

**Evidence Gathering Document for SQA Level 8 Professional Developer Award.**

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

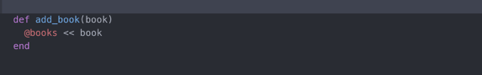
Please fill in each point with screenshot or diagram and description of what you are showing.

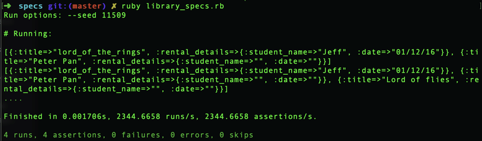
Each point requires details that cover each element of the Assessment Criteria, along with a brief description of the kind of things you should be showing.

**Week 2**

|  |  |  |  |
| --- | --- | --- | --- |
| **Unit** | **Ref** | **Evidence** |  |
| **I&T** | I.T.5 | Demonstrate the use of an array in a program. Take screenshots of:  \*An array in a program  \*A function that uses the array  \*The result of the function running | |
|  |  | **Description: In the second screenshot we can see how with a shovel operator used in the function add\_book() that takes a book as an argument we can add the book to the already existing @books collection.** | |

****

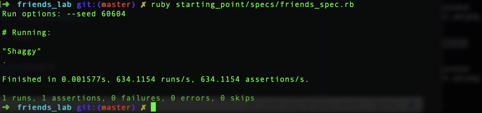
****

****

|  |  |  |  |
| --- | --- | --- | --- |
| **Unit** | **Ref** | **Evidence** |  |
| **I&T** | I.T.6 | Demonstrate the use of a hash in a program. Take screenshots of:  \*A hash in a program  \*A function that uses the hash  \*The result of the function running | |
|  |  | **Description: The Function get\_tv\_show() takes a person as an argument and prints in the output the favourite tv show for a give person.** | |

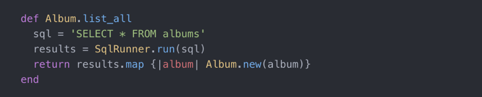
****

****

****

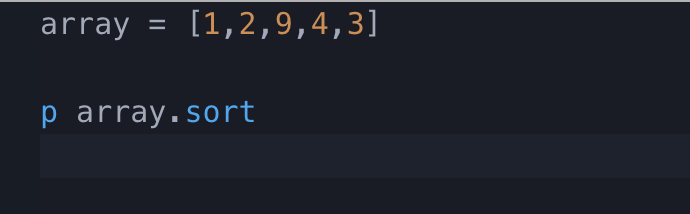
**Week 3**

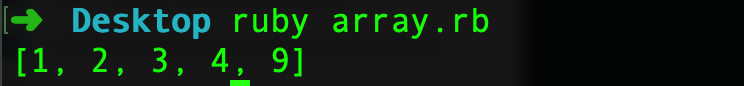
|  |  |  |  |
| --- | --- | --- | --- |
| **Unit** | **Ref** | **Evidence** |  |
| **I&T** | I.T.3 | Demonstrate searching data in a program. Take screenshots of:  \*Function that searches data  \*The result of the function running | |
|  |  | **Description:** The method list\_all called on Album does an sql query on a database and return all the  Albums presents on the table | |

****

****

|  |  |  |  |
| --- | --- | --- | --- |
| **Unit** | **Ref** | **Evidence** |  |
| **I&T** | I.T.4 | Demonstrate sorting data in a program. Take screenshots of:  \*Function that sorts data  \*The result of the function running | |
|  |  | **Description:** I have called the method sort on the array and the number not previously in order are in order on the output | |

****

****

**Week 5 and 6**

|  |  |  |  |
| --- | --- | --- | --- |
| **Unit** | **Ref** | **Evidence** |  |
| **A&D** | A.D.1 | A Use Case Diagram | |
|  |  | **Description:**  This case diagram shows how a user(Trader) should be able to perform a set of action on the platform built | |

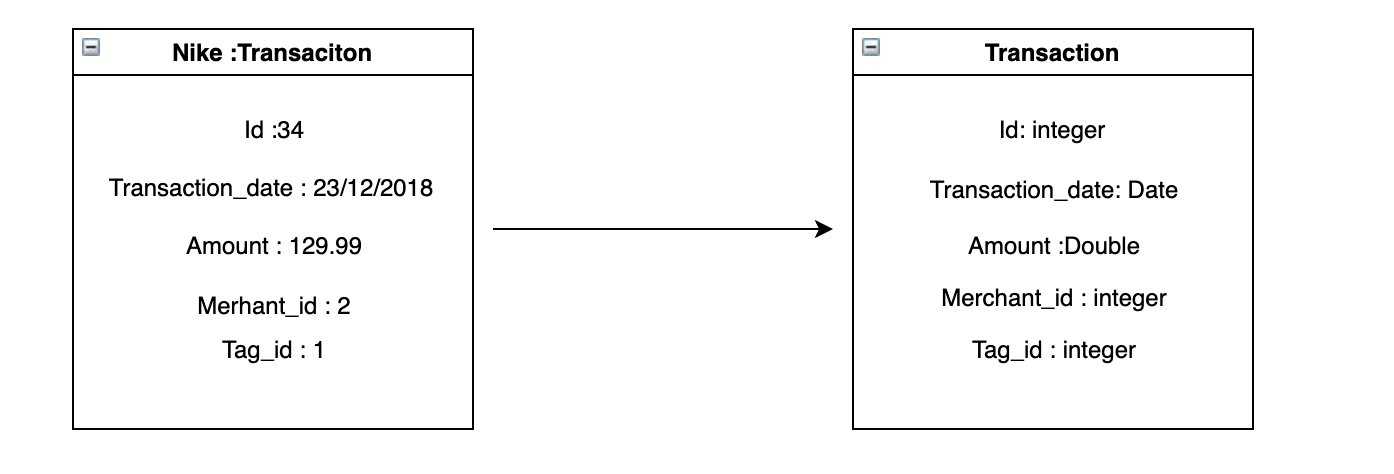
****

|  |  |  |  |
| --- | --- | --- | --- |
| **Unit** | **Ref** | **Evidence** |  |
| **A&D** | A.D.2 | A Class Diagram | |
|  |  | **Description:** An example of class diagram from the Ruby solo project, the two external classes Merchants and Tags have a relationship one to many with the Class Transaction in the middle. | |

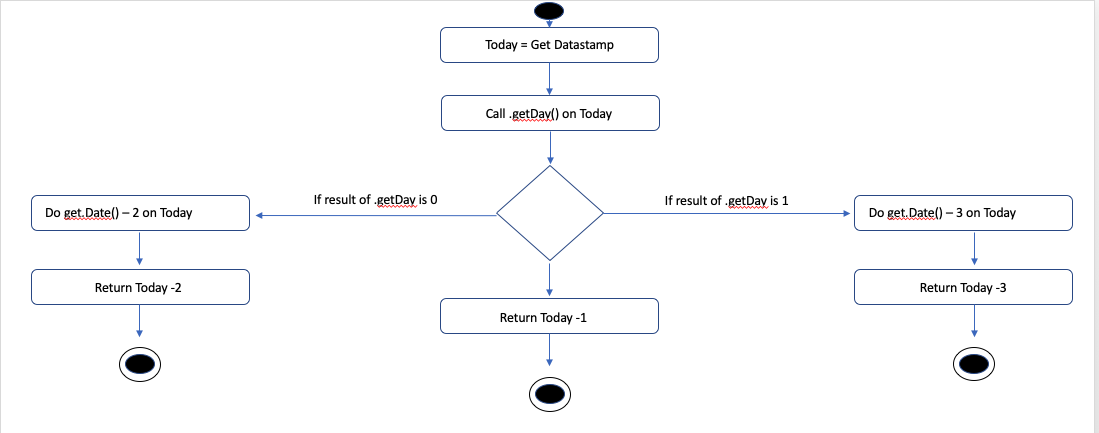
****

|  |  |  |  |
| --- | --- | --- | --- |
| **Unit** | **Ref** | **Evidence** |  |
| **A&D** | A.D.3 | An Object Diagram | |
|  |  | **Description: The screenshot shows a transaction object with the attributes.** | |

**Paste Screenshot here**

****

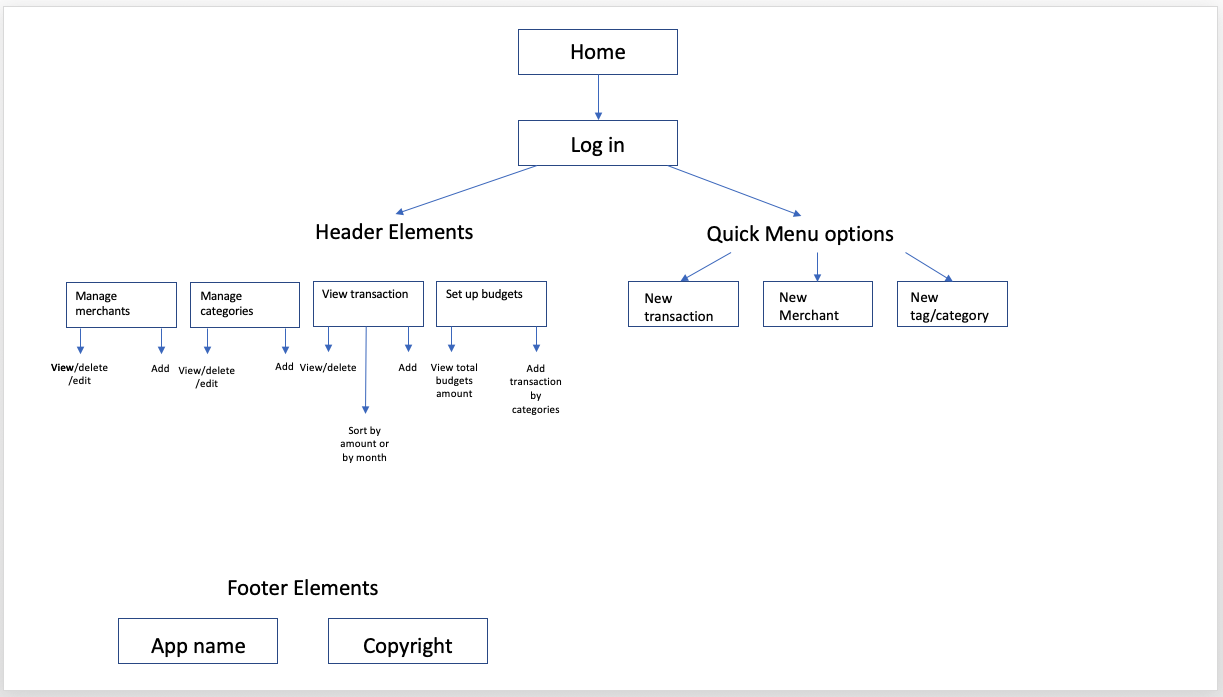
|  |  |  |  |
| --- | --- | --- | --- |
| **Unit** | **Ref** | **Evidence** |  |
| **A&D** | A.D.4 | An Activity Diagram | |
|  |  | **Description:** This case diagram shows how from a starting point of creating a Date data using Datastamp the application performs different action depending on the return value of the method getDay( ) | |

