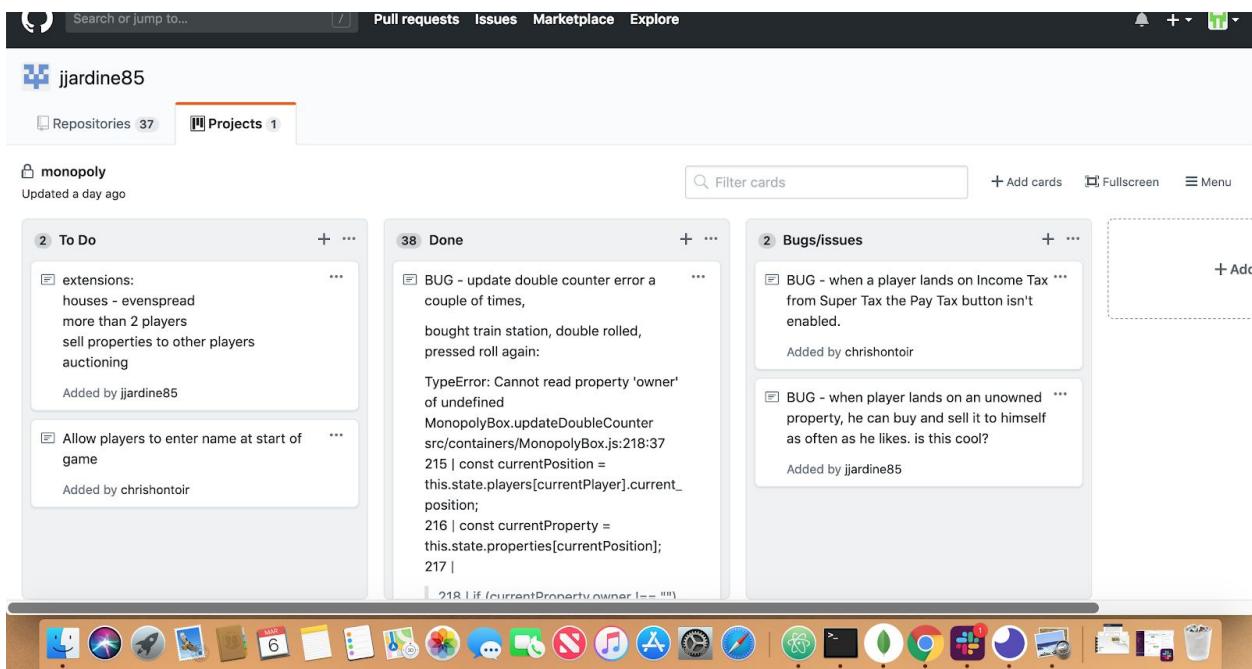
Write your own MVP with some specific goals to be achieved based on the game you choose to model.

You might use persistence to keep track of the state of the game or track scores/wins. Other extended features will depend on the game you choose.



Above is the official brief provided to my group by our instructors for our group project. We had a week to create a working version of a browser game that was based on an existing game. As mentioned above, we opted to re-create an online version of Monopoly. We created specific goals for our MVP, which can be seen in the photo below (under P.3).

P.3. Planning during group project:



The above screenshots demonstrate a few elements of my group's planning/development process.

- The first screenshot is a slide from the final presentation of our work. As can be seen, we had an initial plan for an MVP and extensions. During a retro meeting part-way through the week, we re-wrote our goals, as we had come on further than expected in the given

