

## **Evidence for Project Unit**

**Hamish Hoad**

**Cohort E-21**

### **P. 1 Github Contributors page**

Evidence for unit

### **P. 2 Project Brief**

Evidence for unit

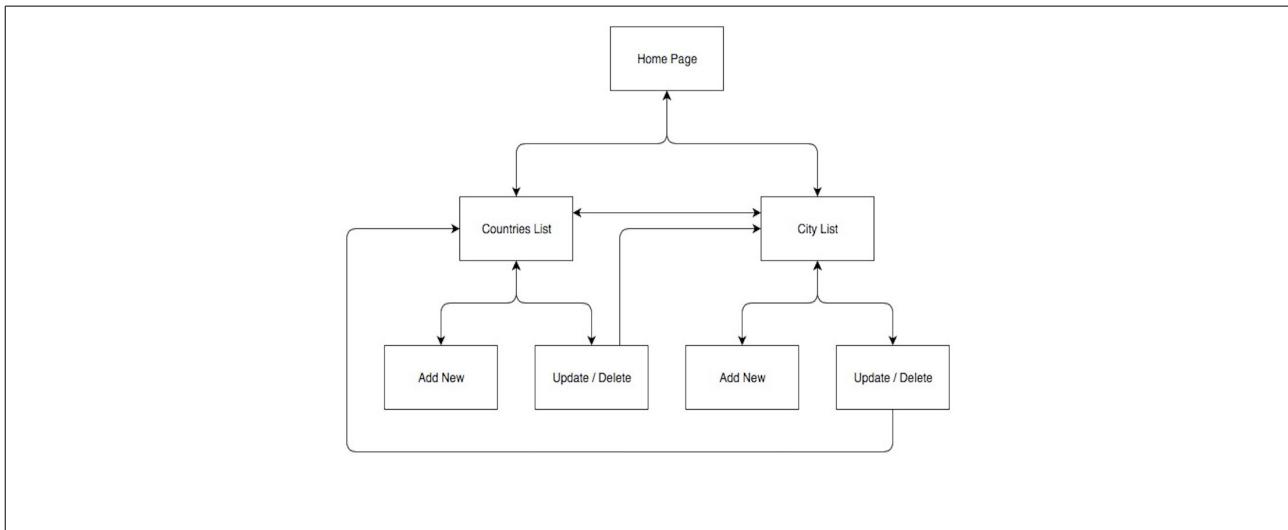
### **P. 3 Use of Trello**

Evidence for unit

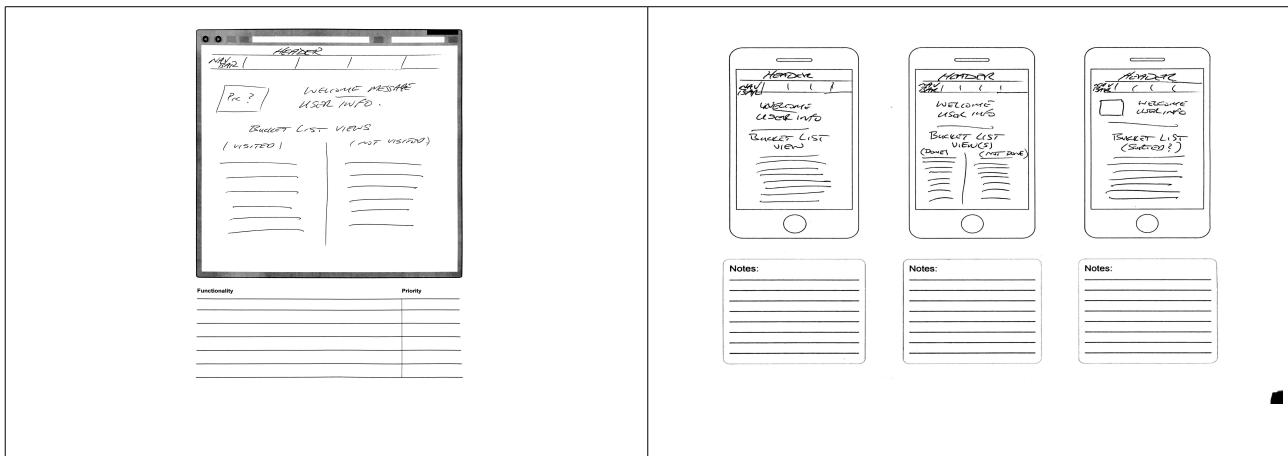
### **P. 4 Acceptance Criteria**

Evidence for unit

## P. 5 User sitemap



## P. 6 Wireframes designs



## P. 7 System interactions diagrams

Evidence for unit

Evidence for unit

## P. 8 Two Object Diagrams

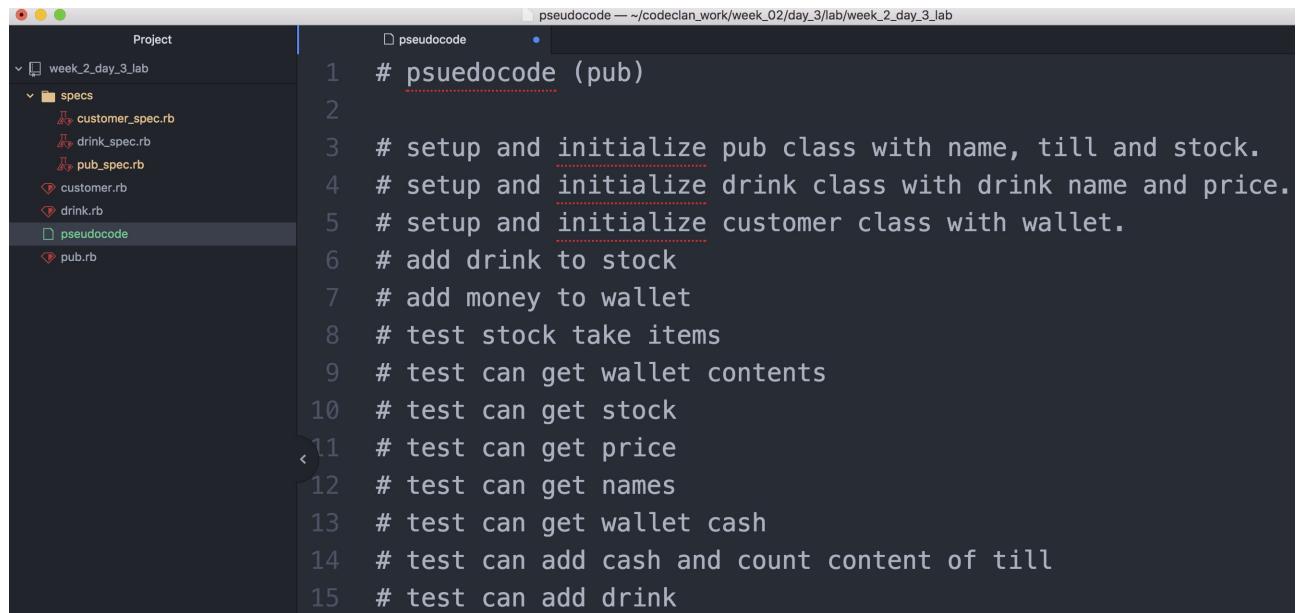
Evidence for unit Evidence for unit

**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.**

## **P. 10 Example of Pseudocode**

Evidence for unit



A screenshot of a terminal window titled "pseudocode" showing pseudocode for a "pub" class. The pseudocode lists various methods and their descriptions:

```
1 # psuedocode (pub)
2
3 # setup and initialize pub class with name, till and stock.
4 # setup and initialize drink class with drink name and price.
5 # setup and initialize customer class with wallet.
6 # add drink to stock
7 # add money to wallet
8 # test stock take items
9 # test can get wallet contents
10 # test can get stock
11 # test can get price
12 # test can get names
13 # test can get wallet cash
14 # test can add cash and count content of till
15 # test can add drink
```

## **P. 11 Github link to one of your projects**

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

No description, website, or topics provided.