****

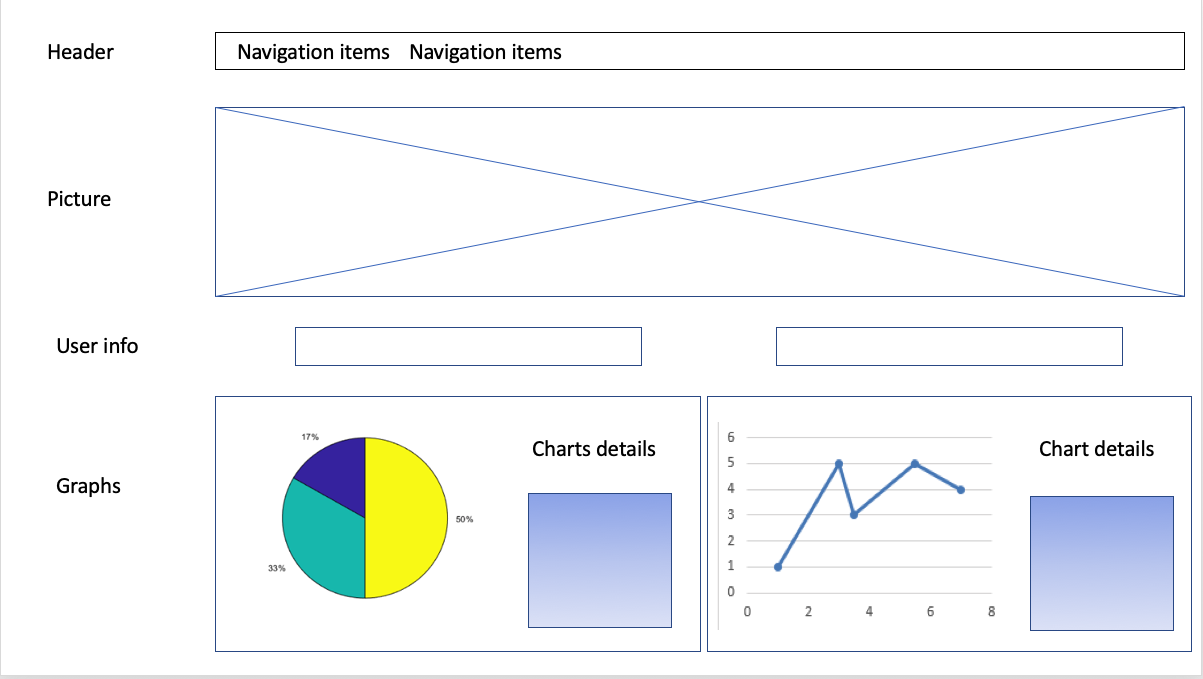
|  |  |  |  |
| --- | --- | --- | --- |
| **Unit** | **Ref** | **Evidence** |  |
| **A&D** | A.D.6 | Produce an Implementations Constraints plan detailing the following factors:  \*Hardware and software platforms  \*Performance requirements  \*Persistent storage and transactions  \*Usability  \*Budgets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

|  |  |  |
| --- | --- | --- |
| **Constraint Category** | **Implementation Constraint** | **Solution** |
| **Hardware and Software Platforms** | Running the app on local-host does only allow you the max power of the computer is running on. | Try to host the application on a free Amazon Web Service with a more powerful server structure |
| **Performance Requirements** | In some cases, the response time of the API we were using was very slow and this showed down the all application. | We decided to store locally the previous date closing price for the shares our platform uses.  This allowed us to call the API only once a day to get the prices for our shares. |
| **Persistent Storage and transactions** | The data is now only saved locally in the machine of the person running the application.  This is not shared with all the people working on the project. | Host our Database on a bigger and more powerful structure accessible for all the people working on the project, perhaps using Dynamo DB provided by Amazon |
| **Usability** | Difficulty to navigate the app without appropriate knowledge in the financial industry | Use of simple terminology to explain more complex financial terms |
| **Budgets** | No budget for project.  Limitation of 5 calls per minute over the API chosen | Present the application to possible sponsor to create a capital usable towards HW and SW solution |
| **Time** | Limited time of a week for the full project.  More time needed to work on the extensions of our project. | Establish a realistic MVP for the product and only once basic functionality have been tested and fully functional start to work on possible extensions. |

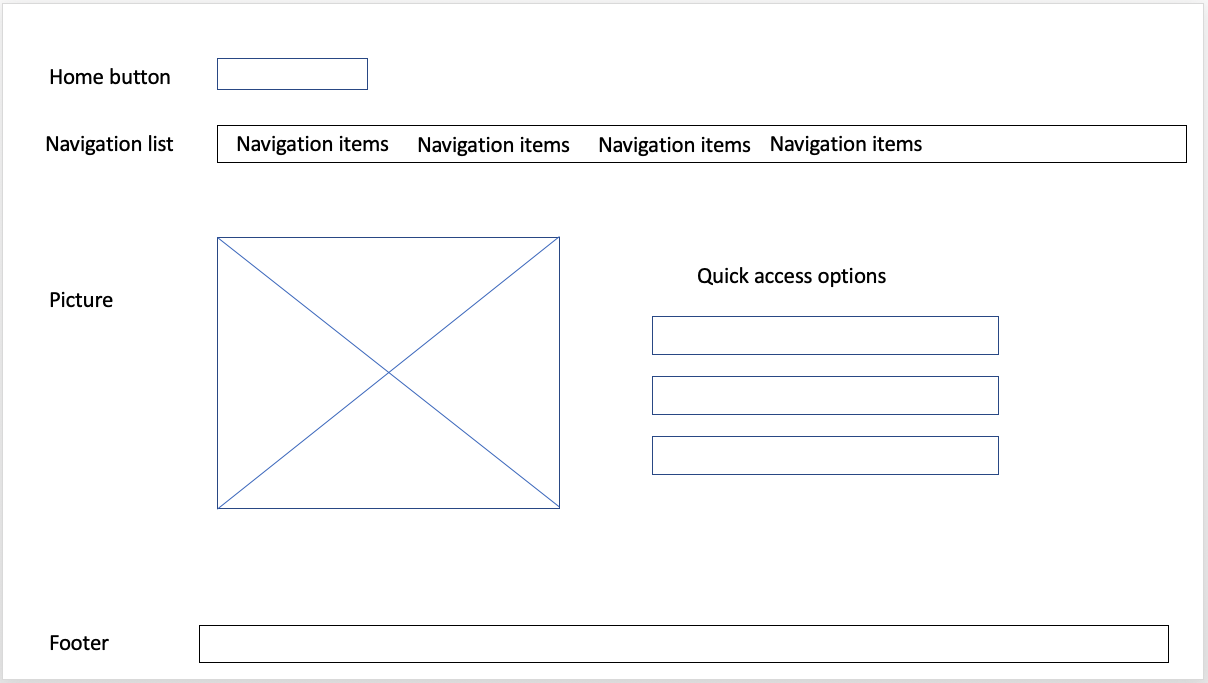
|  |  |  |  |
| --- | --- | --- | --- |
| **Unit** | **Ref** | **Evidence** |  |
| **P** | P.5 | User Site Map | |
|  |  | **Description:** The below picture is the structure of my Transaction Tracker web application built for the Ruby project. | |

****

|  |  |  |  |
| --- | --- | --- | --- |
| **Unit** | **Ref** | **Evidence** |  |
| **P** | P.6 | 2 Wireframe Diagrams | |
|  |  | **Description:** | |

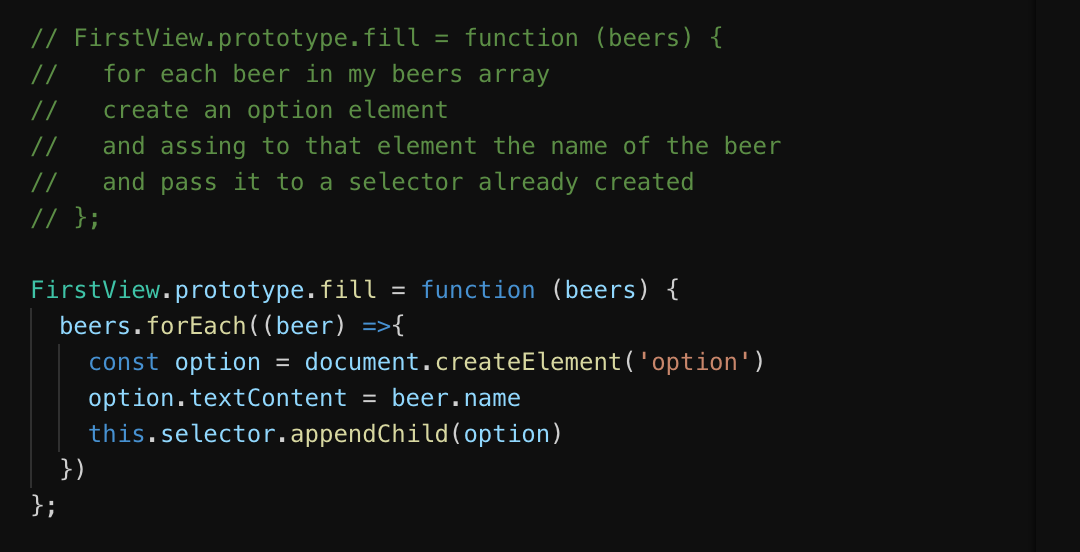
****

The above picture is a Wireframe diagram for the landing page of the JavaScript project

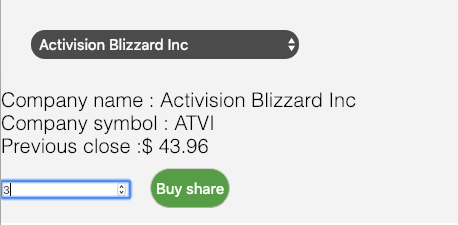
****

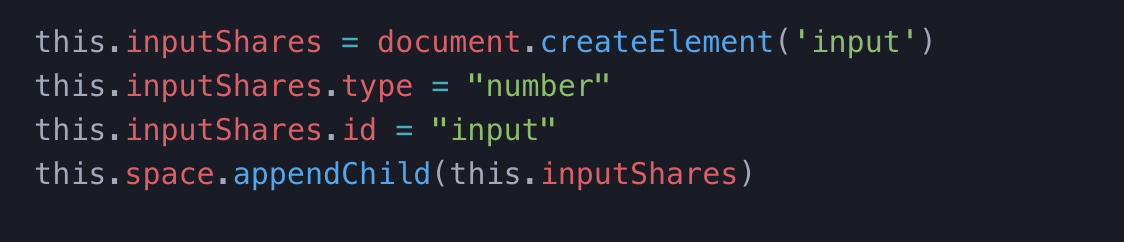
The above is the main page of the solo Ruby project