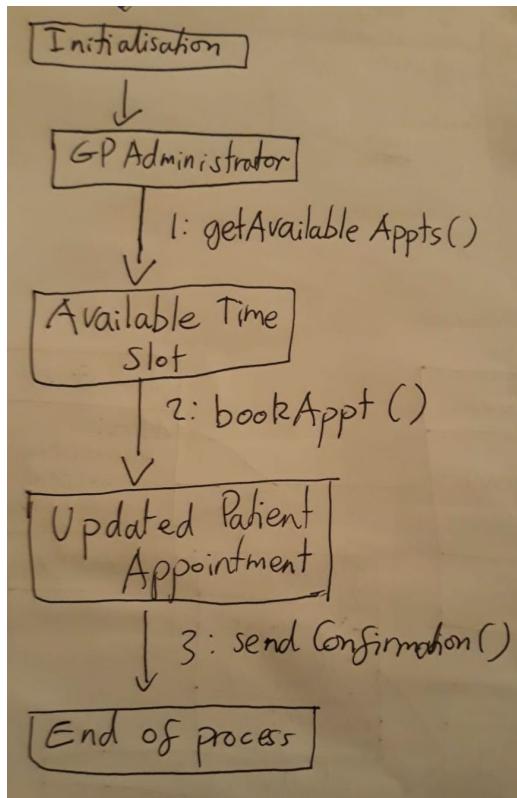
time, whilst also having reached certain barriers that we had not anticipated. As such, we wrote a new set of MVP/extension aims.

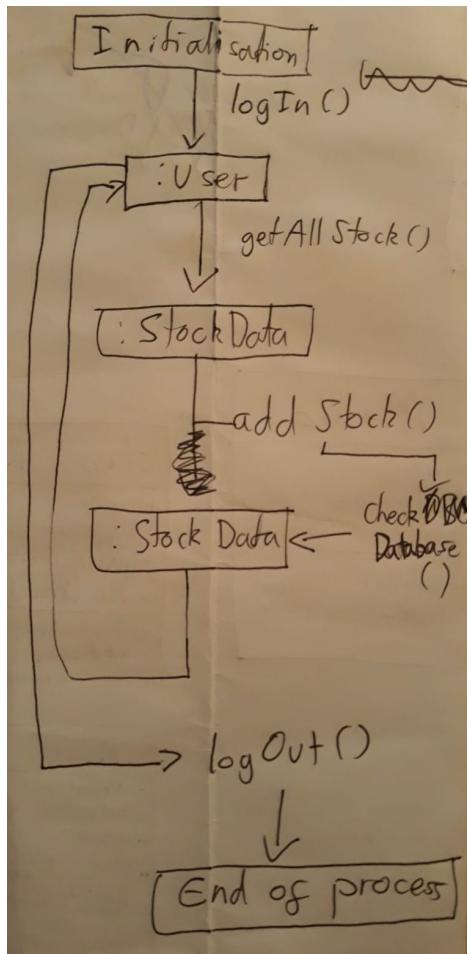
- The first screenshot also shows an initial, minimally-styled version of our board display, as well the final version.
- The second screenshot shows our Kanban board via which we tracked the completion state of different tasks, as well as various bugs that needed fixing.

P.4. Acceptance criteria and test plan:

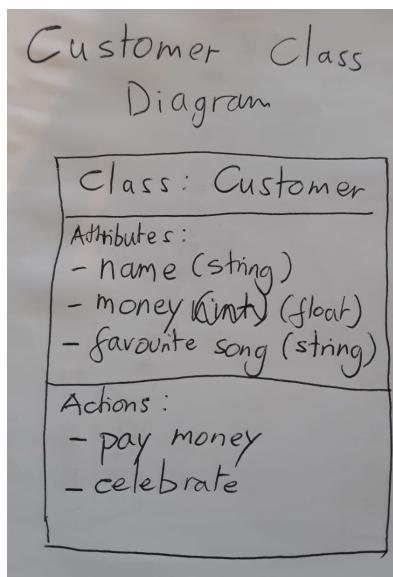
Acceptance Criteria	Expected Result	Pass/Fail
A user is able to view all stock in a single view.	The 'View all stock' link on the Homepage redirects to a table displaying all albums in alphabetical order, and their details.	Pass
A user is able to add stock to the database.	The 'Add album' link on the Homepage redirects to a list of all artists, which the user can click on to take them to the page of albums under that artist. Here they can input new album details in a form at the bottom of the page, submit the form, and see the new album displayed with the rest of the stock.	Pass
A user can only add an album to the database if the artist is already saved to the database.	The list of artists that the user is redirected to when they click 'Add album' has an option at the bottom to 'Add artist to database'. This button will redirect them to a table of all the artists, with a form underneath via which a new artist can be saved to the database. The user can then click on that artist and add an album in the normal way.	Pass
A user is able to edit stock once it has been added to the database.	On the filtered stock pages (i.e. on each artist's table of albums) each album has an 'Edit album' button, which redirects the user to a form that allows them to re-submit the details of that album. This overrides the original details.	Pass