Add topics

66 commits 1 branch 0 releases 1 contributor

Branch: master New pull request

Hamish-h updated user input, tidying code DRY Latest commit 34b8898 8 days ago

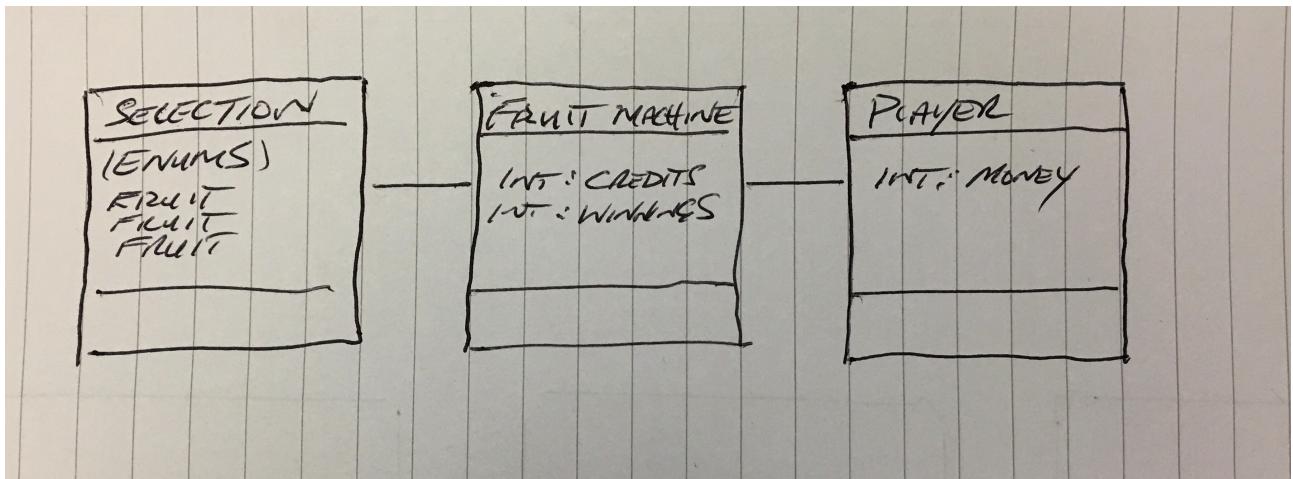
- .gradle initial commit 14 days ago
- .idea updated user input, tidying code DRY 8 days ago
- gradle/wrapper initial commit 14 days ago
- out updated user input, tidying code DRY 8 days ago
- src updated user input, tidying code DRY 8 days ago
- build.gradle initial commit 14 days ago
- gradlew initial commit 14 days ago
- gradlew.bat initial commit 14 days ago
- settings.gradle initial commit 14 days ago

Help people interested in this repository understand your project by adding a README. Add a README