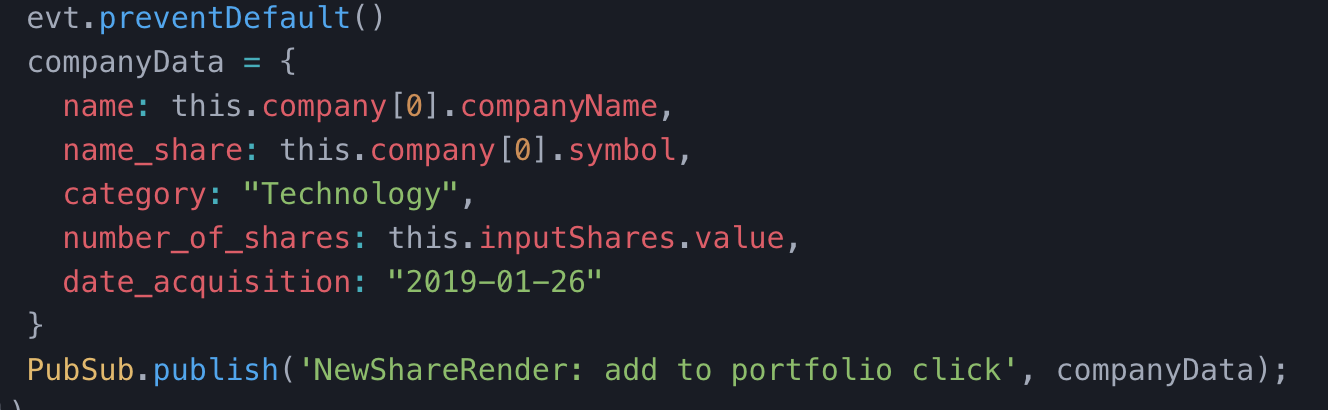
|  |  |  |  |
| --- | --- | --- | --- |
| **Unit** | **Ref** | **Evidence** |  |
| **P** | P.10 | Example of Pseudocode used for a method | |
|  |  | **Description:** | |

****

|  |  |  |  |
| --- | --- | --- | --- |
| **Unit** | **Ref** | **Evidence** |  |
| **P** | P.13 | Show user input being processed according to design requirements. Take a screenshot of:  \* The user inputting something into your program  \* The user input being saved or used in some way | |
|  |  | **Description: In the first screenshot we can see how the user can select the preferred share from a list and in the input box choose the number of shares to buy and we use the input to buy the shares and add this to the portfolio.** | |

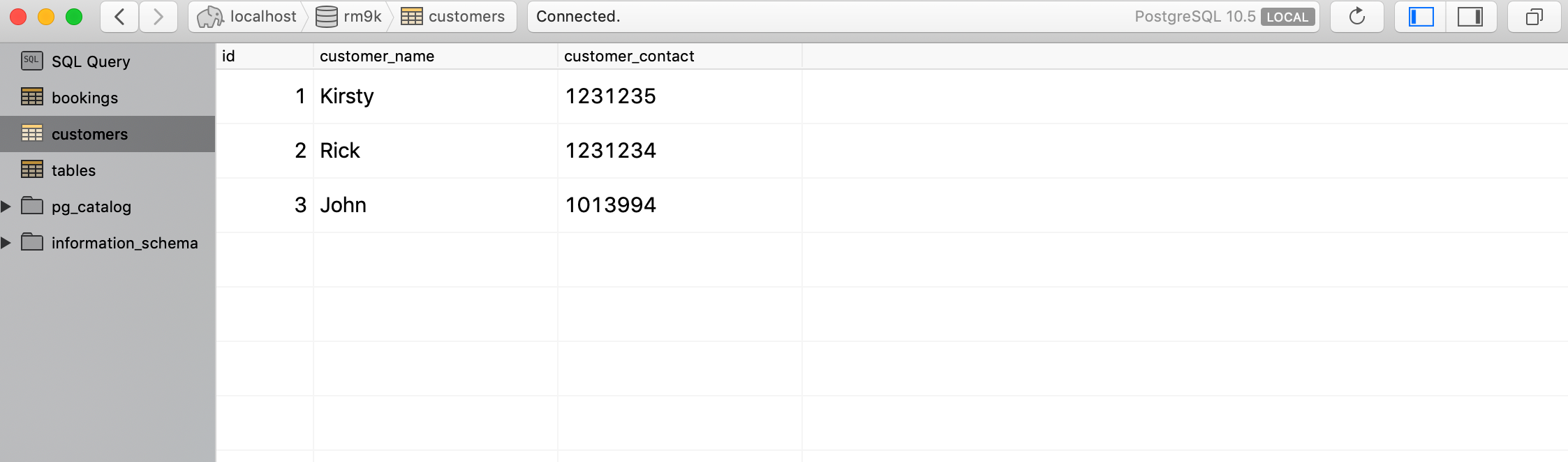
****

****

****

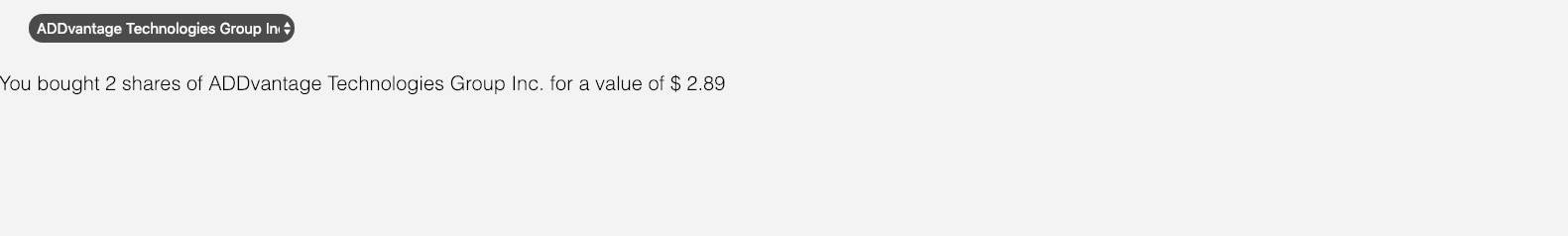
|  |  |  |  |
| --- | --- | --- | --- |
| **Unit** | **Ref** | **Evidence** |  |
| **P** | P.14 | Show an interaction with data persistence. Take a screenshot of:  \* Data being inputted into your program  \* Confirmation of the data being saved | |
|  |  | **Description: An instance of a Class Customer has been saved on the first screenshot and in the second we can see the data being saved on the DB** | |

****

****

|  |  |  |  |
| --- | --- | --- | --- |
| **Unit** | **Ref** | **Evidence** |  |
| **P** | P.15 | Show the correct output of results and feedback to user. Take a screenshot of:  \* The user requesting information or an action to be performed  \* The user request being processed correctly and demonstrated in the program | |
|  |  | **Description: In the first screenshot the user is inputting the number of shares he what to buy and the second screenshot is a confirmation of the shares being bought.** | |

****

****

|  |  |  |  |
| --- | --- | --- | --- |
| **Unit** | **Ref** | **Evidence** |  |
| **P** | P.11 | Take a screenshot of one of your projects where you have worked alone and attach the Github link. | |
|  |  | **Description: A Ruby web application for users to motor their expense and set up budget for particular categories. Built using RESTful API in Ruby, Sinatra and SQL** | |

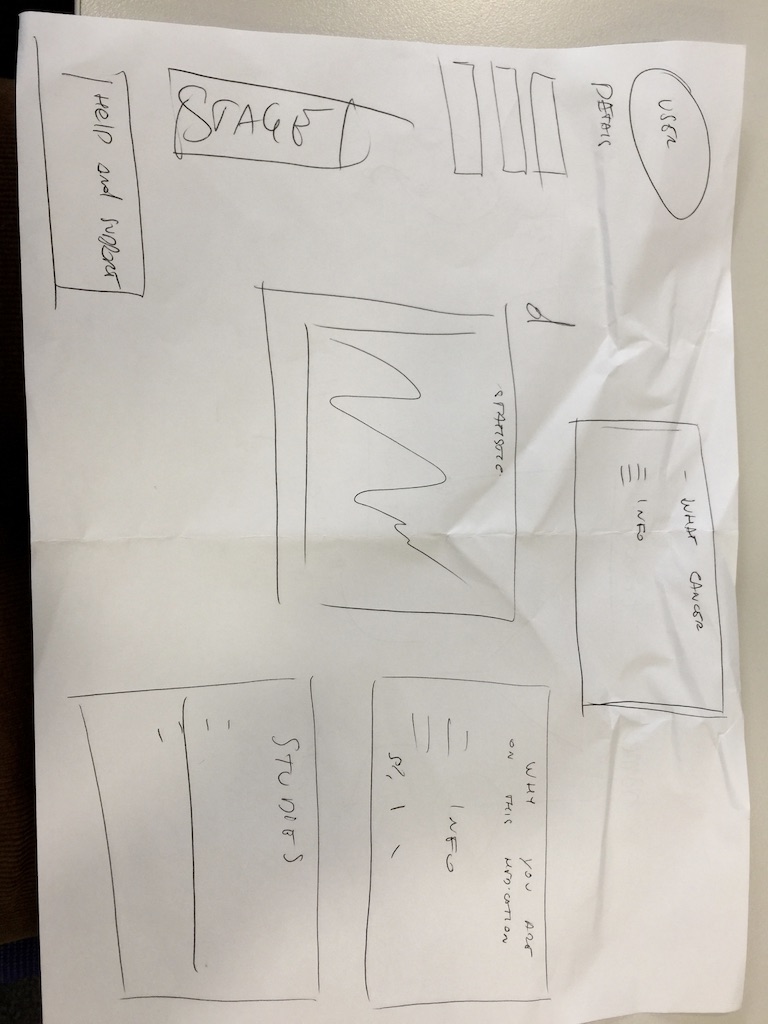
**MONEY-TRACKER**

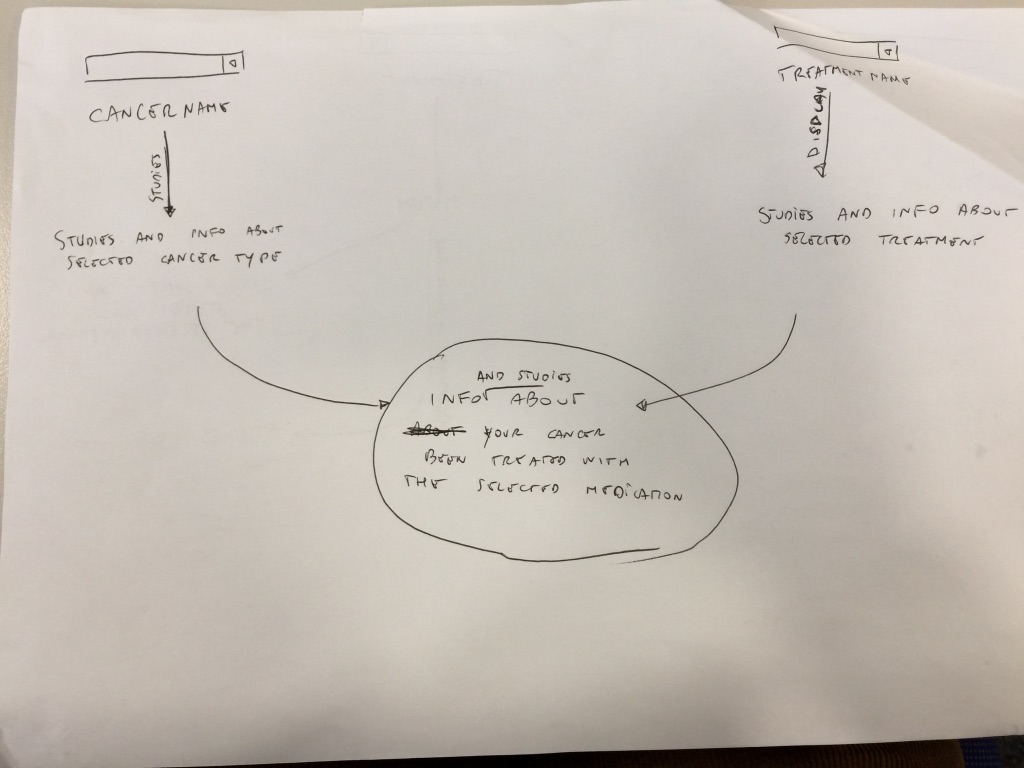
**https://github.com/Davide1988/money\_tracker\_Project**

