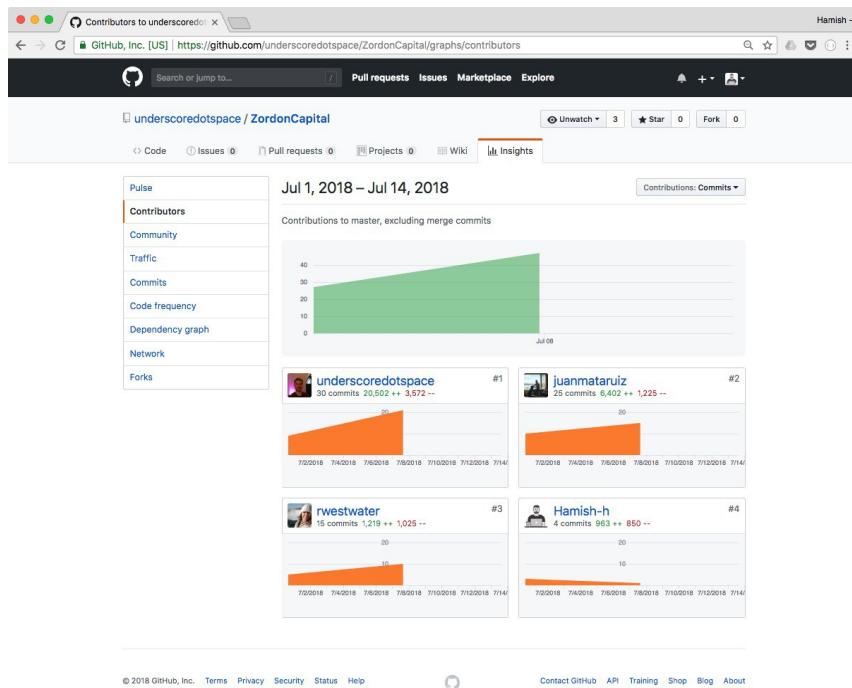


Evidence for Project Unit

Hamish Hoad
Cohort E-21

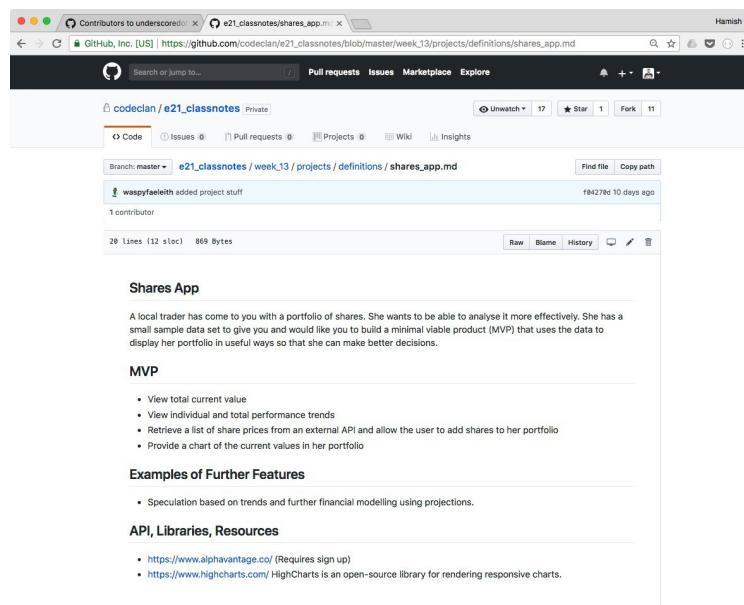
P. 1 Github Contributors page



Evidence for unit

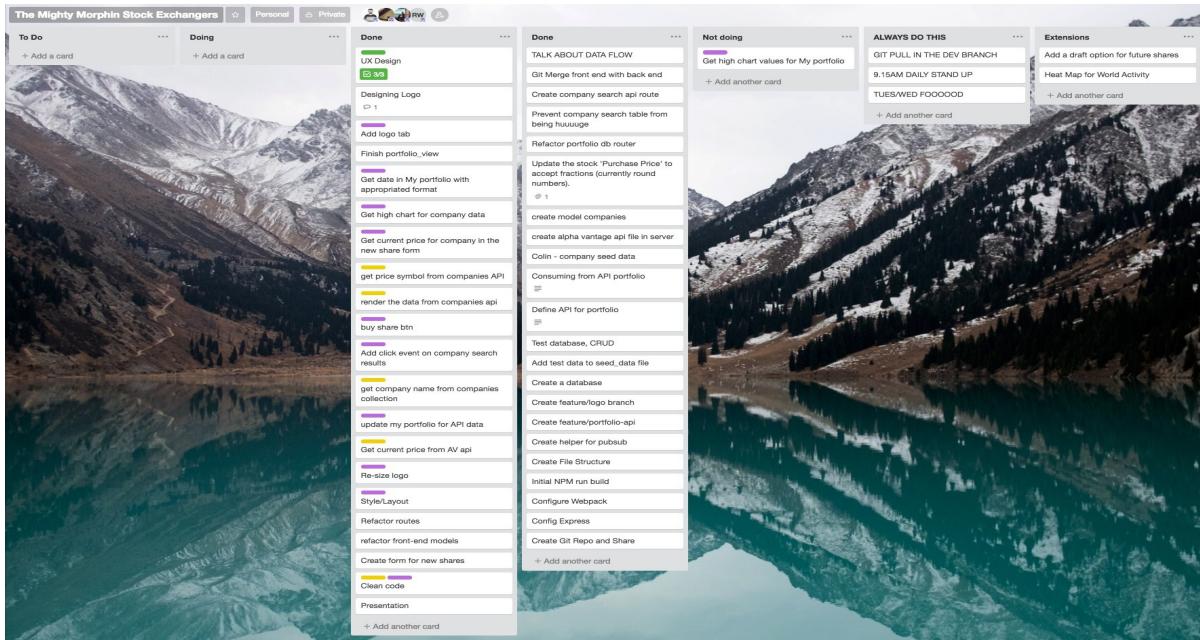
P. 2 Project Brief

Evidence for unit



P. 3 Use of Trello

Evidence for unit



P. 4 Acceptance Criteria

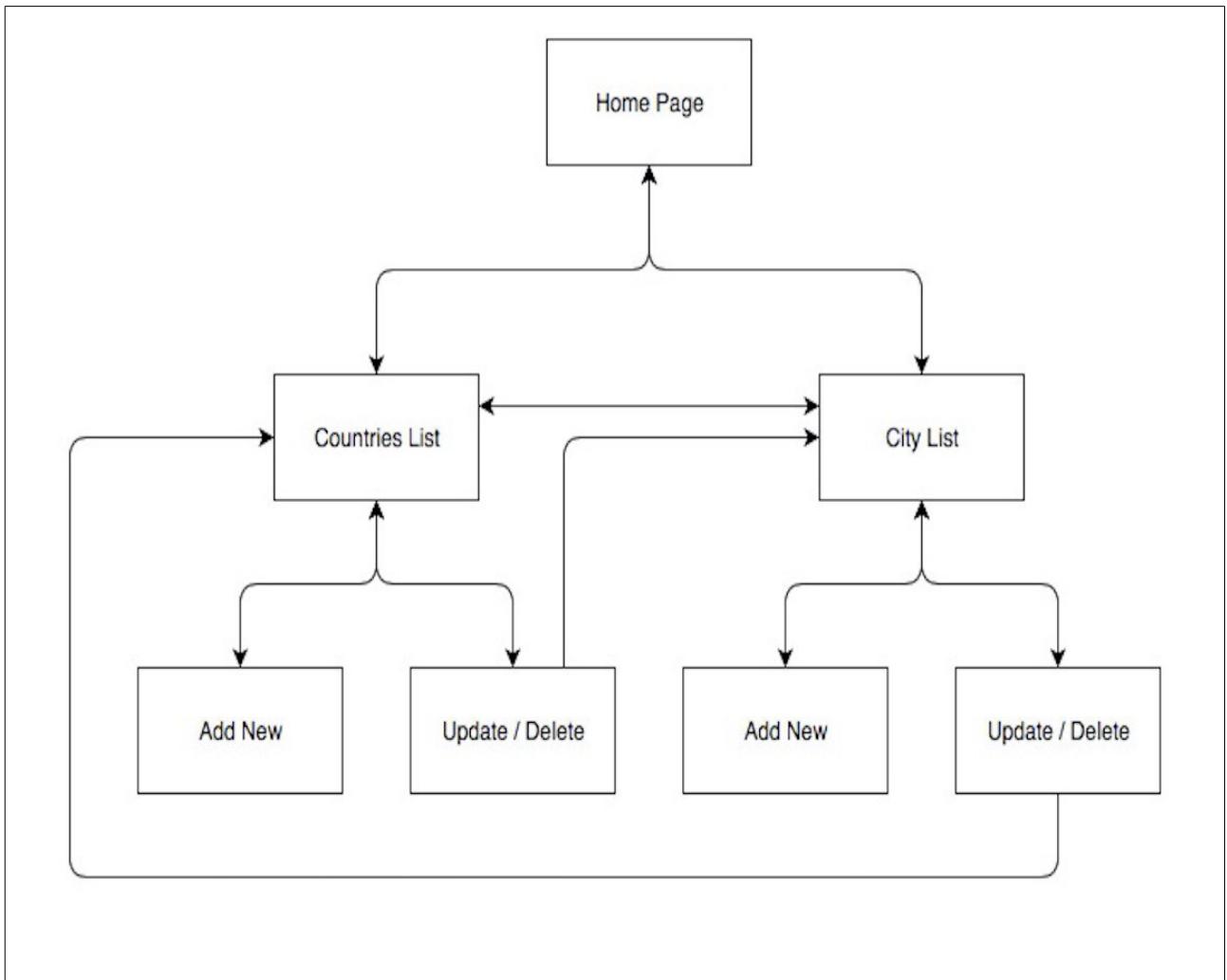
Evidence for unit

Acceptance Criteria and Test Plan

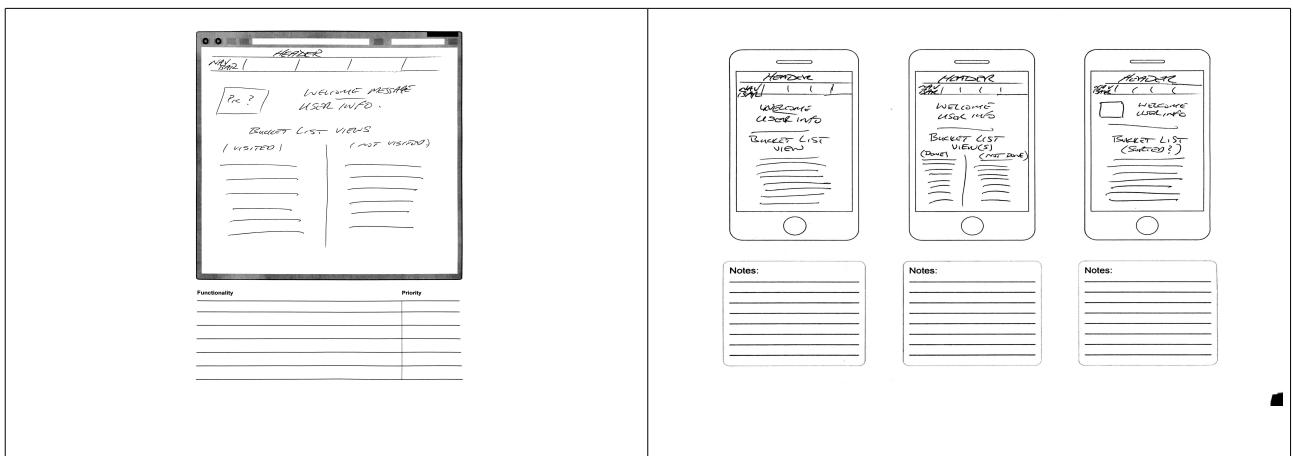
To display and analyse a portfolio of shares in useful ways to make better decisions.

Acceptance Criteria	Expected Result / Output	Pass / Fail
The user should be able to see a portfolio list of shares with the number purchased, the date of purchase, the price paid, the current valuation of each share and the profit and losses that have resulted.	List of purchased shares with the requisite information are displayed when the application opens.	Pass
The user should be able to search for new shares and see the historical share price with its profits and losses to assess the viability for a purchase.	A search box is displayed. The user can search for new shares and access the requisite information.	Pass
The user should be able to purchase shares which are added to the portfolio.	The user can purchase shares which add to the portfolio.	Pass
The user should not be able to purchase shares unless mandatory fields have been completed and a buy process initiated.	Information message appears when attempt to purchase is made without completion of mandatory fields.	Pass