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

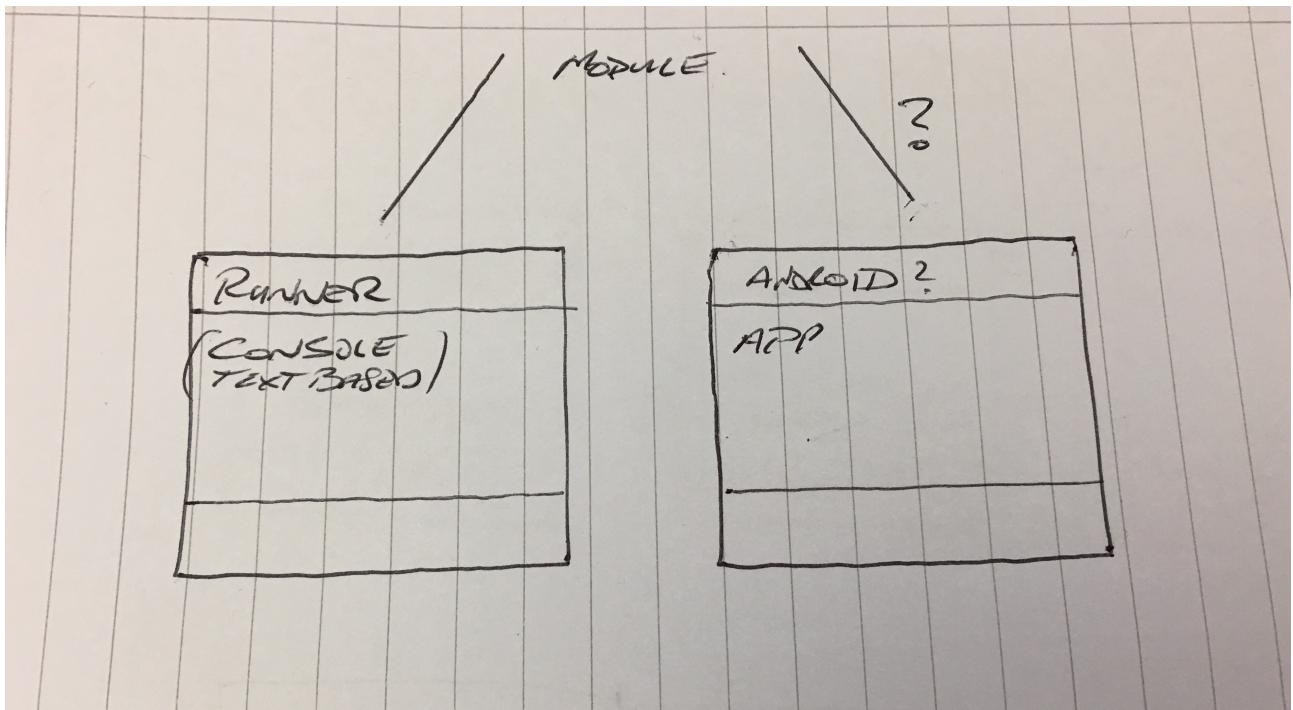
Planning and development – fruit machine