****

|  |  |  |  |
| --- | --- | --- | --- |
| **Unit** | **Ref** | **Evidence** |  |
| **P** | P.12 | Take screenshots or photos of your planning and the different stages of development to show changes. | |
|  |  | **Description: This was the planning for a project I have developed for the Product Forge, the first is a structure of the landing page and the second is graphical representation of two select being and the result of the selection.** | |

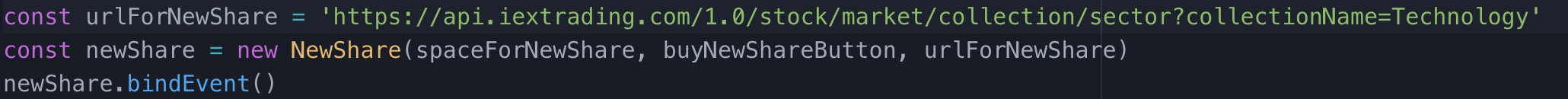
**Paste Screenshot here**

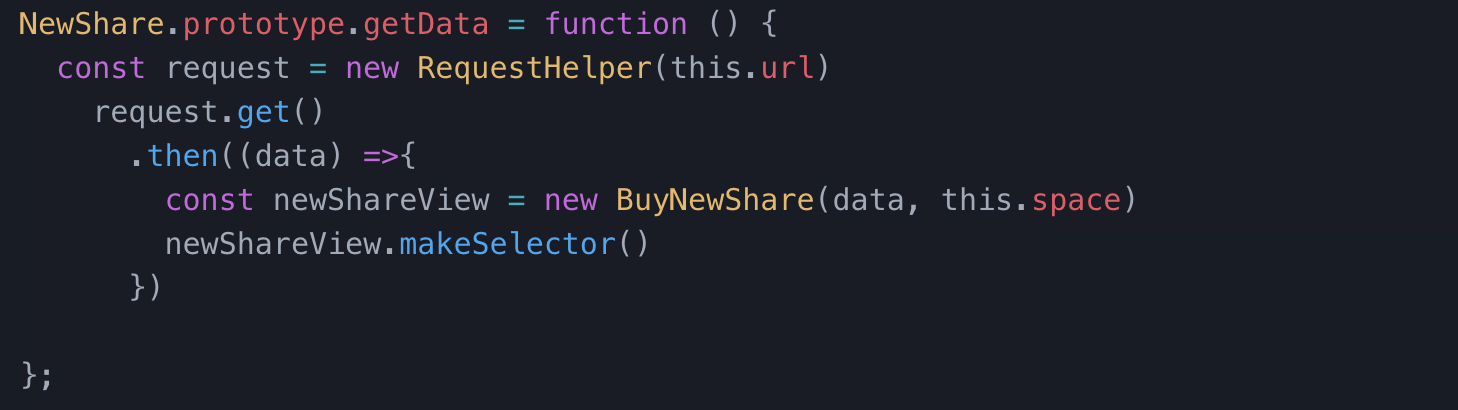
****

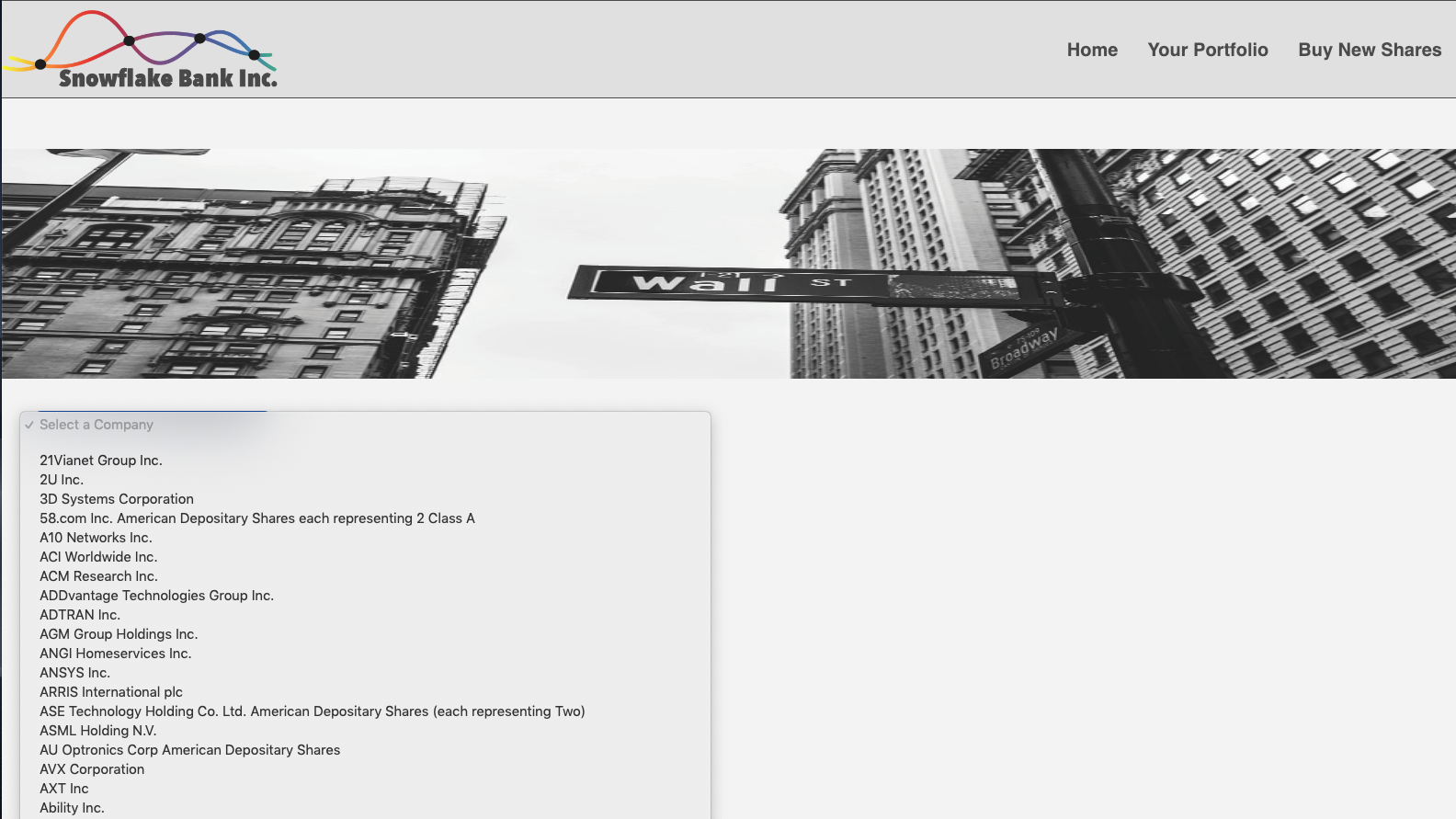
****

**Week 7**

|  |  |  |  |
| --- | --- | --- | --- |
| **Unit** | **Ref** | **Evidence** |  |
| **P** | P.16 | Show an API being used within your program. Take a screenshot of:  \* The code that uses or implements the API  \* The API being used by the program whilst running | |
|  |  | **Description: The URL gets passed as argument into a constructor and then used in the function getData() to when we do a .get() in order to retrieve the data JSON data used to populate our selector in the last screenshot.** | |