P.7. System interaction diagrams:

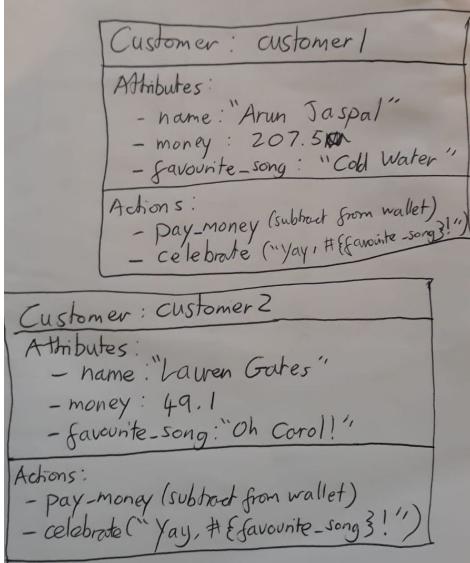




P.8. Object diagrams:



Customer Object Diagrams



P.17. Bug tracking report:

Bug/error	Solution	Date
When a double is rolled when passing go the piece disappears.	Our playerMove function had the passGo function and the checkDoubleCounter in the wrong order, so it was trying to re-position the pieces in places that did not exist. We simply shifted the order of functions around.	26/02/2019
After mortgaging and unmortgaging several times, the players money turns into a float.	The calculation in the mortgaging functions did not round the numbers down before performing calculations. We added a .floor.	28/02/2019
If a player mortgages a property before the other player pays them rent, the pay rent button is disabled and the game gets stuck.	We disabled the mortgage button for current property until the pay rent button is pressed.	27/02/2019
A player is able to buy houses on utilities.	We added a conditional to adding of the buyHouses element that meant it was only added if the property set was neither stations nor utilities.	27/02/2019
When a player lands on an unowned property, they can buy and sell it ad infinitum.	We did not 'fix' this bug as it was not a priority because it did not actually do anything to break the game.	N/A

I.T.7. Use of polymorphism:

The screenshot shows the Atom code editor interface. On the left, the project structure for 'instrument-shop' is visible, containing files like Guitar.class, Guitar.java, instructions, IPlay.class, IPlay.java, Piano.class, Piano.java, Shop.class, and Shop.java. The 'Shop.java' file is currently open in the main editor area, displaying Java code for a 'Shop' class. The 'IPlay.java' file is also listed in the tabs at the top. The status bar at the bottom shows 'Shop.java*' 22:48, LF, UTF-8, Java, 0 files, and a battery icon.

```
Project
instrument-shop
  - Guitar.class
  - Guitar.java
  - instructions
  - IPlay.class
  - IPlay.java
  - Piano.class
  - Piano.java
  - Shop.class
  - Shop.java

Shop.java
  2
  3 public class Shop {
  4
  5     private ArrayList<IPlay> stock;
  6     private String shopName;
  7
  8     public Shop(String name){
  9         this.shopName = name;
 10         this.stock = new ArrayList<IPlay>();
 11     }
 12
 13     public void addStock(IPlay stock) {
 14         this.stock.add(stock);
 15     }
 16
 17     public ArrayList<IPlay> getStock(){
 18         return this.stock;
 19     }
 20
 21     public static void main(String[] args){
 22         Shop cassiaMusic = new Shop("CassiaMusic");
 23         Piano p = new Piano();
 24         Guitar g = new Guitar();
 25
 26         cassiaMusic.addStock(p);
 27         cassiaMusic.addStock(g);
 28
 29         for (int i=0; i<cassiaMusic.getStock().size(); i++ ) {
 30             System.out.println(cassiaMusic.getStock().get(i).play());
 31         }
 32     }
 33 }
```