Rough plan

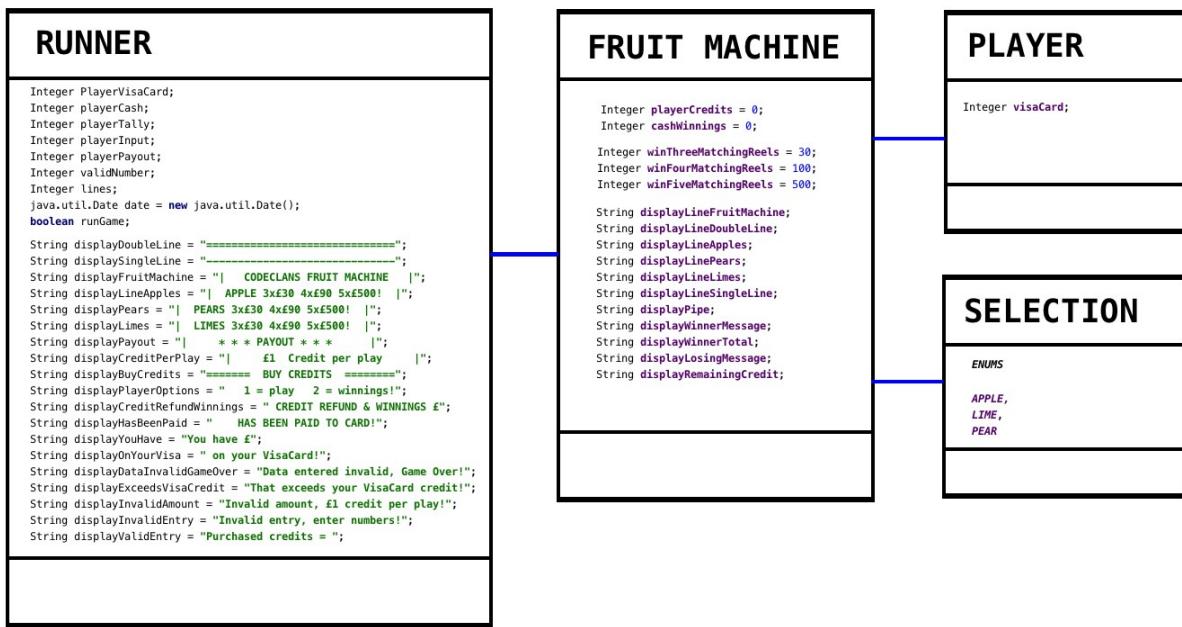


Rough plan for the control function

Java runner and Andriod app.



## Detailed plan



## Coding steps

The screenshot shows an IDE interface with the following details:

- Project Structure:** The project is named "fruitmachine". The current file is "Runner.java" located in the "src/main/java" directory.
- Code Editor:** The code for "Runner.java" is displayed. It imports "java.util.Scanner" and defines a class "Runner" with a static main method. The main method initializes a "FruitMachine" object and a "Player" object. It then displays various messages and values related to the fruit machine, such as "CODECLANS FRUIT MACHINE", "APPLE 3xE30 4xE90 5xE500!", "PEARS 3xE30 4xE90 5xE500!", "LIMES 3xE30 4xE90 5xE500!", and payout options like "1 = play 2 = winnings!". It also handles invalid entries and credit management.
- Output Window:** The bottom pane shows the terminal output of the program. It starts with the fruit machine's logo, followed by a sequence of symbols (|PEAR|APPLE|PEAR|LIME|APPLE|). A message "Unlucky this time, try again?" is displayed, followed by "WINNINGS TOTAL £0", "Your remaining credit is £59", and "You have £40 on your VisaCard!". Finally, the command "1 = play 2 = winnings!" is shown.
- Bottom Bar:** The bar includes icons for Version Control, Terminal, Run, TODO, and Event Log. It also shows a status message "All files are up-to-date (13 minutes ago)" and a timestamp "41:1 LF: UTF-8 Git: master".