P. 5 User sitemap

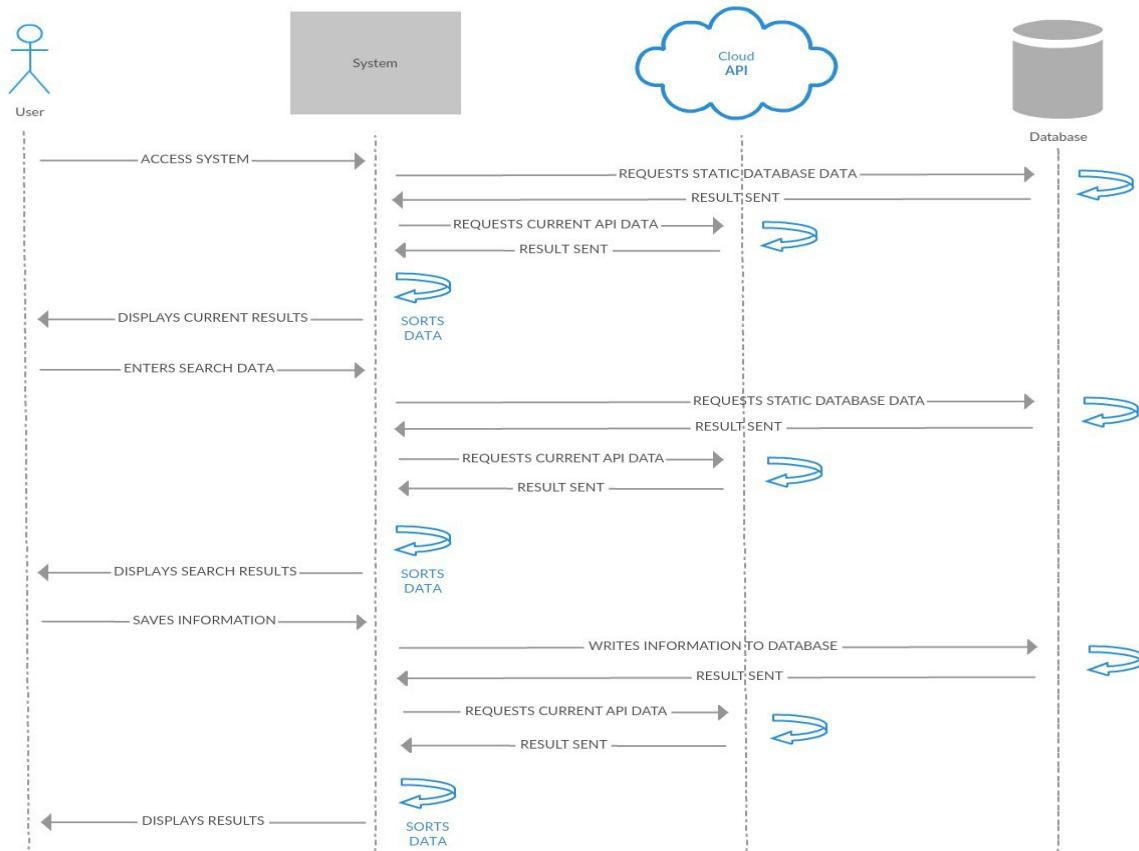


P. 6 Wireframes designs

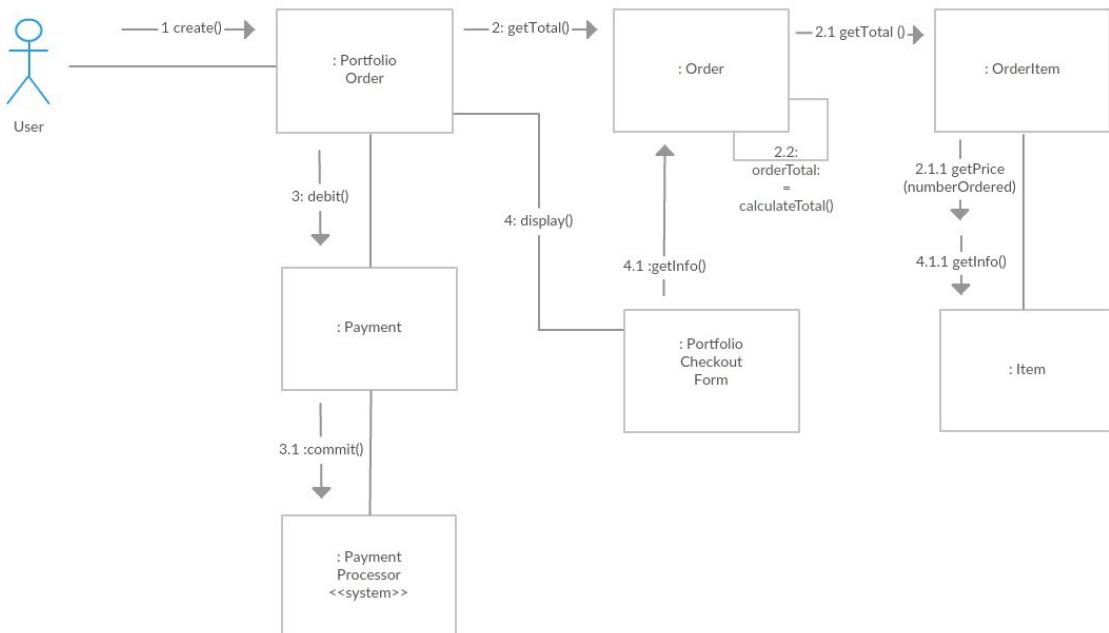


P. 7 System interactions diagrams

Sequence diagram

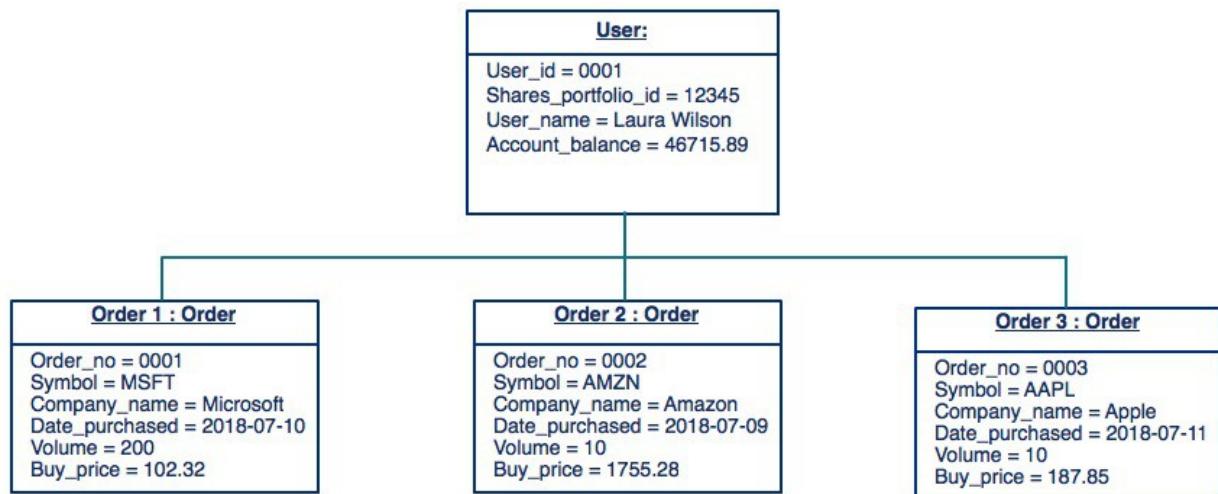


Collaboration diagram



P. 8 Two Object Diagrams

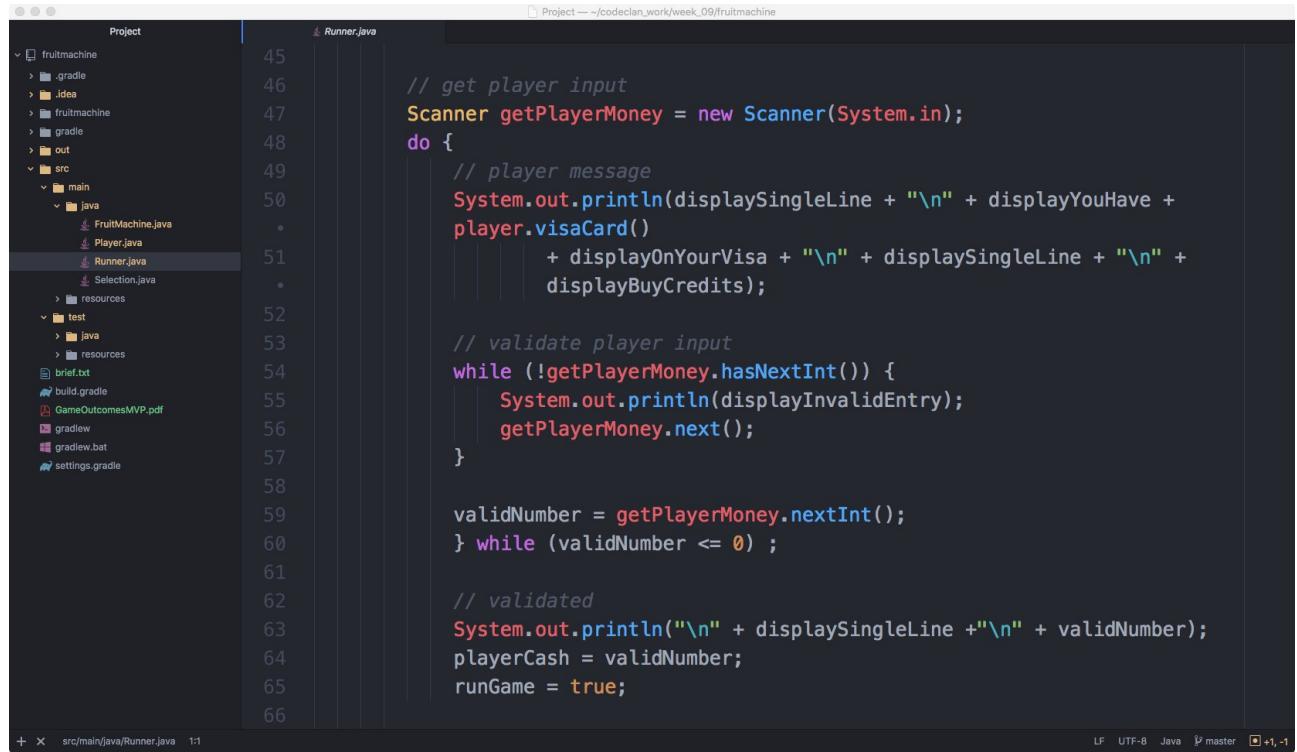
Evidence for unit Evidence for unit



P. 9 Choice of two algorithms (find the algorithms on a program you might have written, show the code you have used.)