The screenshot shows the Atom code editor interface. On the left, the project structure for 'instrument-shop' is visible, containing files like Guitar.class, Guitar.java, instructions, IPlay.class, IPlay.java, Piano.class, Piano.java, Shop.class, and Shop.java. The 'IPlay.java' file is currently selected and open in the main editor area, displaying Java code for a 'IPlay' interface. The 'Shop.java' file is also listed in the tabs at the top. The status bar at the bottom shows 'IPlay.java' 22:48, Project — ~/Documents/codeclan_work/java/instrument-shop, 40%, Fri 20:57, a search icon, and a settings icon.

```
Atom File Edit View Selection Find Packages Window Help
Project — ~/Documents/codeclan_work/java/instrument-shop
Shop.java  Guitar.java  IPlay.java  Piano.java

198
199
200
201
202
203
204
205
206
207
208
```

```
public interface IPlay {
    public String play();
}
```

Terminal Shell Edit View Window Help

Project — ~/Documents/codeclan_work/java/instrument-shop

```

MonopolyBc          Project
198
199      instrument-shop
200          Guitar.class
201          Guitar.java
202          instructions
203          IPlay.class
204          IPlay.java
205          Piano.class
206          Piano.java
207          Shop.class
208          Shop.java
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223

```

```

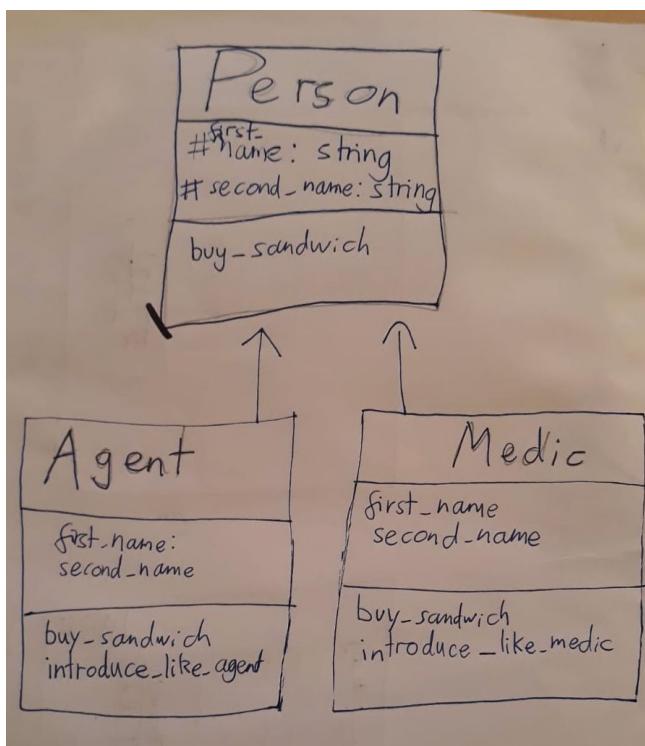
1 public class Piano implements IPlay{
2
3     public String play(){
4         return "Tinkle tinkle";
5     }
6
7
8 }
9
10