## Coding Steps

```

FruitMachine.java - com.codeclan.fruitmachine - (~/codeclan_work/week_09/fruitmachine)
frutemachine (com.codeclan.fruitmachine) ->codeclan
  +-- Project
    +-- src
      +-- main
        +-- java
          +-- com
            +-- codeclan
              +-- fruitmachine
                +-- FruitMachine.java
      +-- test
        +-- java
          +-- com
            +-- codeclan
              +-- fruitmachine
                +-- FruitMachineTest.java
      +-- resources
      +-- gradle
        +-- build.gradle
        +-- gradlew
        +-- gradlew.bat
      +-- settings.gradle
  +-- External Libraries

private Selection randomiseReelChoice() {
    Selection[] array = {Selection.APPLE, Selection.LIME, Selection.PEAR};
    Random choice = new Random();
    int selected = choice.nextInt(array.length);
    return array[selected];
}

public void generateAllChoices(){
    this.setReelOneChoice();
    this.setReelTwoChoice();
    this.setReelThreeChoice();
    this.setReelFourChoice();
    this.setReelFiveChoice();
}

public void setReelOneChoice() { this.reelOneChoice = randomiseReelChoice(); }
public void setReelTwoChoice() { this.reelTwoChoice = randomiseReelChoice(); }
public void setReelThreeChoice() { this.reelThreeChoice = randomiseReelChoice(); }
public void setReelFourChoice() { this.reelFourChoice = randomiseReelChoice(); }
public void setReelFiveChoice() { this.reelFiveChoice = randomiseReelChoice(); }

public Selection getReelOneChoice() { return reelOneChoice; }
public Selection getReelTwoChoice() { return reelTwoChoice; }
public Selection getReelThreeChoice() { return reelThreeChoice; }
public Selection getReelFourChoice() { return reelFourChoice; }
public Selection getReelFiveChoice() { return reelFiveChoice; }

public int getNumberOfPlayerCredits() { return playerCredits; }
public int getWinningReels() { return winMatchingReels; }
public int getWinThreeMatchingReels() { return winThreeMatchingReels; }
public int getWinFourMatchingReels() { return winFourMatchingReels; }
public int getWinFiveMatchingReels() { return winFiveMatchingReels; }

// reel choice
public String getReelOneReel() {
    Selection reelOne = getReelOneChoice();
    Selection reelTwo = getReelTwoChoice();
    Selection reelThree = getReelThreeChoice();
    Selection reelFour = getReelFourChoice();
    Selection reelFive = getReelFiveChoice();

    // deduct credits
    playerCredits -= 1;

    // display output
    System.out.println(displayLineDoubleLine + "\n" + displayLineFruitMachine + "\n" + displayLinePears + "\n" + displayLineLimes + "\n" + displayLineDoubleLine);

    // WINNING LINES
    if ((reelOne == Selection.APPLE && reelTwo == Selection.APPLE && reelThree == Selection.APPLE && reelFour == Selection.APPLE && reelFive == Selection.APPLE)) { cashWinnings = (cashWinnings + winFiveMatchingReels);
        System.out.println(displayLine + 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 line of five apples £500
    else if ((reelOne == Selection.APPLE && reelTwo == Selection.APPLE && reelThree == Selection.APPLE && reelFour == Selection.APPLE && reelFive == Selection.APPLE)) { cashWinnings = (cashWinnings + 500);
        System.out.println(displayLine + 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) || (reelOne == Selection.APPLE && reelTwo == Selection.APPLE && reelThree == Selection.APPLE && reelFive == Selection.APPLE)) { cashWinnings = (cashWinnings + 100);
        System.out.println(displayLine + 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) || (reelOne == Selection.APPLE && reelTwo == Selection.APPLE && reelThree == Selection.APPLE && reelFive == Selection.APPLE)) { cashWinnings = (cashWinnings + 100);
        System.out.println(displayLine + 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) || (reelOne == Selection.APPLE && reelTwo == Selection.APPLE && reelFive == Selection.APPLE) || (reelOne == Selection.APPLE && reelThree == Selection.APPLE && reelFive == Selection.APPLE) || (reelTwo == Selection.APPLE && reelThree == Selection.APPLE && reelFive == Selection.APPLE)) { cashWinnings = (cashWinnings + winThreeMatchingReels);
        System.out.println(displayLine + 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) || (reelOne == Selection.APPLE && reelThree == Selection.APPLE) || (reelOne == Selection.APPLE && reelFive == Selection.APPLE) || (reelTwo == Selection.APPLE && reelThree == Selection.APPLE) || (reelTwo == Selection.APPLE && reelFive == Selection.APPLE) || (reelThree == Selection.APPLE && reelFive == Selection.APPLE)) { cashWinnings = (cashWinnings + winThreeMatchingReels);
        System.out.println(displayLine + 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) || (reelFour == Selection.APPLE) || (reelFive == Selection.APPLE)) { cashWinnings = (cashWinnings + winOneMatchingReel);
        System.out.println(displayLine + reelOne + displayPipe + reelTwo + displayPipe + reelThree + displayPipe + reelFour + displayPipe + reelFive + displayPipe + "\n" + displayLineDoubleLine + "\n" + displayWinnerMessage + "\n" + displayLineSingleLine + "\n" + displayWinnerTotal + cashWinnings + "\n" + displayLineSingleLine + "\n" + displayRemainingCredit + playerCredits);
    }
}