On this example please take a screenshot and write what it is doing and why you decided to use it.

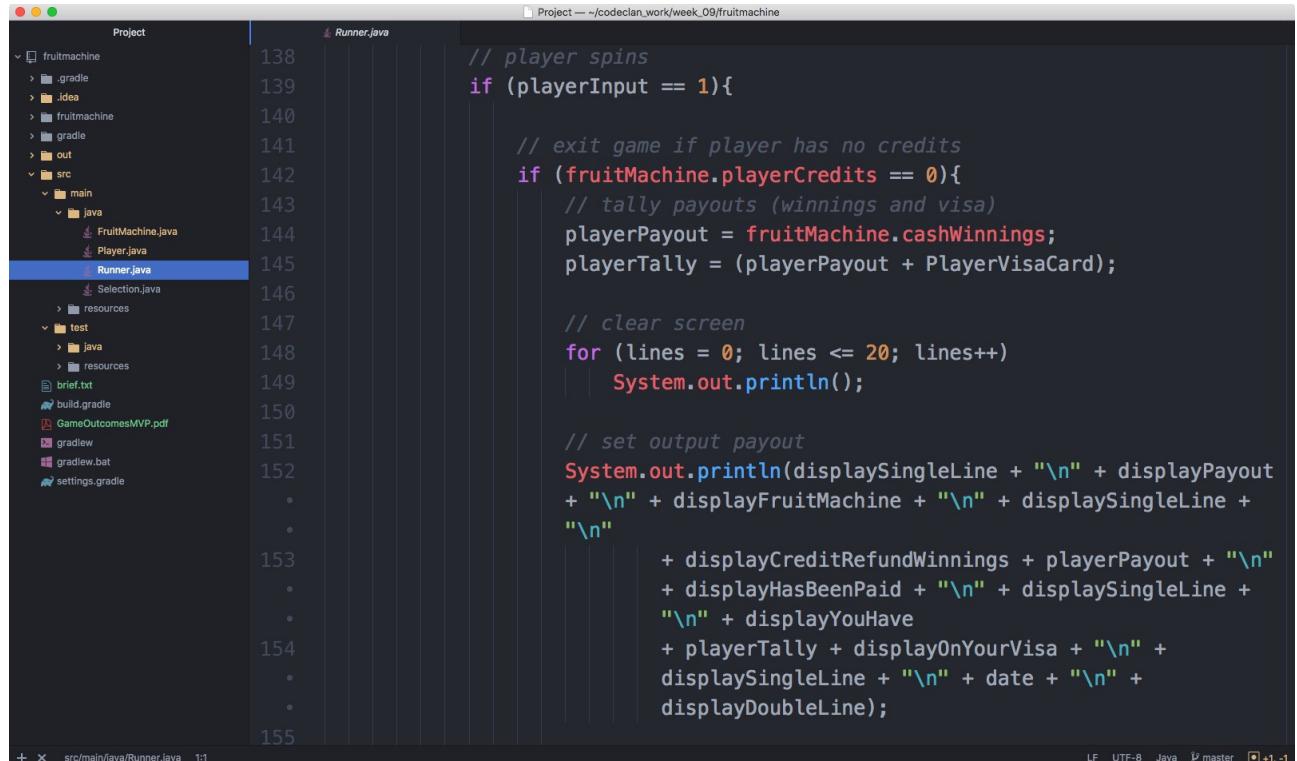
The app is written in Java to mimic the gaming actions of a fruit machine which accepts money and gambles, replicating spinning reels of random fruits with a payout for matching values. The code displays a message to the user, accepts user input and validates that input. **Valid data** proceeds to the game and **Invalid data** returns a request to enter valid data.