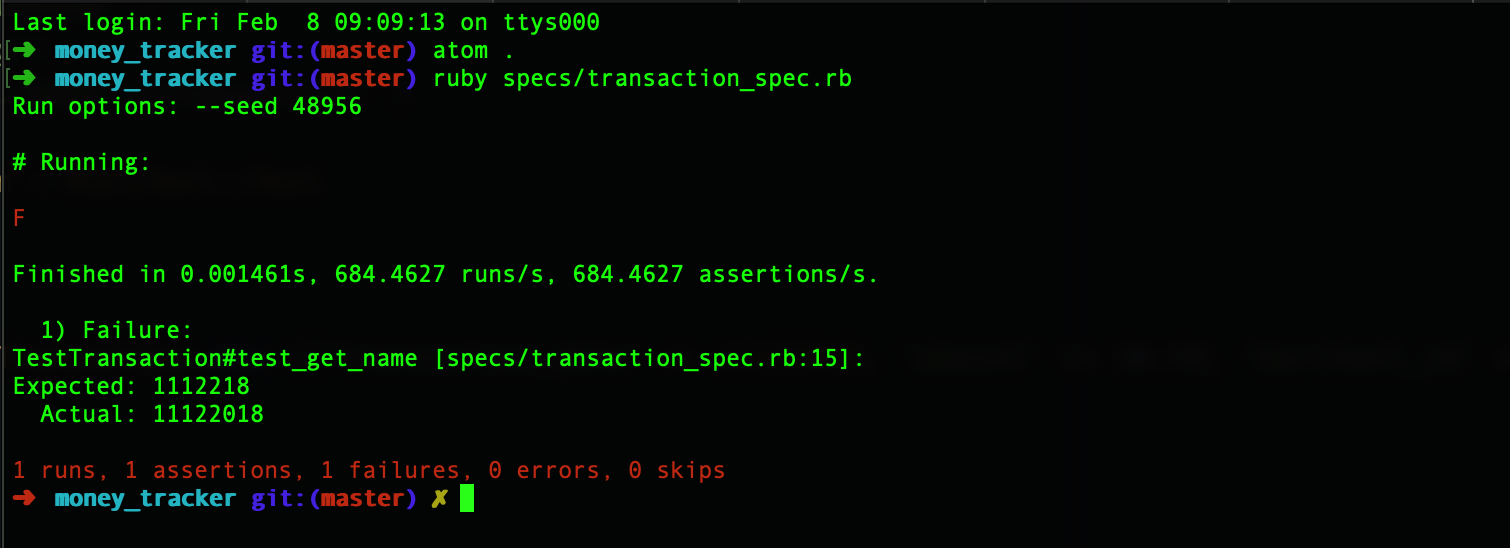
****

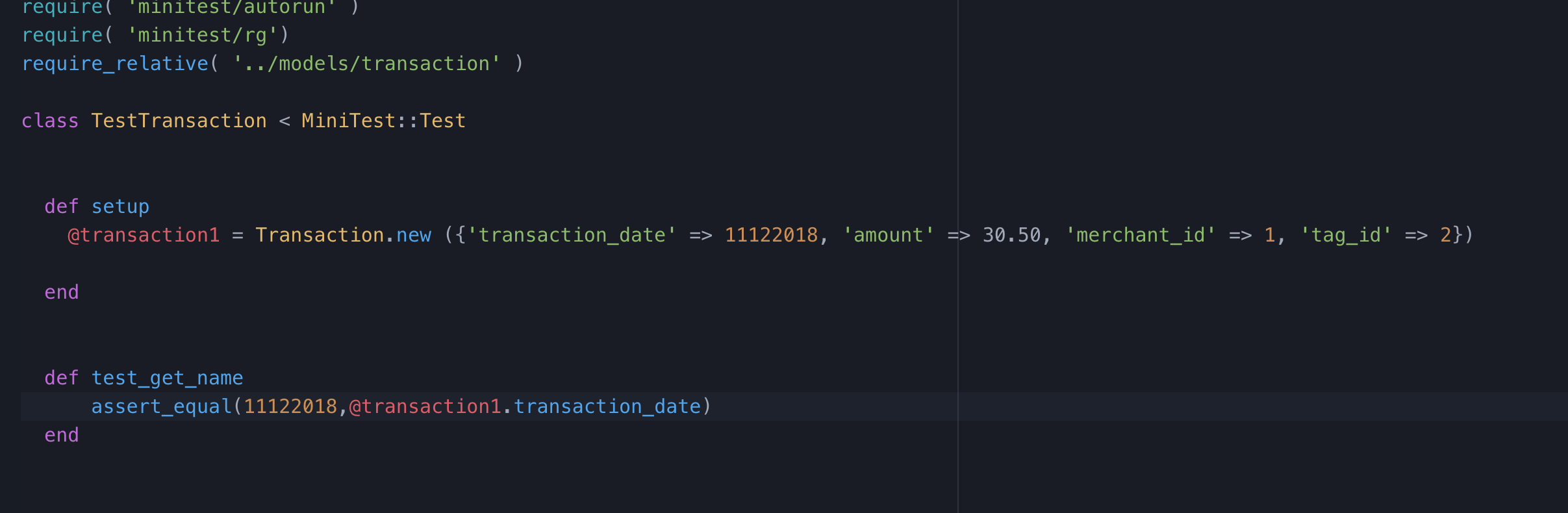
****

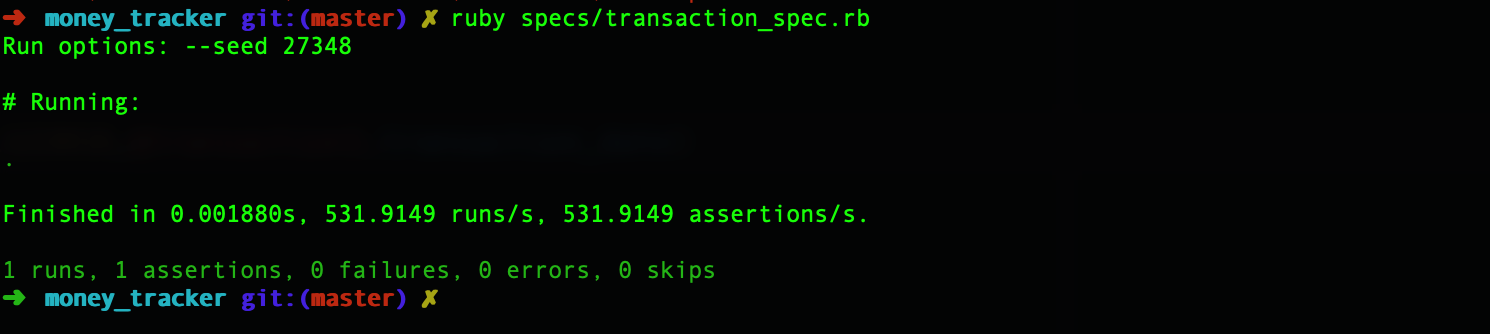
****

|  |  |  |  |
| --- | --- | --- | --- |
| **Unit** | **Ref** | **Evidence** |  |
| **P** | P.18 | Demonstrate testing in your program. Take screenshots of:  \* Example of test code  \* The test code failing to pass  \* Example of the test code once errors have been corrected  \* The test code passing | |
|  |  | **Description: In the first screen-shot you can see how an instance of the class transaction has been. The function test\_get\_name() is asserting that the first argument is the same as calling the method .transaction\_date() on the instance @transacion1. The second screen shot demonstrate that the result of running this test is failing as we are asserting the wrong date, this error is fixed in the 3rd screenshot and we can see that after that the test comes back positive.** | |

****

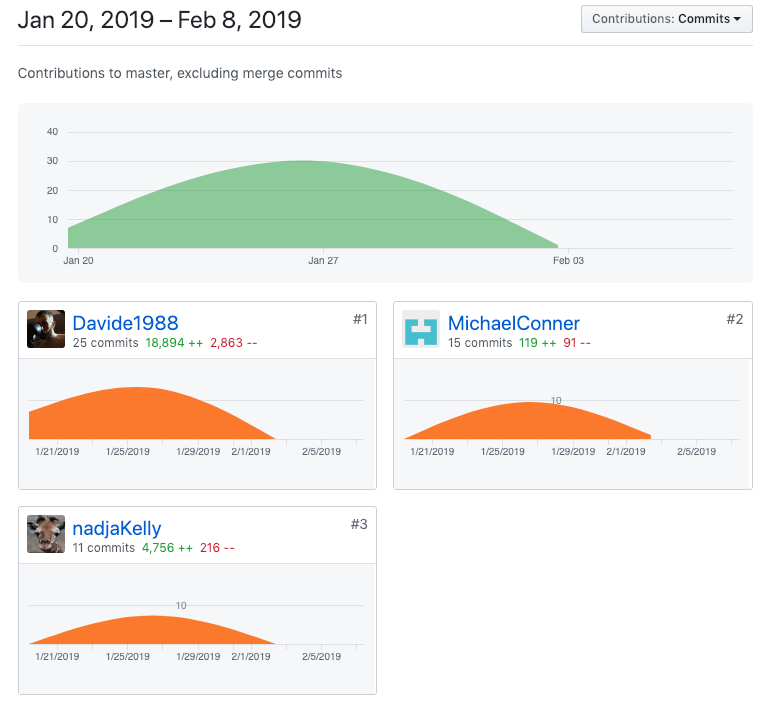
****

****

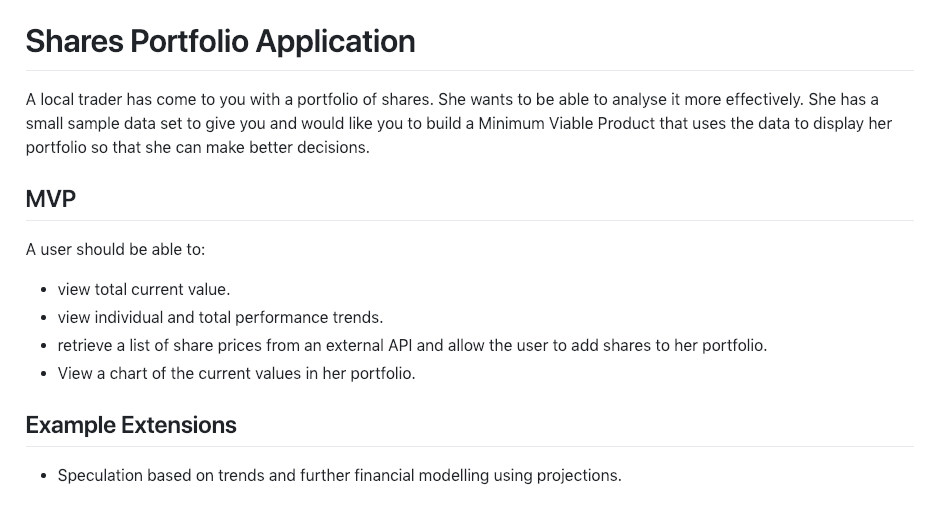
****

**Week 9**

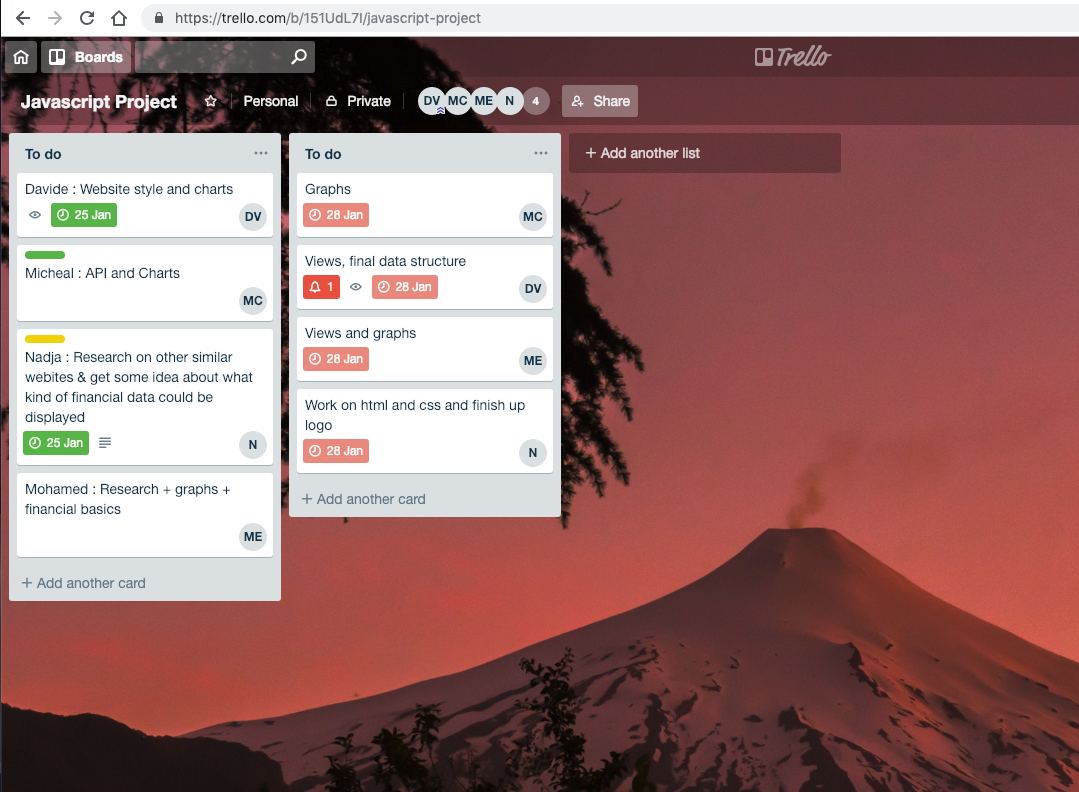
|  |  |  |  |
| --- | --- | --- | --- |
| **Unit** | **Ref** | **Evidence** |  |
| **P** | P.1 | Take a screenshot of the contributor’s page on Github from your group project to show the team you worked with. | |
|  |  | **Description: The screenshot of the collaboration for the team Project.** | |