```

## 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(displayLine + 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(displayLine + reelOne + displayPipe + reelTwo + displayPipe + reelThree + displayPipe + reelFour + 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) && (reelFive == Selection.APPLE) || (reelOne == Selection.APPLE) && (reelTwo == Selection.APPLE) && (reelFour == Selection.APPLE) && (reelFive == Selection.APPLE)) { cashWinnings = (cashWinnings + 100);
    System.out.println(displayLine + 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(displayLine + reelOne + displayPipe + reelTwo + displayPipe + reelThree + displayPipe + reelFour + 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) || (reelOne == Selection.APPLE) && (reelThree == Selection.APPLE) || (reelOne == Selection.APPLE) && (reelFive == Selection.APPLE) || (reelTwo == Selection.APPLE) && (reelThree == Selection.APPLE) || (reelTwo == Selection.APPLE) && (reelFive == Selection.APPLE) || (reelThree == Selection.APPLE) && (reelFive == Selection.APPLE)) { cashWinnings = (cashWinnings + winThreeMatchingReels);
    System.out.println(displayLine + 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) || (reelFour == Selection.APPLE) || (reelFive == Selection.APPLE)) { cashWinnings = (cashWinnings + winOneMatchingReel);
    System.out.println(displayLine + 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

```

=====
|   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!

```

## Tests

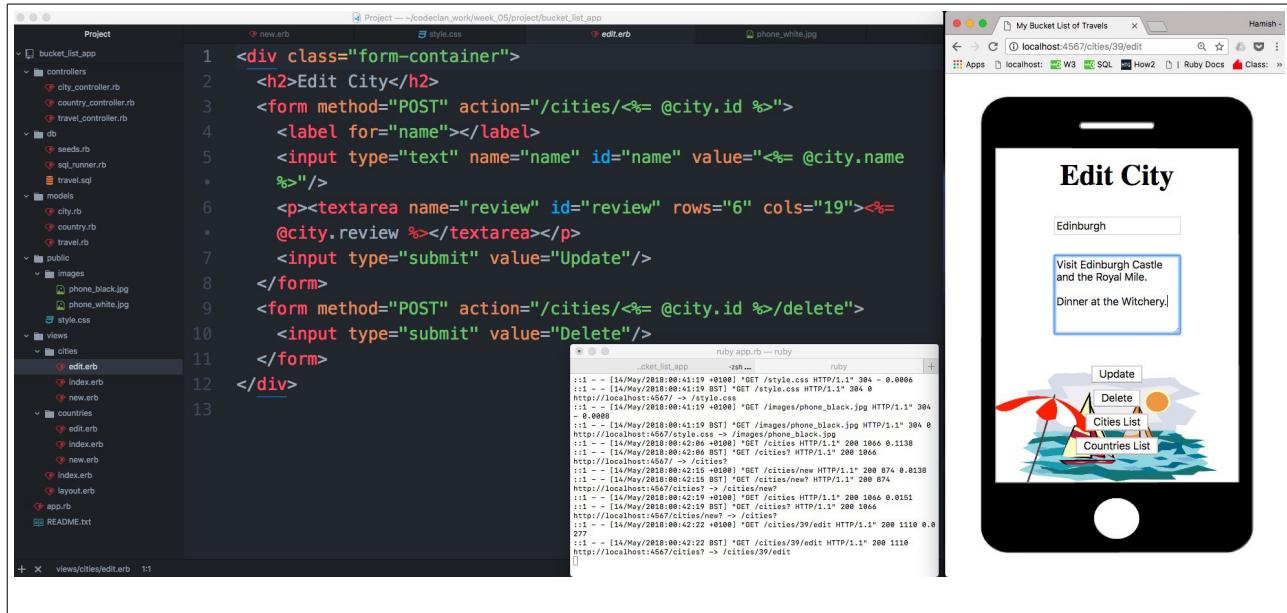


FruitMachineTest

- 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 developer's environment with several windows open:

- Project View:** Shows the project structure with files like bucket\_list\_app, controllers, db, models, public, and views/cities/editerb.
- Code Editor:** Displays the contents of editerb.rb, showing Ruby code for a form to edit a city.
- Terminal:** Shows a log of HTTP requests from a browser session, including file uploads and database interactions.
- Mobile Phone Simulation:** A black smartphone icon displaying a UI for "Edit City". It has fields for "Edinburgh", "Visit Edinburgh Castle and the Royal Mile.", "Dinner at the Witchery!", and a "Update" button. Below the phone are buttons for "Delete", "Cities List", and "Countries List".