```

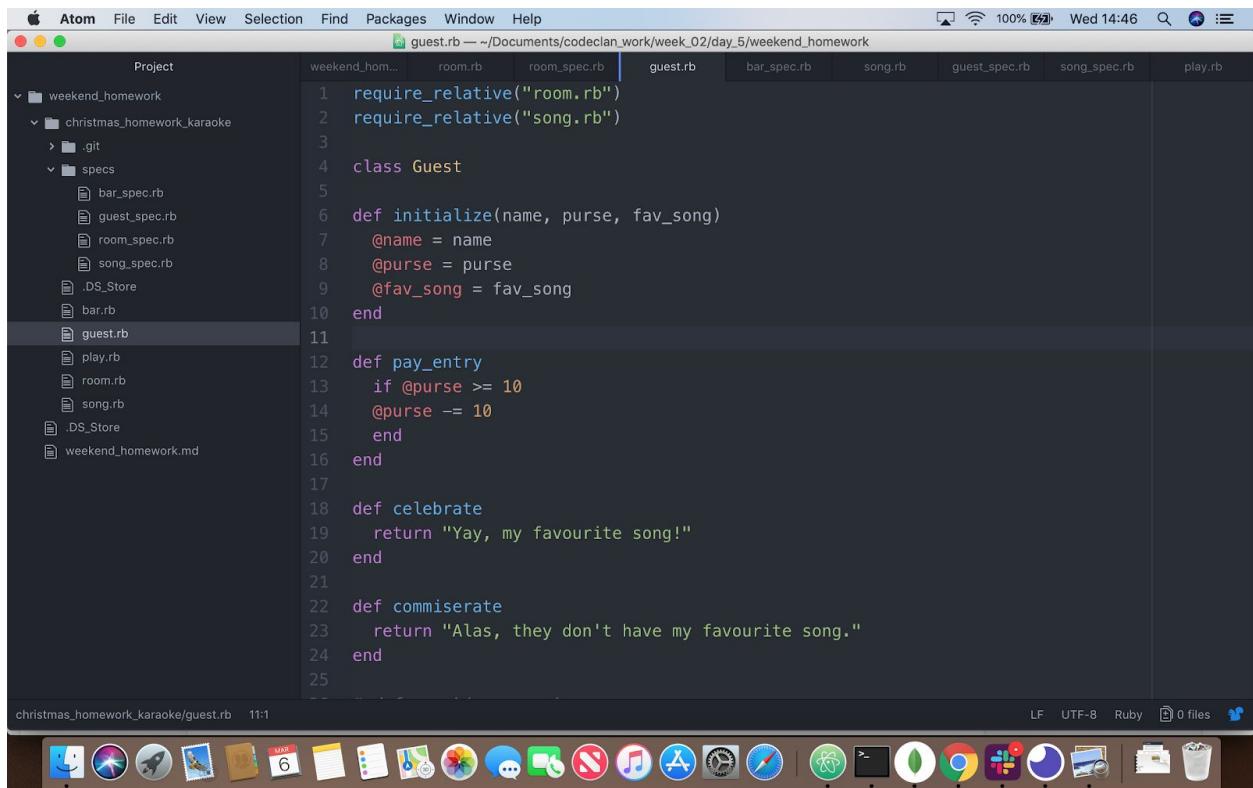
instrument-shop — user@users-MacBook-Pro — zsh — 80x24
..java/students ..instrument-shop ..strument-shop +
↳ instrument-shop java Shop.javae
Error: Could not find or load main class Shop.javae
↳ instrument-shop java Shop
Tinkle tinkle
Twangle twangle
↳ instrument-shop

src/contains Piano.java 1:3 LF UTF-8 Java 0 files

A.D.5. Inheritance diagram:



I.T.1 Use of encapsulation:



The screenshot shows the Atom code editor interface on a Mac OS X desktop. The window title is "guest.rb — ~/Documents/codeclan_work/week_02/day_5/weekend_homework". The left sidebar displays a project structure with files like room.rb, room_spec.rb, guest.rb, bar_spec.rb, song.rb, guest_spec.rb, song_spec.rb, play.rb, and .DS_Store. The main editor area contains the following Ruby code:

```
1 require_relative("room.rb")
2 require_relative("song.rb")
3
4 class Guest
5
6   def initialize(name, purse, fav_song)
7     @name = name
8     @purse = purse
9     @fav_song = fav_song
10  end
11
12  def pay_entry
13    if @purse >= 10
14      @purse -= 10
15    end
16  end
17
18  def celebrate
19    return "Yay, my favourite song!"
20  end
21
22  def commiserate
23    return "Alas, they don't have my favourite song."
24  end
25
```

The status bar at the bottom shows "christmas_homework_karaoke/guest.rb 11:1" and icons for various Mac applications.