****

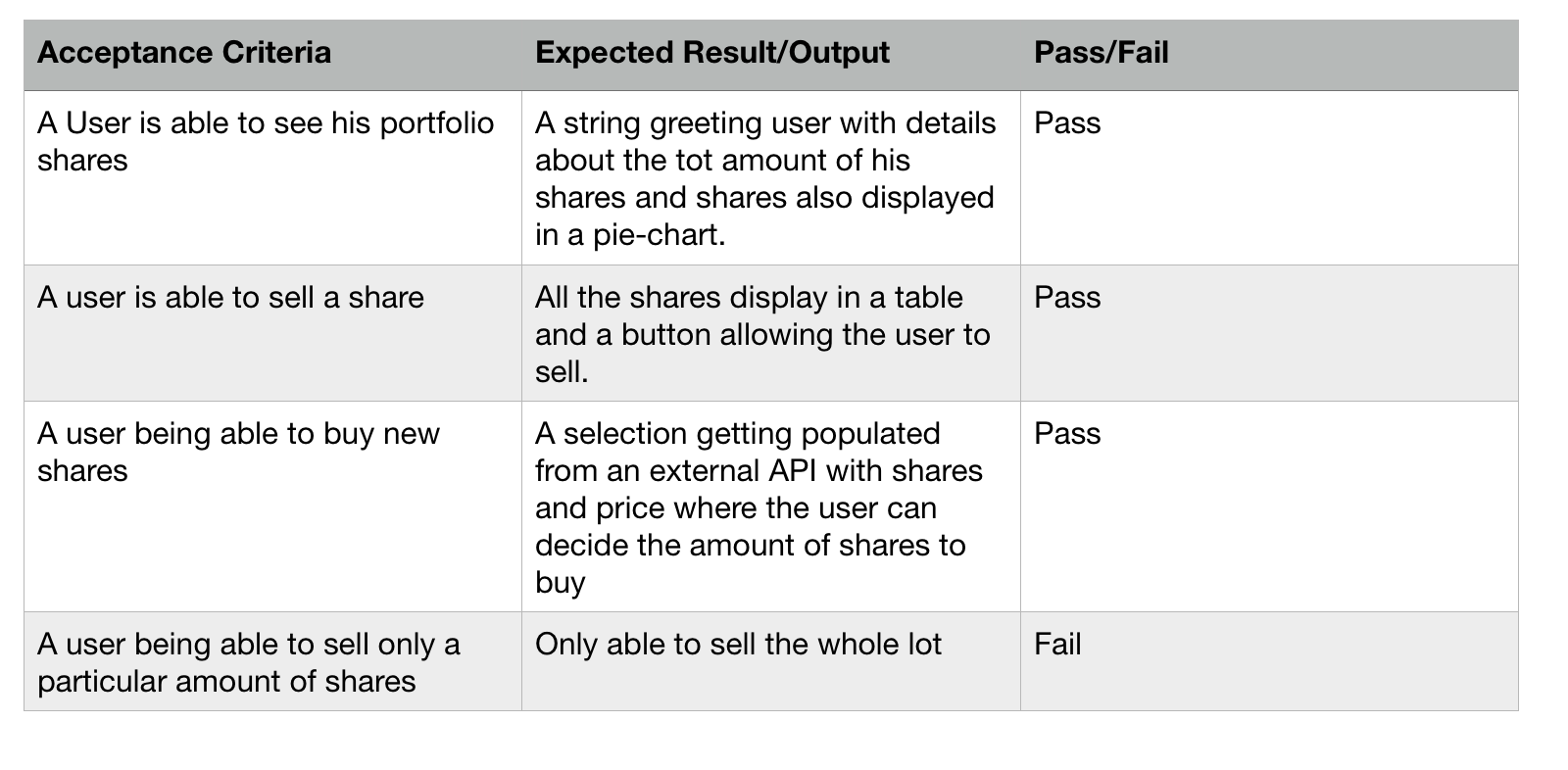
|  |  |  |  |
| --- | --- | --- | --- |
| **Unit** | **Ref** | **Evidence** |  |
| **P** | P.2 | Take a screenshot of the project brief from your group project. | |
|  |  |  | |

****

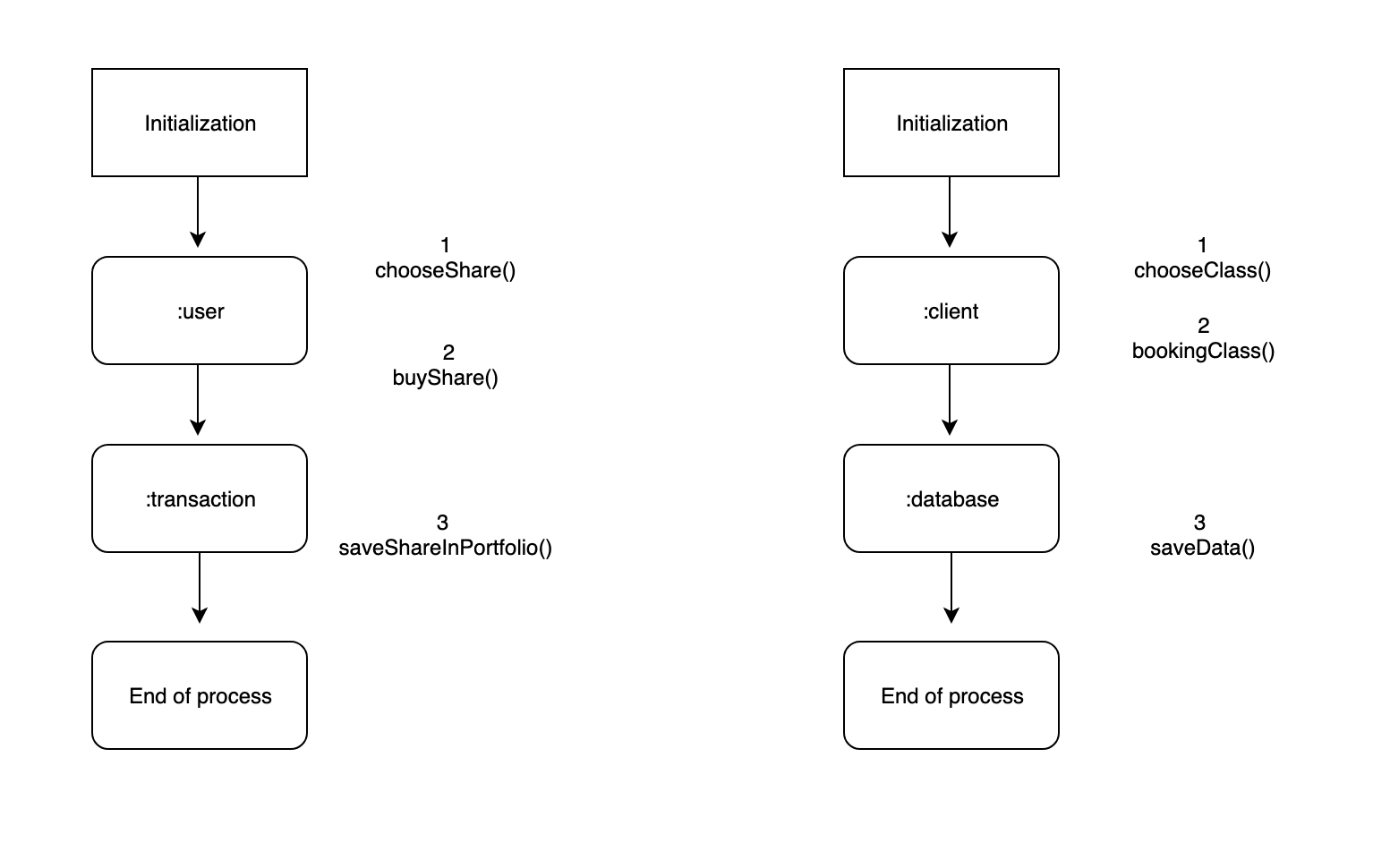
|  |  |  |  |
| --- | --- | --- | --- |
| **Unit** | **Ref** | **Evidence** |  |
| **P** | P.3 | Provide a screenshot of the planning you completed during your group project, e.g. Trello MOSCOW board. | |
|  |  | **Description: A few actions being reported on Trello of task to do for the project.** | |