## 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 displays a development environment with multiple windows:

- Code Editor:** Shows the `edit.erb` file containing the code for the "Edit City" form. The form includes fields for name, review, and a submit button labeled "Update".
- Database Browser:** Shows the PostgreSQL 9.0 database with the `cities` table. A new row has been added with id 39, name "Edinburgh", and review "Visit Edinburgh Castle and the Royal Mile.".
- Terminal:** Shows a log of requests and responses. Key entries include a POST request to add Edinburgh and a subsequent GET request to view the updated list.
- Browser:** Shows the "Edit City" page with the input field set to "Edinburgh" and the review text area containing "Visit Edinburgh Castle and the Royal Mile. Dinner at the Witchery.".
- Mobile Device Mockup:** Shows a smartphone displaying the "Edit City" screen with the same input values.

## P. 15 User output result

The screenshot is divided into four quadrants:

- Top Left:** Shows a smartphone displaying the "New City" screen with an input field containing "Iceland" and a "Add City" button.
- Top Right:** Shows a smartphone displaying the "City List" screen with a list of cities including "Edinburgh", "Iceland", "Madrid", "Berlin", "Port of Spain", and "Paris".
- Bottom Left:** Shows a smartphone displaying the "Edit City" screen with an input field containing "Iceland" and a "Update" button.
- Bottom Right:** Shows a screenshot of the PostgreSQL 9.0 database browser with the `cities` table. A new row has been added with id 47, name "Iceland", and review "Visit the Blue lagoon".

**User interaction:** The user inputs "Iceland" and clicks "Add City".

**System Response:** The system returns a list of cities, including the newly added "Iceland".

**User Action:** The user updates the city information on the mobile device.

**Database Update:** The database table shows the new entry with id 47, name "Iceland", and review "Visit the Blue lagoon".

## P. 16 Bug tracking report showing the errors diagnosed and corrected.

Evidence for unit

## P. 17 Testing your program

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

Example of test code

with the test NOT passing

The terminal shows the command `rake` being run, which outputs:

```
hamish@Hamish-MBP:~/weekend homework$ rake
(in /Users/hamish/Harmonis/MBP/weekend homework/PDA_Static_and_Dynamic_Task_A)
rake aborted!
SyntaxError: syntax error, unexpected end-of-input, expecting keyword_end (SyntaxError)
  from /Users/hamish/Harmonis/MBP/weekend homework/PDA_Static_and_Dynamic_Task_A/spec/testing_task_2_spec.rb:1:in `block in <top-level block>'
```

The code editor panes show the following files:

- card.rb:** A Ruby class definition for a Card with methods initialize, checkforAce, highest\_card, and cards\_total.
- testing\_task\_2.rb:** A Ruby class definition for a CardGame with methods checkforAce, highest\_card, and self.cards\_total.
- testing\_task\_2\_spec.rb:** A MiniTest::Test subclass with setup and test\_check\_for\_ace methods. It includes require\_relative('card.rb') and assert\_equal("Clubs", @card.value).

Example of test code

with the test then passing

The terminal shows the command `rake` being run, which outputs:

```
hamish@Hamish-MBP:~/weekend homework$ rake
(in /Users/hamish/Harmonis/MBP/weekend homework/PDA_Static_and_Dynamic_Task_A)
Run options: --seed 59756
# Running:

Finishes in 0.000000s, 1650.1651 runs/s, 1650.1651 assertions/s.
1 runs, 1 assertions, 0 failures, 0 errors, 0 skips
PDA_Static_and_Dynamic_Task_A
```

The code editor panes show the same files as the previous screenshot, but the testing code has been corrected:

- testing\_task\_2\_spec.rb:** The `assert_equal` statement now uses `1` instead of `"Clubs"`.