A screenshot of an IDE showing the Java code for a fruit machine application. The project structure on the left shows files like .gradle, .idea, gradle, out, src (containing main/java with FruitMachine.java, Player.java, Runner.java, Selection.java), resources, test, and build.gradle. The Runner.java file is open in the editor, showing code from line 45 to 66. The code handles player input, validation, and game setup.

```
45 // get player input
46 Scanner getPlayerMoney = new Scanner(System.in);
47 do {
48     // player message
49     System.out.println(displaySingleLine + "\n" + displayYouHave +
50         player.visaCard() +
51             + displayOnYourVisa + "\n" + displaySingleLine + "\n" +
52             displayBuyCredits);
53
54     // validate player input
55     while (!getPlayerMoney.hasNextInt()) {
56         System.out.println(displayInvalidEntry);
57         getPlayerMoney.nextInt();
58     }
59
60     validNumber = getPlayerMoney.nextInt();
61     } while (validNumber <= 0) ;
62
63     // validated
64     System.out.println("\n" + displaySingleLine +"\n" + validNumber);
65     playerCash = validNumber;
66     runGame = true;
```

The code checks if the user / player has e-money in the fruit machine and if not, cashes out – tallying up any winnings which are displayed as a receipt credited back to his visa card.

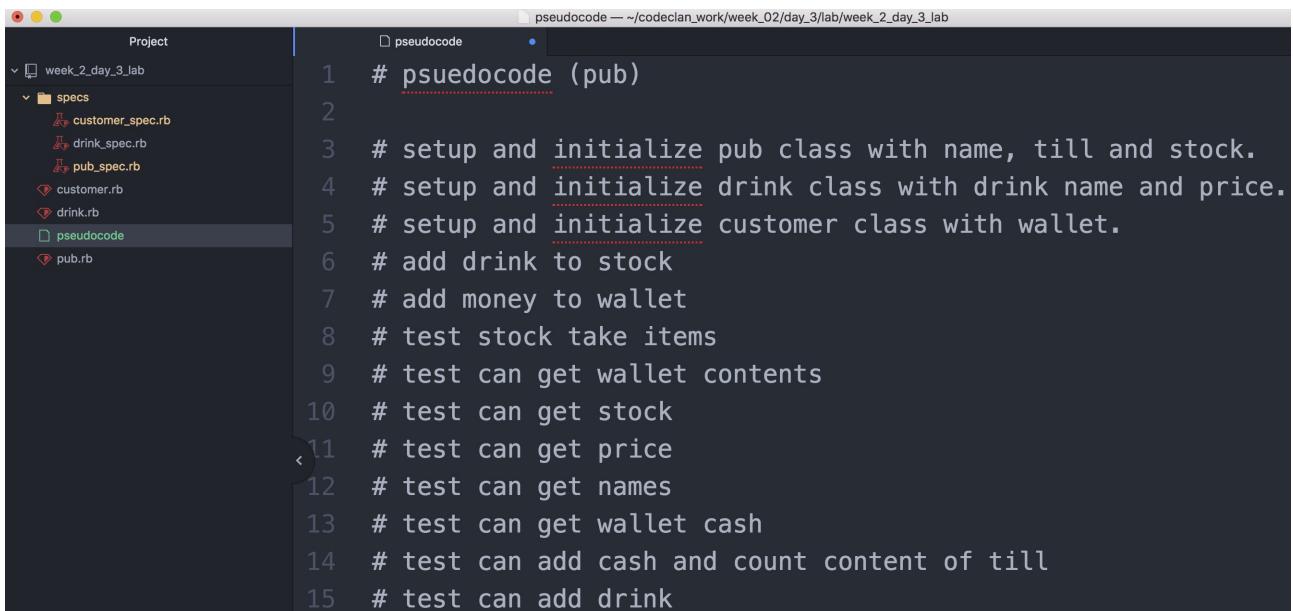


A screenshot of an IDE showing the Java code for a fruit machine application. The project structure on the left shows files like .gradle, .idea, gradle, out, src (containing main/java with FruitMachine.java, Player.java, Runner.java, Selection.java), resources, test, and build.gradle. The Runner.java file is open in the editor, showing code from line 138 to 155. The code handles player spins, exit conditions, and output payout calculations.

```
138 // player spins
139 if (playerInput == 1){
140
141     // exit game if player has no credits
142     if (fruitMachine.playerCredits == 0){
143         // tally payouts (winnings and visa)
144         playerPayout = fruitMachine.cashWinnings;
145         playerTally = (playerPayout + PlayerVisaCard);
146
147         // clear screen
148         for (lines = 0; lines <= 20; lines++)
149             System.out.println();
150
151         // set output payout
152         System.out.println(displaySingleLine + "\n" + displayPayout +
153             + "\n" + displayFruitMachine + "\n" + displaySingleLine +
154                 + "\n" +
155                     + displayCreditRefundWinnings + playerPayout + "\n" +
                     + displayHasBeenPaid + "\n" + displaySingleLine +
                     "\n" + displayYouHave +
                     + playerTally + displayOnYourVisa + "\n" +
                     displaySingleLine + "\n" + date + "\n" +
                     displayDoubleLine);
```

P. 10 Example of Pseudocode

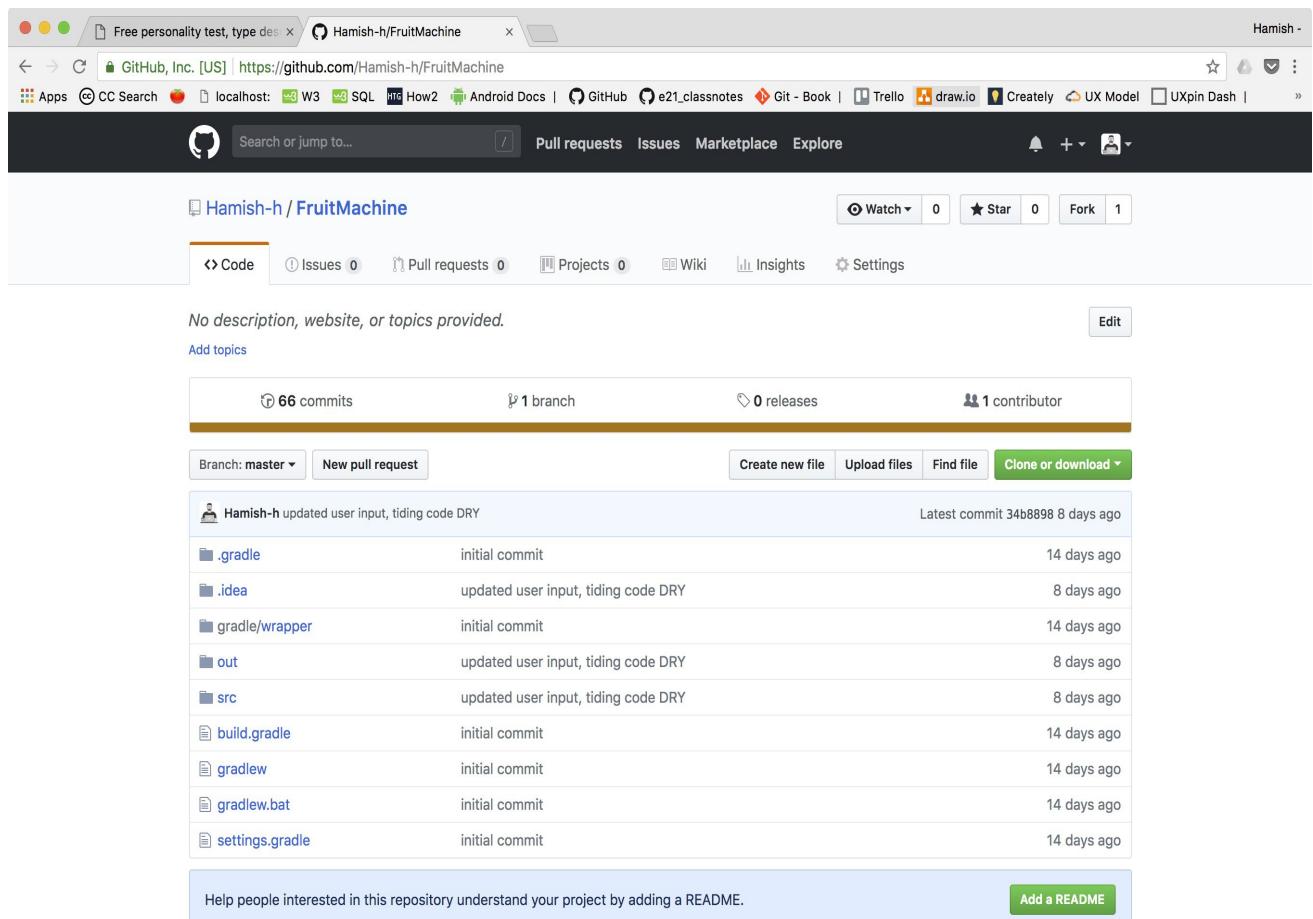
Evidence for unit



```
# psuedocode (pub)
# setup and initialize pub class with name, till and stock.
# setup and initialize drink class with drink name and price.
# setup and initialize customer class with wallet.
# add drink to stock
# add money to wallet
# test stock take items
# test can get wallet contents
# test can get stock
# test can get price
# test can get names
# test can get wallet cash
# test can add cash and count content of till
# test can add drink
```