****

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

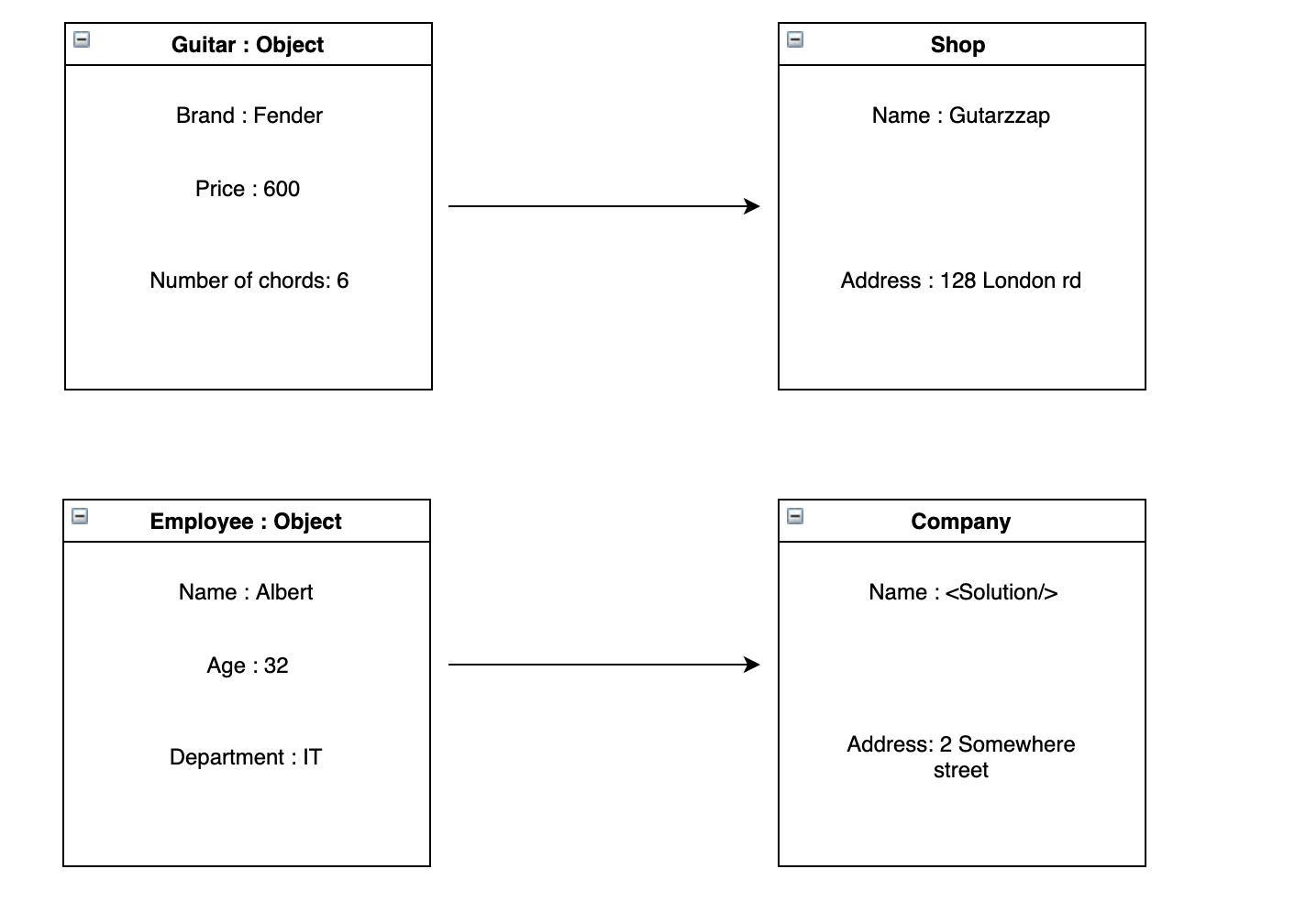
****

|  |  |  |  |
| --- | --- | --- | --- |
| **Unit** | **Ref** | **Evidence** |  |
| **P** | P.7 | Produce two system interaction diagrams (sequence and/or collaboration diagrams). | |
|  |  | **Description: The first screenshot on the left side shows the sequence of a user choosing a share, then by clicking the buy button trig an event that we add this to his portfolio.**  **The second diagram shows how a client choose a class to attend, book this under their credential and ultimately saving the data into the DB.** | |

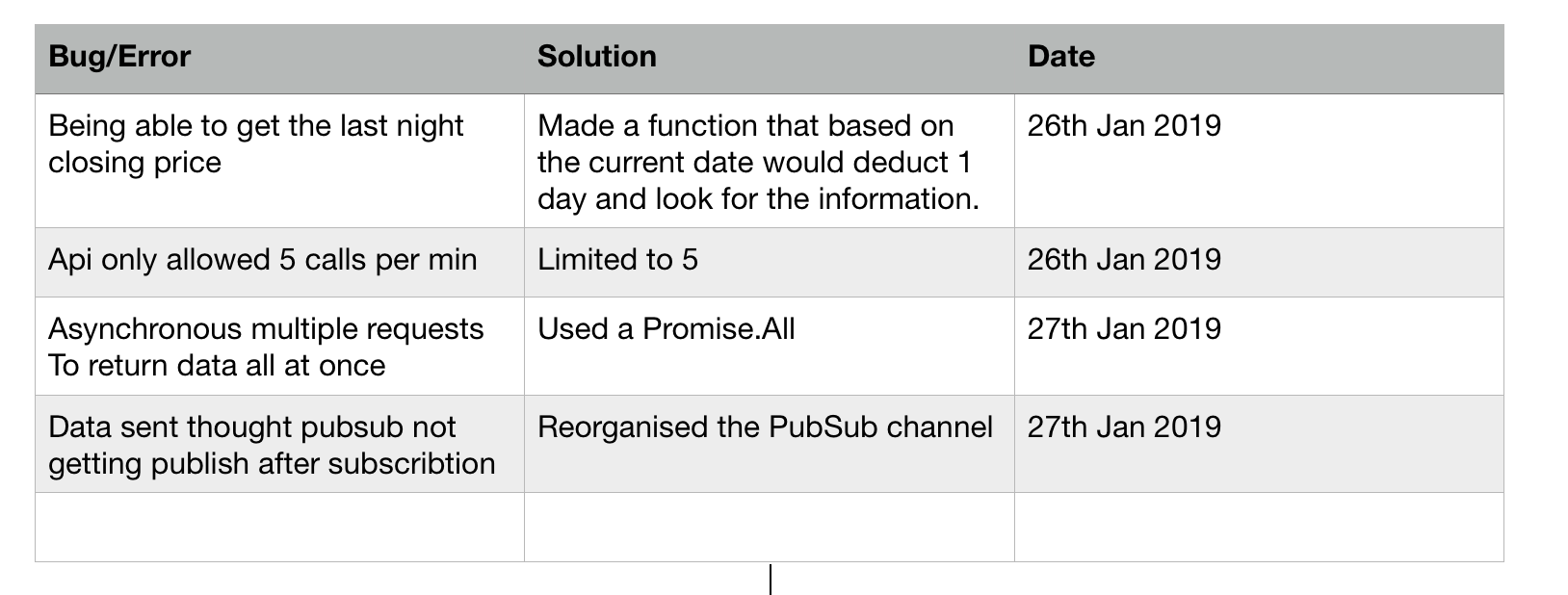
****

**Description here**

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

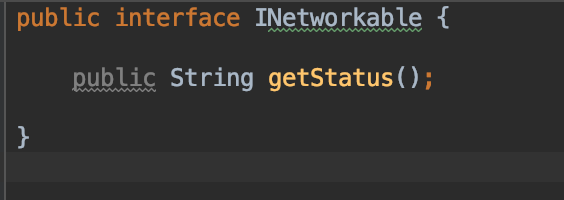
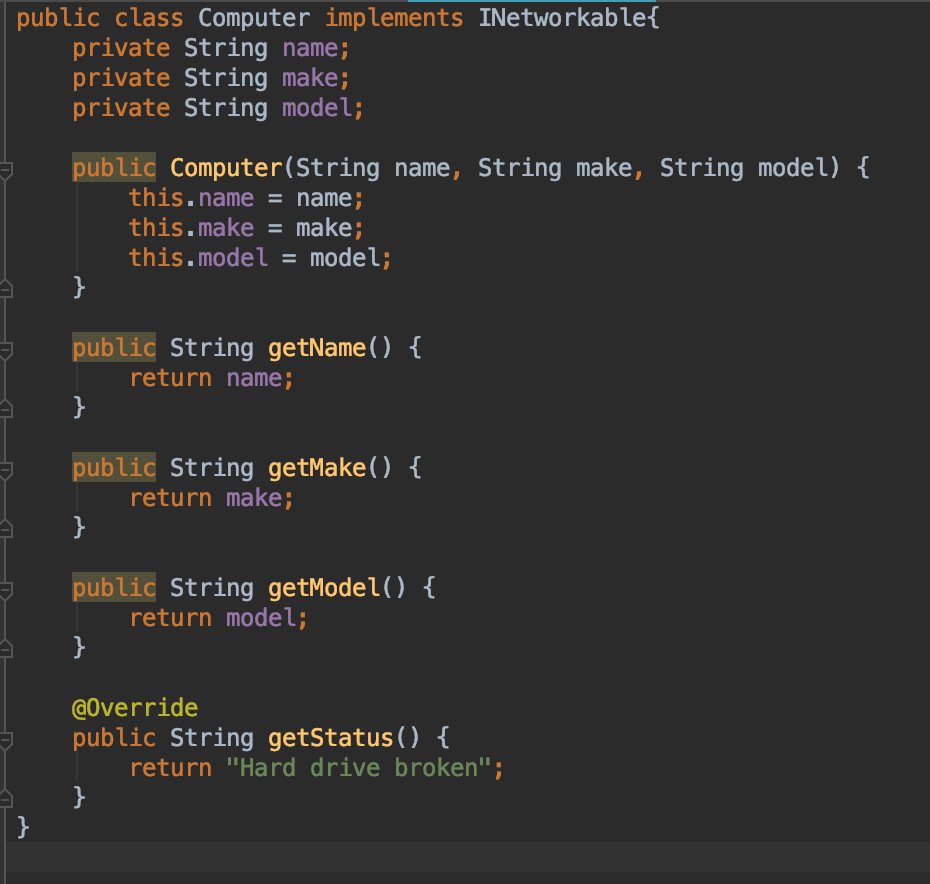
****

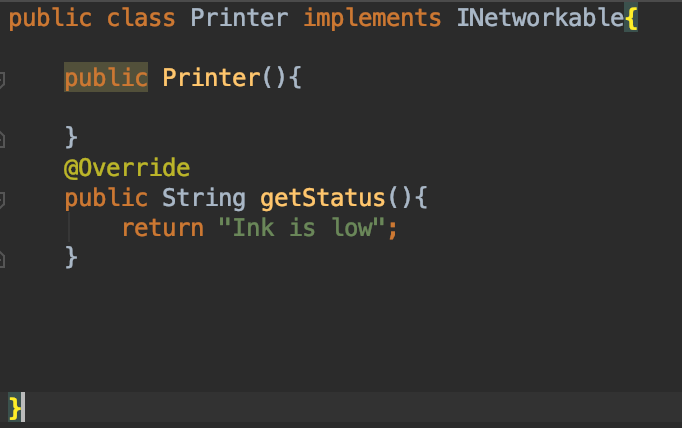
|  |  |  |  |
| --- | --- | --- | --- |
| **Unit** | **Ref** | **Evidence** |  |
| **P** | P.17 | Produce a bug tracking report | |
|  |  | **Description:** | |