Terminal Shell Edit View Window Help guest_spec.rb — ~/Documents/codeclan_work/week_02/day_5/weekend_homework

```

Project
└─ weekend_homework
    └─ christmas_homework_karaoke
        ├ .git
        └─ specs
            └─ guest_spec.rb
            └─ room_spec.rb
            └─ song_spec.rb
        └─ .DS_Store
        └─ bar.rb
        └─ guest.rb
        └─ play.rb
        └─ room.rb
        └─ song.rb
    └─ .DS_Store
    └─ weekend_homework.md

6
7 class TestGuest < MiniTest::Test
8
9 def setup
10     @guest1 = Guest.new("Arun Jaspal", 7, "Cold Water")
11     @guest2 = Guest.new("Natasha Thottacherry", 10, "I'm a Believer")
12     @guest3 = Guest.new("Mary Clare Doran", 40, "Hotel California")
13     @guest4 = Guest.new("Cassia Lonsdale", 98, "I'm a Believer")
14     @guest5 = Guest.new("Tim Fozard", 151, "My Way")
15     @guest6 = Guest.new("Ryan McKay", 30, "22")
16     @guest7 = Guest.new("Helena Jones", 70, "22")
17     @guest8 = Guest.new("Jon Badcock", 45, "22")
18 end
19
20 def test_get_name
21     assert_equal("Arun Jaspal", @guest1.name)
22 end
23
24 def test_get_purse
25     assert_equal(7, @guest1.purse)
26 end
27
28 def test_get_fav_song
29     assert_equal("Cold Water", @guest1.fav_song)
30 end
31

```

Running:
EE
Finished in 0.001088s, 2757.3529 runs/s, 0.0000 assertions/s.
1) Error:
TestGuest#test_get_name:
NoMethodError: undefined method 'name' for #<Guest:0x007fa1950ce2f0
specs/guest_spec.rb:21:in `test_get_name'
2) Error:
TestGuest#test_get_fav_song:
NoMethodError: undefined method 'fav_song' for #<Guest:0x007fa1950b
specs/guest_spec.rb:29:in `test_get_fav_song'
3) Error:
TestGuest#test_get_purse:
NoMethodError: undefined method 'purse' for #<Guest:0x007fa1950af46
specs/guest_spec.rb:25:in `test_get_purse'
3 runs, 0 assertions, 0 failures, 3 errors, 0 skips
christmas_homework_karaoke git:(master) ✘

LF UTF-8 Ruby 0 files

Atom File Edit View Selection Find Packages Window Help guest.rb — ~/Documents/codeclan_work/week_02/day_5/weekend_homework

```

Project
└─ weekend_homework
    └─ christmas_homework_karaoke
        ├ .git
        └─ specs
            └─ guest_spec.rb
            └─ room_spec.rb
            └─ song_spec.rb
        └─ .DS_Store
        └─ bar.rb
        └─ guest.rb
        └─ play.rb
        └─ room.rb
        └─ song.rb
    └─ .DS_Store
    └─ weekend_homework.md

1 require_relative("room.rb")
2 require_relative("song.rb")
3
4 class Guest
5
6     attr_accessor :name, :purse, :fav_song
7
8     def initialize(name, purse, fav_song)
9         @name = name
10        @purse = purse
11        @fav_song = fav_song
12    end
13
14    def pay_entry
15        if @purse >= 10
16            @purse -= 10
17        end
18    end
19
20    def celebrate
21        return "Yay, my favourite song!"
22    end
23
24    def commiserate
25        return "Alas, they don't have my favourite song."
26    end

```

christmas_homework_karaoke/guest.rb 7:1

Terminal Shell Edit View Window Help Project — ~/Documents/codeclan_work/week_02/day_5/weekend_homework