P. 11 Github link to one of your projects

<https://github.com/Hamish-h/FruitMachine>

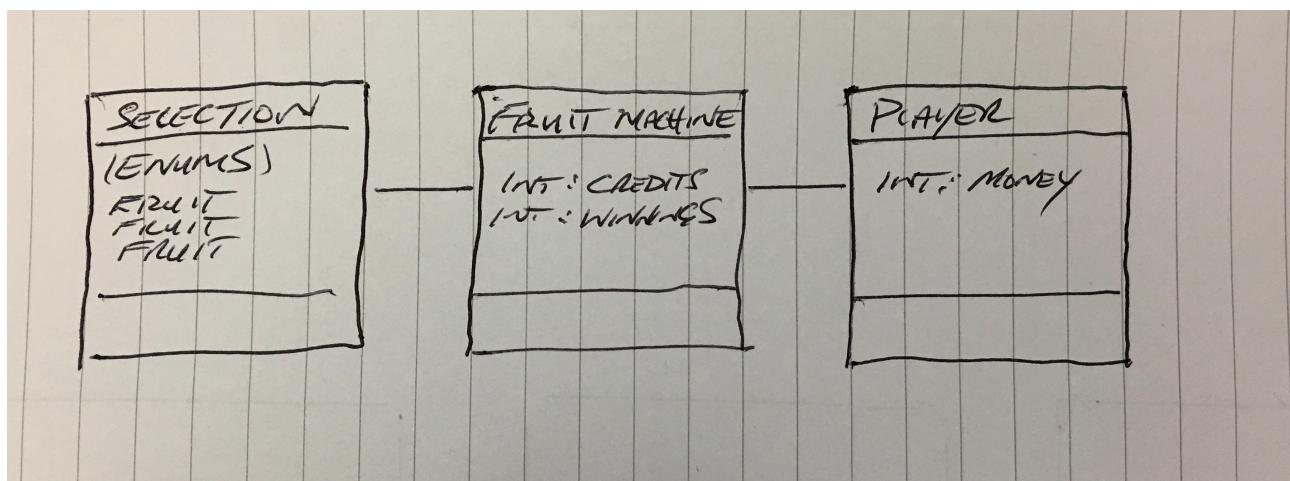


The screenshot shows a GitHub repository page for 'Hamish-h/FruitMachine'. The repository has 66 commits, 1 branch, 0 releases, and 1 contributor. The latest commit was 34b8898, 8 days ago. The repository page includes sections for Code, Issues (0), Pull requests (0), Projects (0), Wiki, Insights, and Settings. There is also a 'New pull request' button. A note at the bottom says 'Help people interested in this repository understand your project by adding a README.' and a 'Add a README' button.

P. 12 Screenshot of your planning and the different stages of development to show changes.

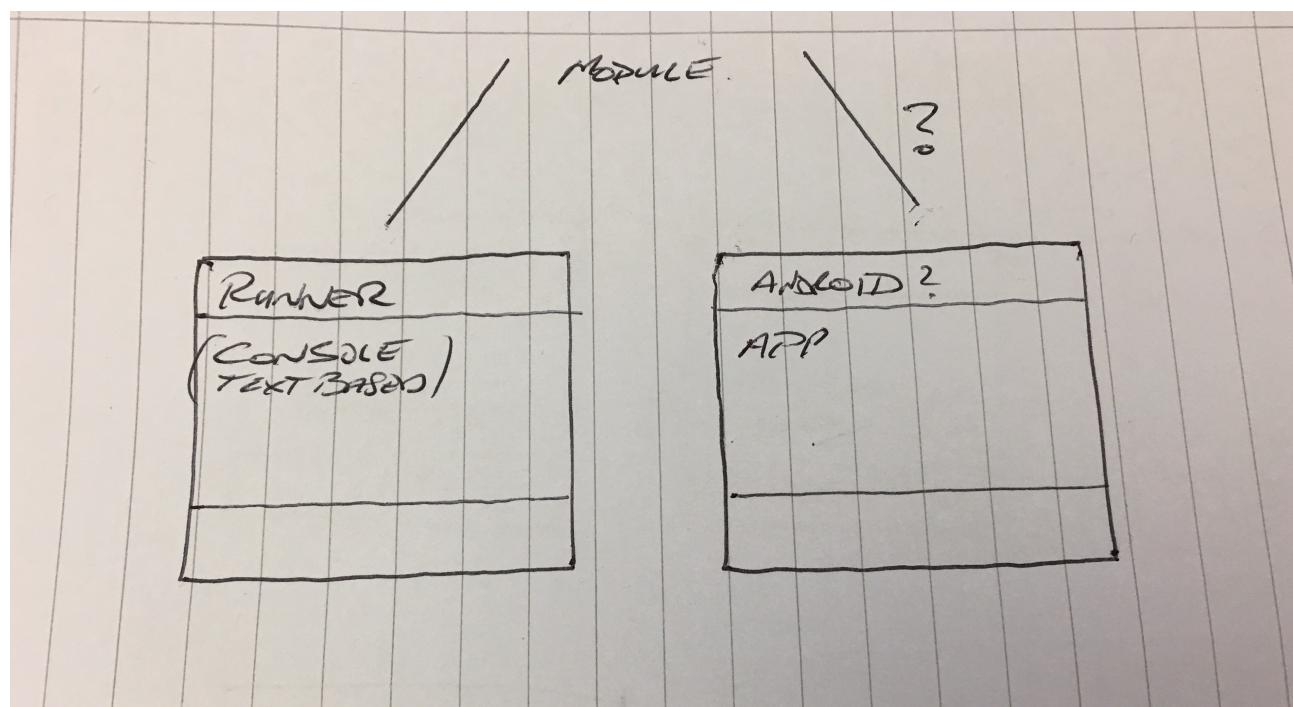
Planning and development – fruit machine