weekend_homework

christmas_homework_karaoke

- .git
- specs
- bar_spec.rb
- guest_spec.rb
- room_spec.rb
- song_spec.rb

.DS_Store

bar.rb

guest.rb

play.rb

room.rb

song.rb

.DS_Store

weekend_homework.md

```
6
7 class TestGuest < MiniTest::Test
8
9 def setup
10    @guest1 = Guest.new("Arun Jaspal", 7, "Cold Water")
11    @guest2 = Guest.new("Natasha Thottacherry")
12    @guest3 = Guest.new("Mary Clare Doran", 40)
13    @guest4 = Guest.new("Cassia Lonsdale", 98, "My")
14    @guest5 = Guest.new("Tim Fozard", 151, "Helena Jones", 70, "Cold Water")
15    @guest6 = Guest.new("Ryan McKay", 30, "22")
16    @guest7 = Guest.new("Helena Jones", 70, "Cold Water")
17    @guest8 = Guest.new("Jon Badcock", 45, "22")
18 end
19
20 def test_get_name
21    assert_equal("Arun Jaspal", @guest1.name)
22 end
23
24 def test_get_purse
25    assert_equal(7, @guest1.purse)
26 end
27
28 def test_get_fav_song
29    assert_equal("Cold Water", @guest1.fav_song)
30 end
```

christmas_homework_karaoke — user@users-MacBook-Pro —

node - npm TMPDIR=/var/folders ..ework_karaoke

specs/guest_spec.rb:21:in 'test_get_name'

2) Error:

TestGuest#test_get_fav_song:

NoMethodError: undefined method `fav_song' for #<Guest:0x007fa1950b460>

specs/guest_spec.rb:29:in 'test_get_fav_song'

3) Error:

TestGuest#test_get_purse:

NoMethodError: undefined method `purse' for #<Guest:0x007fa1950af460>

specs/guest_spec.rb:25:in 'test_get_purse'

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

christmas_homework_karaoke git:(master) ✘ ruby specs/guest_spec.

Run options: --seed 24694

Running:

...

Finished in 0.001101s, 2724.7956 runs/s, 2724.7956 assertions/s.

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

christmas_homework_karaoke git:(master) ✘

christmas_homework_karaoke/specs/guest_spec.rb 35:6

LF UTF-8 Ruby 0 files

- The above screenshots demonstrate encapsulation. The first two screenshots show the user attempting to access the attributes of the Guest class (screenshot 1), from a different document and failing (screenshot 2, console).
 - The second two screenshots show the same code but with attribute reader methods added (screenshot 3, line 6), which, if called, allow the user to read (or re-write) those attributes. The same tests now pass (screenshot 4, console).

I.T.2. Use of inheritance:

Atom File Edit View Selection Find Packages Window Help

person.rb — ~/Documents/codeclan_work/week_02/day_5

```
agent_spec.rb medic_spec.rb medic.rb agent.rb person_spec.rb person.rb
```

```
1 class Person
2
3   attr_reader :first_name, :last_name
4
5   def initialize(first_name, last_name)
6     @first_name = first_name
7     @last_name = last_name
8   end
9
10  def say_name
11    return "My name is #{@first_name} #{@last_name}."
12  end
13
14 end
15
```

person.rb* 15:1

LF UTF-8 Ruby 0 files

Atom File Edit View Selection Find Packages Window Help

agent.rb — ~/Documents/codeclan_work/week_02/day_5

```
agent_spec.rb medic_spec.rb medic.rb agent.rb person_spec.rb person.rb
```

```
1 require_relative("person")
2
3 class Agent < Person
4
5   def say_name
6     return "The names #{@last_name}. #{@first_name} #{@last_name}."
7   end
8
9 end
10
```

agent.rb 9:4

LF UTF-8 Ruby 0 files

```

agent_spec.rb          medic_spec.rb          medic.rb           agent.rb           person_spec.rb      person.rb
1  require ("minitest/autorun")
2  require ("minitest/rg")
3  require_relative ("../agent.rb")
4
5
6  class TestAgent < Minitest::Test
7
8    def setup
9      @agent = Agent.new("Chris", "Hontoir")
10   end
11
12  def test_say_name_like_agent
13    assert_equal("The names Hontoir. Chris Hontoir.", @agent.say_name)
14  end
15
16
17 end
18

```

```

week_02 cd day_4
day_4 ls
enumeration.rb enums_practice.rb start_point
day_4 ..
week_02 cd day_5
day_5 ls
agent.rb person.rb weekend_homework
medic.rb specs
day_5 atom
day_5 atom .
day_5 ls
agent.rb person.rb weekend_homework
medic.rb specs
day_5 ruby specs/agent_spec.rb
Run options: --seed 2412
# Running:
.

Finished in 0.001078s, 927.6438 runs/s, 927.6438 assertions/s.

1 runs, 1 assertions, 0 failures, 0 errors, 0 skips
day_5

```

The above screenshots demonstrate the use of class inheritance in a program.