Rough plan

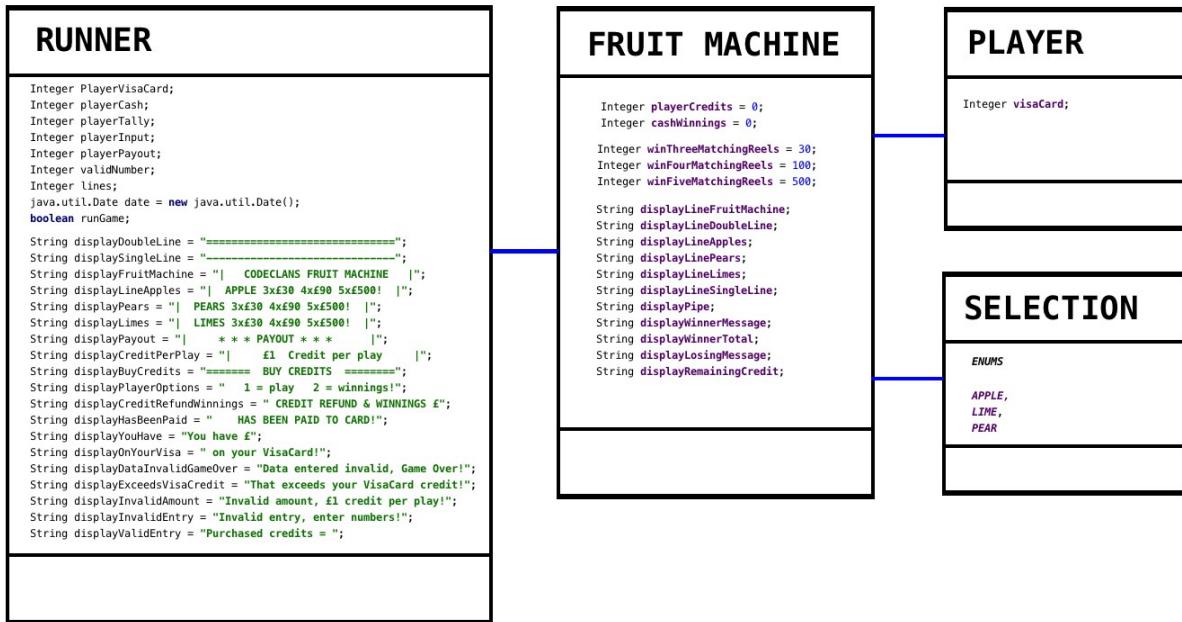


Rough plan addition for the control function

Java runner and Andriod app.



Detailed plan



Coding steps

The screenshot shows the Java code for the `Runner` class in an IDE. The code is identical to the one shown in the Detailed Plan diagram. The execution output window shows the game's interface and the results of a run.

```

import java.util.Scanner;

public class Runner {

    public static void main(String[] args) {
        FruitMachine fruitMachine = new FruitMachine();
        Player player = new Player();

        // runner messages for player
        String displayDoubleLine = "=====";
        String displaySingleLine = "-----";
        String displayFruitMachine = "| CODECLANS FRUIT MACHINE |";
        String displayLineApples = "| APPLE 3xE30 4xE90 5xE500! |";
        String displayPears = "| PEARS 3xE30 4xE90 5xE500! |";
        String displayLines = "| LIMES 3xE30 4xE90 5xE500! |";
        String displayPayout = "| * * * PAYOUT * * * |";
        String displayCreditPerPlay = "| £1 Credit per play |";
        String displayBuyCredits = "===== BUY CREDITS =====";
        String displayPlayerOptions = " 1 = play 2 = winnings!";
        String displayCreditRefundWinnings = " CREDIT REFUND & Winnings £";
        String displayHasBeenCalled = " HAS BEEN PAID TO CARD!";
        String displayYouHave = " You have £";
        String displayOnYourVisa = " on your VisaCard!";
        String displayDataInvalidGameOver = "Data entered invalid, Game Over!";
        String displayExceedsVisaCredit = "That exceeds your VisaCard credit!";
        String displayInvalidAmount = "Invalid amount, £1 credit per play!";
        String displayInvalidEntry = "Invalid entry, enter numbers!";
        String displayValidEntry = "Purchased credits = ";

        // runner values
        Integer PlayerVisaCard;
        Integer playerCash;
    }
}

```

Output:

```

=====
| CODECLANS FRUIT MACHINE |
| APPLE 3xE30 4xE90 5xE500! |
| PEARS 3xE30 4xE90 5xE500! |
| LIMES 3xE30 4xE90 5xE500! |

|PEAR|APPLE|PEAR|LIME|APPLE|
=====

Unlucky this time, try again?

WINNINGS TOTAL £0

Your remaining credit is £59

You have £40 on your VisaCard!

1 = play 2 = winnings!

```

Coding Steps



The screenshot shows an IDE interface with the following details:

- Project Structure:** The project is named "fruitMachine" and contains a package "com.codetan.fruitmachine".
- File List:** The package contains files "FruitMachine.java", "FruitMachineTest.java", and "PlayerTest.java".
- Code Editor:** The main file "FruitMachine.java" is open, showing Java code for a fruit machine game. The code includes methods for displaying lines, generating random choices, and calculating credits based on player input.
- Toolbars and Menus:** Standard Java IDE menus like File, Edit, View, Tools, Window, and Help are visible at the top.
- Status Bar:** The status bar at the bottom shows the file path "FruitMachine.java" and the line number "100".

Coding Steps

```

// WINNING LINES

// winning line of five apples £500
if ((reelOne == Selection.APPLE) && (reelTwo == Selection.APPLE) && (reelThree == Selection.APPLE) && (reelFour == Selection.APPLE) && (reelFive == Selection.APPLE)){ cashWinnings = (cashWinnings + winFiveMatchingReels);
    System.out.println(displayPipe + reelOne + displayPipe + reelTwo + displayPipe + reelThree + displayPipe + reelFour + displayPipe + reelFive + displayPipe + "\n" + displayLineDoubleLine + "\n"
        + displayWinnerMessage + "\n" + displayLineSingleLine + "\n" + displayWinnerTotal + cashWinnings + "\n" + displayLineSingleLine + "\n" + displayRemainingCredit + playerCredits);
}

// winning lines of four apples £100
else if ((reelOne == Selection.APPLE) && (reelTwo == Selection.APPLE) && (reelThree == Selection.APPLE) && (reelFour == Selection.APPLE) ) { cashWinnings = (cashWinnings + 100);
    System.out.println(displayPipe + reelOne + displayPipe + reelTwo + displayPipe + reelThree + displayPipe + reelFour + displayPipe + reelFive + displayPipe + "\n" + displayLineDoubleLine + "\n"
        + displayWinnerMessage + "\n" + displayLineSingleLine + "\n" + displayWinnerTotal + cashWinnings + "\n" + displayLineSingleLine + "\n" + displayRemainingCredit + playerCredits);
}

// winning lines of four apples £100
else if ((reelTwo == Selection.APPLE) && (reelThree == Selection.APPLE) && (reelFour == Selection.APPLE) && (reelFive == Selection.APPLE) ) { cashWinnings = (cashWinnings + 100);
    System.out.println(displayPipe + reelOne + displayPipe + reelTwo + displayPipe + reelThree + displayPipe + reelFour + displayPipe + reelFive + displayPipe + "\n" + displayLineDoubleLine + "\n"
        + displayWinnerMessage + "\n" + displayLineSingleLine + "\n" + displayWinnerTotal + cashWinnings + "\n" + displayLineSingleLine + "\n" + displayRemainingCredit + playerCredits);
}

// winning lines of three apples £30
else if ((reelOne == Selection.APPLE) && (reelTwo == Selection.APPLE) && (reelThree == Selection.APPLE) ) { cashWinnings = (cashWinnings + winThreeMatchingReels);
    System.out.println(displayPipe + reelOne + displayPipe + reelTwo + displayPipe + reelThree + displayPipe + reelFour + displayPipe + reelFive + displayPipe + "\n" + displayLineDoubleLine + "\n"
        + displayWinnerMessage + "\n" + displayLineSingleLine + "\n" + displayWinnerTotal + cashWinnings + "\n" + displayLineSingleLine + "\n" + displayRemainingCredit + playerCredits);
}

// winning Lines of three apples £30
else if ((reelTwo == Selection.APPLE) && (reelThree == Selection.APPLE) && (reelFour == Selection.APPLE) ) { cashWinnings = (cashWinnings + winThreeMatchingReels);
    System.out.println(displayPipe + reelOne + displayPipe + reelTwo + displayPipe + reelThree + displayPipe + reelFour + displayPipe + reelFive + displayPipe + "\n" + displayLineDoubleLine + "\n"
        + displayWinnerMessage + "\n" + displayLineSingleLine + "\n" + displayWinnerTotal + cashWinnings + "\n" + displayLineSingleLine + "\n" + displayRemainingCredit + playerCredits);
}

// winning lines of three apples £30
else if ((reelThree == Selection.APPLE) && (reelFour == Selection.APPLE) && (reelFive == Selection.APPLE) ) { cashWinnings = (cashWinnings + winThreeMatchingReels);
    System.out.println(displayPipe + reelOne + displayPipe + reelTwo + displayPipe + reelThree + displayPipe + reelFour + displayPipe + reelFive + displayPipe + "\n" + displayLineDoubleLine + "\n"
        + displayWinnerMessage + "\n" + displayLineSingleLine + "\n" + displayWinnerTotal + cashWinnings + "\n" + displayLineSingleLine + "\n" + displayRemainingCredit + playerCredits);
}

```

Output

Tests

```
=====| CODECLAN FRUIT MACHINE |=====
| APPLE 3x£30 4x£90 5x£500 ! |
| PEARS 3x£30 4x£90 5x£500 ! |
| LIMES 3x£30 4x£90 5x£500 ! |
=====| PEAR|APPLE|PEAR|LIME|APPLE|
=====

Unlucky this time, try again?

WINNINGS TOTAL £0

Your remaining credit is £59

You have £40 on your VisaCard!

1 = play   2 = winnings!
```

- ▶  **FruitMachineTest**
 - ✓ canGetWinThreeMatchingReels
 - ✓ canGetCashWinnings
 - ✓ CanGameSpin
 - ✓ canGetWinFiveMatchingReels
 - ✓ CanGetAnyRandomAnswerReelFive
 - ✓ CanGetAnyRandomAnswerReelFour
 - ✓ canGetPlayerCredits
 - ✓ CanGetAnyRandomAnswerReelOne
 - ✓ CanGetAnyRandomAnswerReelTwo
 - ✓ canGetWinFourMatchingReels
 - ✓ CanGetAnyRandomAnswerReelThree

P. 13 User input

User inputs and edits a City, adding places to visit.

The screenshot shows a code editor with the file `views/cities/edit.erb` open. The code contains HTML for an edit form and a delete form. To the right, a mobile phone displays the "Edit City" screen for Edinburgh. The screen shows a text input field with "Edinburgh", a textarea with "Visit Edinburgh Castle and the Royal Mile.", and a "Dinner at the Witchery." button. Below the form are navigation buttons for "Update", "Delete", "Cities List", and "Countries List".

```
<div class="form-container">
  <h2>Edit City</h2>
  <form method="POST" action="/cities/<%= @city.id %>">
    <label for="name"></label>
    <input type="text" name="name" id="name" value="<%= @city.name %>" />
    <p><textarea name="review" id="review" rows="6" cols="19"><%= @city.review %></textarea></p>
    <input type="submit" value="Update"/>
  </form>
  <form method="POST" action="/cities/<%= @city.id %>/delete">
    <input type="submit" value="Delete"/>
  </form>
</div>
```

P. 14 Interaction with data persistence

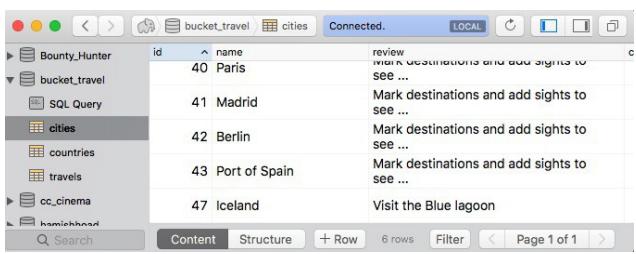
Make sure you show the input being added.

User inputs a City, adds places to visit and saves the information which is then added to the database as shown.

The screenshot shows the same code editor and mobile phone as before. The mobile phone now shows the updated city information: "Visit Edinburgh Castle and the Royal Mile." and "Dinner at the Witchery.". Below the phone is a terminal window showing database logs for the PostgreSQL database. The logs show the insertion of a new row into the "cities" table with id 39 and review "Visit Edinburgh Castle and the Royal Mile.".

```
i:1 -- [14/May/2018:08:44:01 +0100] "POST /cities/39 HTTP/1.1" 303 - 0.0374
i:1 -- [14/May/2018:08:44:01 +0100] "POST /cities/39/edit HTTP/1.1" 303 0
http://localhost:4567/cities/39/edit
i:1 -- [14/May/2018:08:44:01 +0100] "GET /cities/39 HTTP/1.1" 200 1066 0.0103
i:1 -- [14/May/2018:08:44:01 +0100] "POST /cities/39 HTTP/1.1" 200 1066 0.0139
i:1 -- [14/May/2018:08:44:01 +0100] "GET /cities/39/edit HTTP/1.1" 200 1066 0.0151
i:1 -- [14/May/2018:08:42:19 +0100] "GET /cities/39/edit HTTP/1.1" 200 1066 0.0151
i:1 -- [14/May/2018:08:42:22 +0100] "GET /cities/39/edit HTTP/1.1" 200 1110 0.277
i:1 -- [14/May/2018:08:42:22 +0100] "GET /cities/39/edit HTTP/1.1" 200 1110 0.277
http://localhost:4567/cities/39/edit
http://localhost:4567/cities/39/edit
```

P. 15 User output result

 <p>User inputs information and clicks add city which adds information to the database.</p>	 <p>The system then returns a list of cities showing the new city has been added to the database.</p>																																
 <p>The user can then update (or delete) city information, such as a place to visit.</p>	 <table border="1"> <thead> <tr> <th></th> <th>id</th> <th>name</th> <th>review</th> </tr> </thead> <tbody> <tr> <td>Bounty_Hunter</td> <td>40</td> <td>Paris</td> <td>Mark destinations and add sights to see ...</td> </tr> <tr> <td>bucket_travel</td> <td>41</td> <td>Madrid</td> <td>Mark destinations and add sights to see ...</td> </tr> <tr> <td>cities</td> <td>42</td> <td>Berlin</td> <td>Mark destinations and add sights to see ...</td> </tr> <tr> <td>countries</td> <td>43</td> <td>Port of Spain</td> <td>Mark destinations and add sights to see ...</td> </tr> <tr> <td>travels</td> <td>47</td> <td>Iceland</td> <td>Visit the Blue lagoon</td> </tr> <tr> <td>cc_cinema</td> <td></td> <td></td> <td></td> </tr> <tr> <td>homiehhaha</td> <td></td> <td></td> <td></td> </tr> </tbody> </table> <p>The information is then added to the database as shown in the above tables.</p>		id	name	review	Bounty_Hunter	40	Paris	Mark destinations and add sights to see ...	bucket_travel	41	Madrid	Mark destinations and add sights to see ...	cities	42	Berlin	Mark destinations and add sights to see ...	countries	43	Port of Spain	Mark destinations and add sights to see ...	travels	47	Iceland	Visit the Blue lagoon	cc_cinema				homiehhaha			
	id	name	review																														
Bounty_Hunter	40	Paris	Mark destinations and add sights to see ...																														
bucket_travel	41	Madrid	Mark destinations and add sights to see ...																														
cities	42	Berlin	Mark destinations and add sights to see ...																														
countries	43	Port of Spain	Mark destinations and add sights to see ...																														
travels	47	Iceland	Visit the Blue lagoon																														
cc_cinema																																	
homiehhaha																																	

P. 16 Show an API being used within your program.

The code that uses / implements the API.

```

1 const Request = require('../helpers/request_helper.js');
2 const PubSub = require('../helpers/pub_sub.js');
3
4 const QuizMaster = function () {
5   // null or "" empty string
6   this.text = null;
7 }
8 // get data
9 QuizMaster.prototype.getData = function () {
10   // set the url
11   const request = new Request('https://opentdb.com/api.php?amount=1&difficulty=medium&type=multiple');
12   // oncomplete
13   request.get((data) => {
14     // console.log("request get results ", data);
15     // collect and assign data
16     this.category = data.results[0].category;
17     this.question = data.results[0].question;
18     this.incorrect_answers = data.results[0].incorrect_answers;
19     this.correct_answer = data.results[0].correct_answer;
20     // console.log("this.category", this.category);
21     // publish category
22     PubSub.publish('QuizMaster:quizMaster-loaded', this.category);
23     console.log(this.category);
24     console.log(this.question);
25     // publish
26     PubSub.publish('QuizMaster:quizMaster-loaded', this.question);
27     // publish incorrect answers
28     PubSub.publish('QuizMaster:quizMaster-loaded', this.incorrect_answers);
29     // publish correct answer
30     PubSub.publish('QuizMaster:quizMaster-loaded', this.correct_answer);
31   });
32 }
33 // export
34 module.exports = QuizMaster;