- Screenshot 1 shows the Person class, which has the attributes of `first_name` and `last_name` (lines 5-8).
- Screenshot 2 shows the Agent class, which inherits from the Person class (line 3). Agent also has its own `say_name` method (line 6), which is different from the Person `say_name` method. (The Agent's `say_name` function still uses `first_name` and `last_name`, even though these attributes have not been explicitly initialized in the Agent class.)
- Screenshot 3 demonstrates the function working (see passing test in the console) by creating a new Agent that takes in the `first_name` and `last_name` parameters defined in the Person class. (The Agent defines no parameters explicitly, but takes in these names because it inherits from the Person class.)

P.9. Algorithms:

Atom File Edit View Selection Find Packages Window Help

album.rb — ~/Documents/codeclan_work/project1

Project

project1

db

 inventory.sql

 sql_runner.rb

models

 specs

 album_spec.rb

 artist_spec.rb

 album.rb

 artist.rb

public

views

config.ru

controller.rb

Gemfile

Gemfile.lock

Procfile

seeds.rb

Gemfile

Procfile

inventory... seeds.rb album.rb tables.css index.erb sql_rune... controller... artist.rb edit.erb structure...

16 # if @stock == nil || @stock == 0
17 # @stock = 1
18 # end
19 end
20
21 def check_db()
22 albums = Album.select_1()
23 albums2 = []
24 for album in albums
25 if (album.title == self.title &&
26 album.buying_cost == self.buying_cost &&
27 album.price == self.price &&
28 album.artist_id == self.artist_id)
29 albums2 << album
30 end
31 end
32 if albums2.length == 1
33 albums2[0].update_stock(self, albums2[0])
34 elsif albums2 == []
35 self.save
36 end
37 end
38
39
40 def update_check()

models/album.rb 38:1

LF UTF-8 Ruby ↻ master ⚡ Fetch 0 files



Atom File Edit View Selection Find Packages Window Help

MonopolyBox.js — ~/Documents/codeclan_work/project2/monopoly

MonopolyBox.js PlayerProperty.js Main.js CardDisplay.js Home.js Rules.js App.css PlayerProperty... Winner.js index.css data.js

198 else {
199 this.findNewPositionFromJail();
200 }
201
202 findNewPositionFromJail(){
203 // If double:
204 if (this.state.game.current_roll1 === this.state.game.current_roll2){
205 const total = this.state.game.current_roll1 + this.state.game.current_roll2;
206 const newPosition = this.state.players[this.state.game.current_player].current_position + total;
207 this.setStateHelper("players", this.state.game.current_player, "current_position", newPosition);
208 this.setStateHelper("players", this.state.game.current_player, "jail_counter", 0);
209 }
210 // If failed to roll a double 3 times:
211 else if (this.state.players[this.state.game.current_player].jail_counter === 4){
212 const updatedMoney = this.state.players[this.state.game.current_player].money - 50;
213 const total = this.state.game.current_roll1 + this.state.game.current_roll2;
214 const newPosition = this.state.players[this.state.game.current_player].current_position + total;
215 this.setStateHelper("players", this.state.game.current_player, "money", updatedMoney);
216 this.setStateHelper("players", this.state.game.current_player, "current_position", newPosition);
217 this.setStateHelper("players", this.state.game.current_player, "jail_counter", 0);
218 }
219 }
220
221 passGo(){
222 if (this.state.players[this.state.game.current_player].current_position > 39){

src/containers/MonopolyBox.js 406:11 (1, 8)

LF UTF-8 JavaScript ↻ master ⚡ Fetch 0 files