```

The API being used by the program while running.

```
1 const Request = require('../helpers/request_helper.js');
2 const PubSub = require('../helpers/pub_sub.js');
3
4 const QuizMaster = function () {
5   // null or "" empty string
6   this.text = null;
7 }
8 // get data
9 QuizMaster.prototype.getData = function () {
10   // set the url
11   const request = new Request('https://opentdb.com/api.php?amount=1&difficulty=medium&type=multiple');
12   // oncomplete
13   request.get((data) => {
14     // console.log("request get results ", data);
15     // collect and assign data
16     this.category = data.results;
17     this.question = data.results;
18     this.incorrect_answers = data.results;
19     this.correct_answer = data.results;
20
21   npm run build — npm — node -n pm
22 [.src/models/quizmaster.js] 1.21 KiB {main} [built]
23 [.src/views/quizmaster_view.js] 823 bytes {main} [built]
24
25
26 Webpack is watching the files...
27
28 Hash: f3d7f194be2d3fb4434
29 Version: webpack 4.13.0
30 Time: 88ms
31 Built at: 16/07/2018 00:34:48
32 Asset Size Chunks Names
33 bundle.js 8.62 KiB main [emitted] main
34 [.src/app.js] 455 bytes {main} [built]
35 [.src/helpers/pub_sub.js] 301 bytes {main} [built]
36 [.src/helpers/request_helper.js] 397 bytes {main} [built]
37 [.src/models/quizmaster.js] 1.21 KiB {main} [built]
38 [.src/views/quizmaster_view.js] 823 bytes {main} [built]
39
40
41 mongod — mongod — mongod
42
43 te access to data and configuration is unrestricted.
44 2018-07-16T00:34:43.552+0100 I CONTROL [initandlisten]
45 2018-07-16T00:34:43.552+0100 I CONTROL [initandlisten] ** WARNING: This server
46 is bound to localhost.
47 2018-07-16T00:34:43.552+0100 I CONTROL [initandlisten] ** Remote syste
48 ms will be unable to connect to this server.
49 2018-07-16T00:34:43.552+0100 I CONTROL [initandlisten] ** Start the se
50 rver with --bind_ip <address> to specify which IP
51 2018-07-16T00:34:43.552+0100 I CONTROL [initandlisten] ** addresses it
52 should serve responses from, or with --bind_ip_all to
53 2018-07-16T00:34:43.552+0100 I CONTROL [initandlisten] ** bind to all
54 interfaces. If this behavior is desired, start the se
55 2018-07-16T00:34:43.552+0100 I CONTROL [initandlisten] ** server with
56 --bind_ip 127.0.0.1 to disable this warning.
57 2018-07-16T00:34:43.552+0100 I CONTROL [initandlisten]
58 2018-07-16T00:34:43.552+0100 I CONTROL [initandlisten]
59 2018-07-16T00:34:43.552+0100 I CONTROL [initandlisten] ** WARNING: soft rlimits
60 too low. Number of files is 256, should be at least 1000
61 2018-07-16T00:34:43.569+0100 I FTDC [initandlisten] Initializing full-time d
62 iagnostic data capture with directory '/data/db/diagnostic.data'
63 2018-07-16T00:34:43.569+0100 I NETWORK [initandlisten] waiting for connections
64 on port 27017
65
```

P. 17 Bug tracking report showing the errors diagnosed and corrected.

Evidence for unit

Bug Tracking Report

Record of bugs in a Javascript Software Product.

The back-end server models connect to the front end public src form views.	Failed	Corrected spelling mistake in the server/models module.exports name.	Passed
The database data populates views.	Failed	Updated bracket formatting of the seed_data JSON file and repopulated the database.	Passed
The API data populates the information on user screen.	Failed	Updated constructor require URL path to the API file.	Passed
The app calculates a price for an item.	Failed	Updated the item name from lower case to camel case.	Passed
The user input value box should prevent negative numbers.	Failed	Restricted user input to a number equal to or greater than 1.	Passed
The html form populated with API data should not be overwritten.	Failed	Amended the html input type value to equal disabled.	Passed

P. 18 Testing your program

Show the test code, the test not passing.....and then the test fixed.

Example of test code

with the test failing to pass

NOT Passing

```

Project
└── weekend_homework
    ├── PDA_Static_and_Dynamic_Task_A
    │   ├── spec
    │   │   └── testing_task_2_spec.rb
    │   ├── card.rb
    │   ├── Static_A_Dynamic_Testing.md
    │   └── testing_task_1.rb
    └── DS_Store

    └── PDA_Static_and_Dynamic_Task_A.zip

card.rb
1  class Card
2      attr_reader :suit, :value
3
4
5  def initialize(suit, value)
6      @suit = suit
7      @value = value;
8  end
9 end
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31

testing_task_2.rb
6  require_relative('card.rb')
7  class CardGame
8
9
10  def checkforAce(card)
11      if card.value == 1
12          return true
13      else
14          return false
15      end
16  end
17
18  def highest_card(card1, card2)
19      if card1.value > card2.value
20          return card1.name
21      else
22          card2
23      end
24
25  def self.cards_total(cards)
26      total
27      for card in cards
28          total += card.value
29      end
30  end
31

testing_task_2_spec.rb
8  class CardGameTest < MiniTest::Test
9
10  def setup()
11      @card = CardGame.new("Clubs", 1)
12  end
13
14  def test_check_for_ace
15      assert_equal("Clubs", @card.value)
16  end
17
18  end
19

hamish@hamish-MBP:~/dynamic_Task_A$ ruby testing_task_2.rb
from specs/testing_task_2_spec.rb:6:in `require_relative'
  from specs/testing_task_2_spec.rb:6:in `<main>'
PDA_Static_and_Dynamic_Task_A (Gem::LoadError)
  require_relative: /Users/hamish/Downloads/weekend_homework/PDA_Static_and_Dynamic_Task_A/spec/testing_task_2.rb:6:in `from_file'
  require_relative: /Users/hamish/Downloads/weekend_homework/PDA_Static_and_Dynamic_Task_A/spec/testing_task_2.rb:6:in `<main>'
syntax error, unexpected IDBNITF18, expecting ',' (SyntaxError)
  from specs/testing_task_2_spec.rb:6:in `<main>'

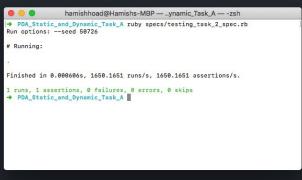
hamish@hamish-MBP:~/dynamic_Task_A$ ruby testing_task_2_spec.rb:10:in `require_relative': /Users/hamish/Downloads/weekend_homework/PDA_Static_and_Dynamic_Task_A/spec/testing_task_2.rb:10:in `from_file'
  require_relative: /Users/hamish/Downloads/weekend_homework/PDA_Static_and_Dynamic_Task_A/spec/testing_task_2.rb:10:in `<main>'
syntax error, unexpected end-of-input, expecting keyword_end (SyntaxError)
  from specs/testing_task_2_spec.rb:10:in `<main>'

hamish@hamish-MBP:~/dynamic_Task_A$
```

Example of test code once errors have been corrected

with the test then passing, fixed

Passing



The screenshot shows a terminal window with the following output:

```
HannishHamis@Hannishs-MBP:~/yarnic/Task_A$ ./bin/rake test
Run options: --seed 48725
# Running:
.
Finished in 0.000000s, 1659.1651 runs/s, 1659.1651 assertions/s.
1 runs, 1 assertions, 0 failures, 0 errors, 0 skips
+ POM_Static_and_Dynamic_Task_A
```

Project structure (left):

- Project
- └ weekend homework
- └ POM_Static_and_Dynamic_Task_A
- └ spec
- └ testing_task_2_spec.rb
- └ card.rb
- └ cardGame.rb
- └ testing_task_2.rb
- └ DB_Store
- └ StaticAndDynamicTesting.md
- └ testing_task_1.rb
- └ testing_task_2.rb
- └ DB_Store
- └ POM_Static_and_Dynamic_Task_A.rb

Code snippets (center and right):

card.rb (left):

```
1 class Card
2   attr_reader :suit, :value
3
4   def initialize(suit, value)
5     @suit = suit
6     @value = value
7   end
8 end
9
10
```

cardGame.rb (center):

```
5 require_relative('card.rb')
6 class CardGame
7
8   def check_for_ace(card)
9     if card.value == 1
10      return true
11    else
12      return false
13    end
14  end
15
16  def highest_card(card1, card2)
17    if card1.value > card2.value
18      return card1.name
19    else
20      card2
21    end
22  end
23
24  def self.cards_total(cards)
25    total
26    for card in cards
27      total += card.value
28    end
29    return "You have a total of" + total
30  end
31
32  end
33
```

testing_task_2.rb (right):

```
1 #!/usr/bin/env ruby
2 require('minitest/autorun')
3 require('minitest/rg')
4
5 require_relative('../card.rb')
6 require_relative('../testing_task_2.rb')
7
8 class CardGameTest < MiniTest::Test
9
10  def setup()
11    @card1 = Card.new("Clubs", 1)
12    @card2 = Card.new("Clubs", 2)
13  end
14
15  def test_check_for_ace
16    assert_equal(1, @card1.value)
17  end
18
19 end
```