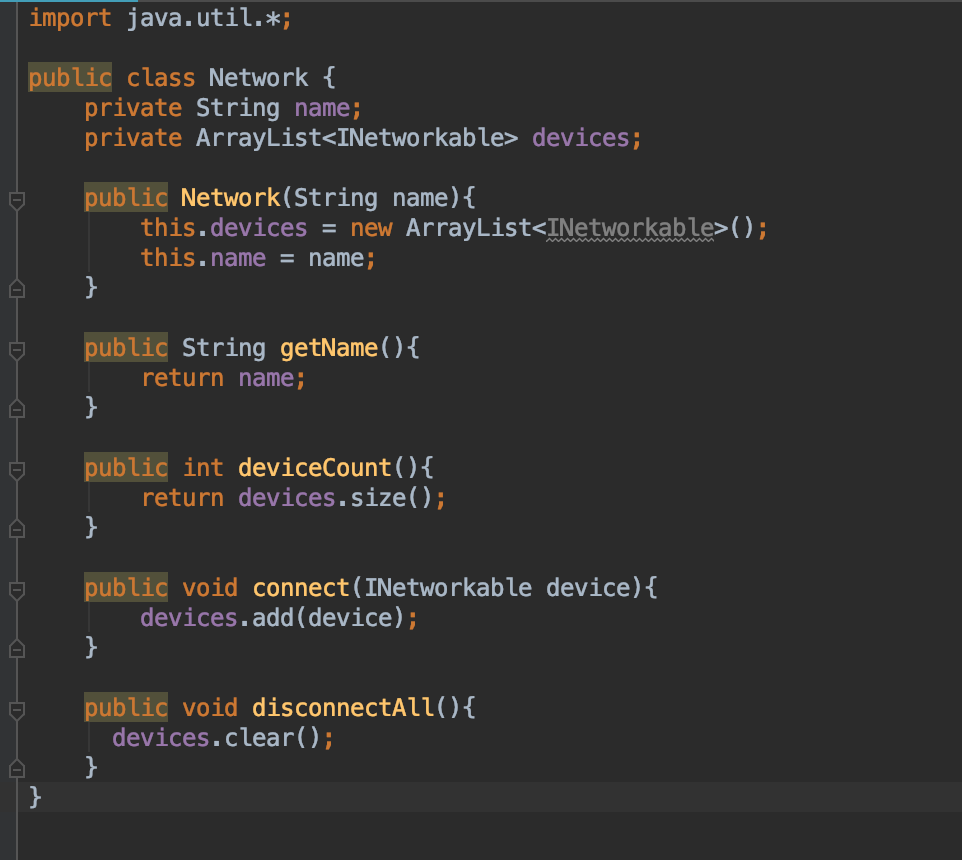
****

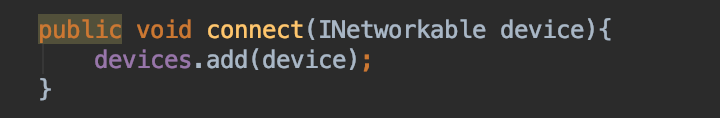
**Week 12**

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

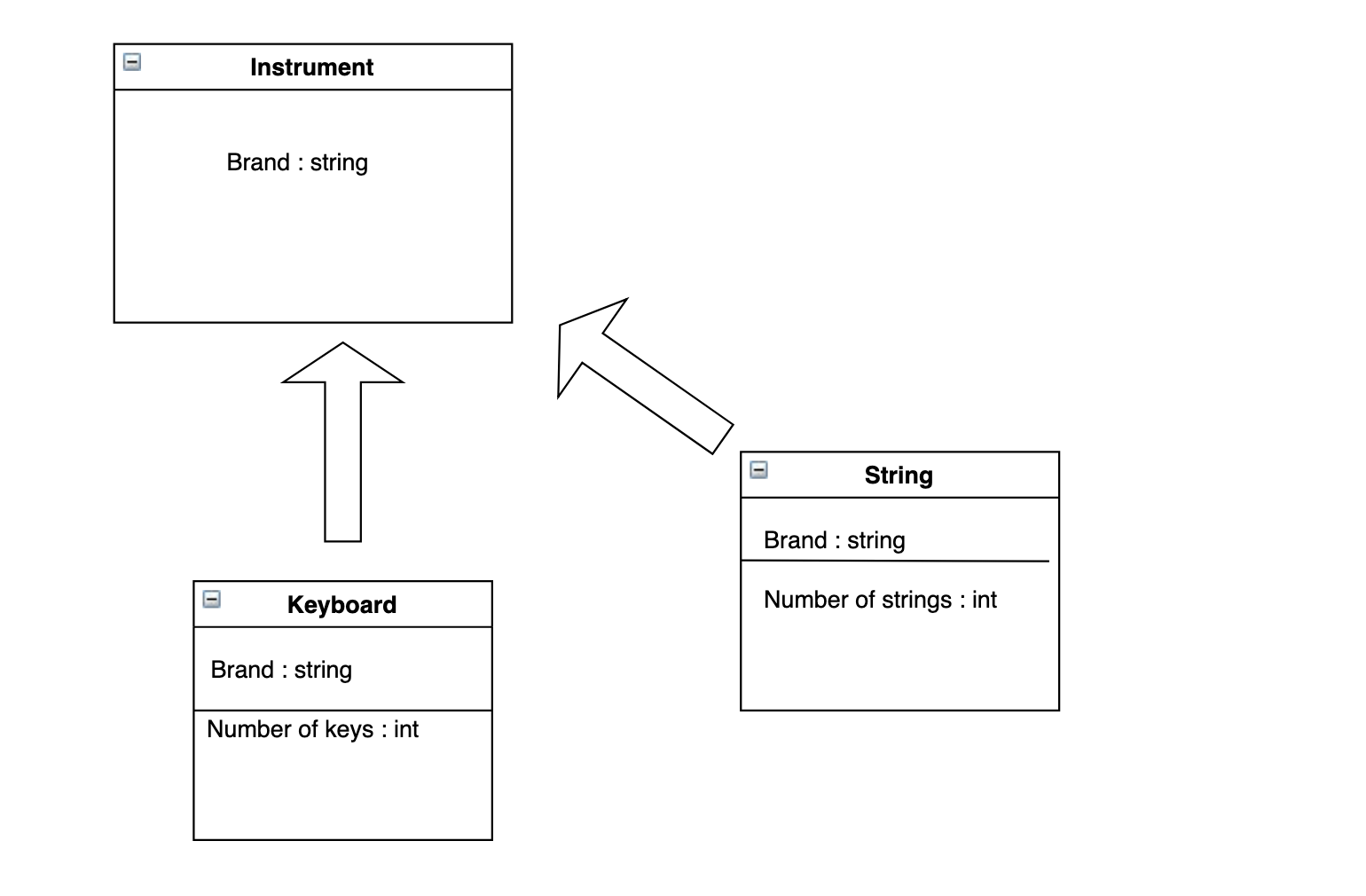
****

****

****

By implementing the INetworkable interface to the class Computer and Printer we have the ability to group this classes together and override a method on for each of them with a different behaviour.

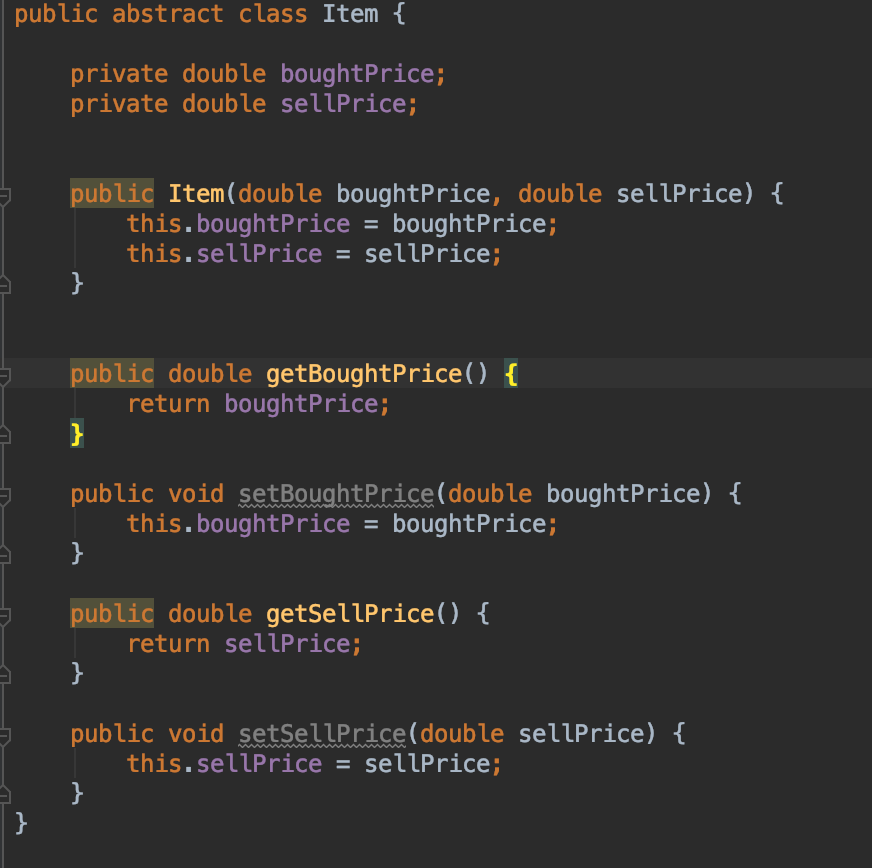
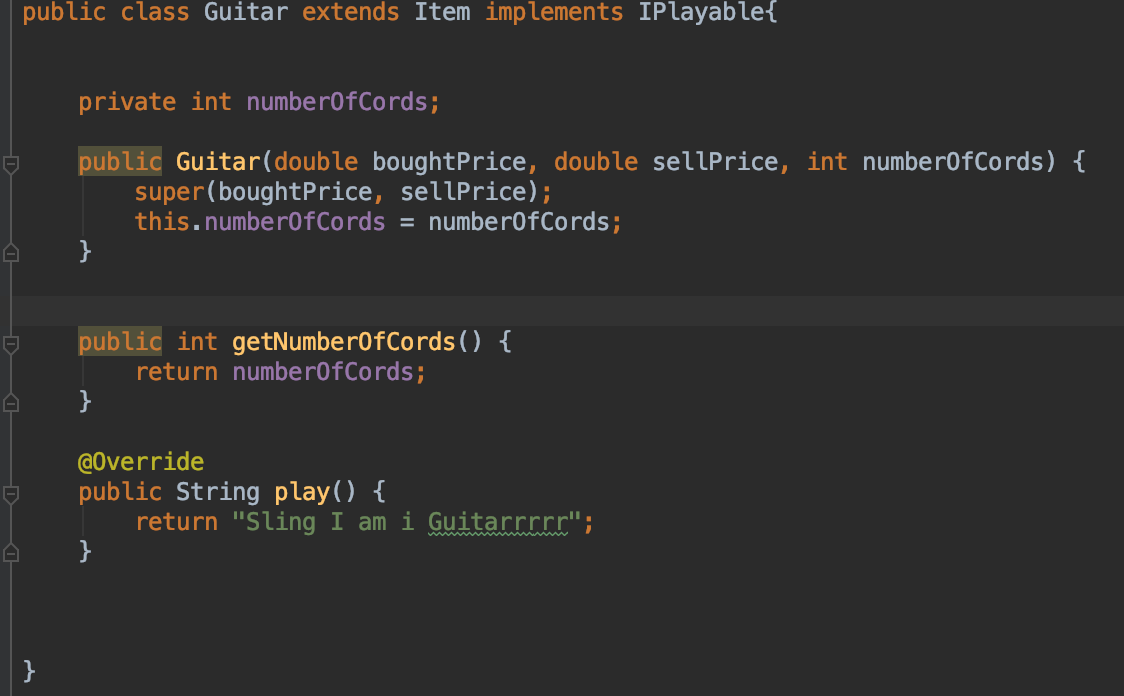
|  |  |  |  |
| --- | --- | --- | --- |
| **Unit** | **Ref** | **Evidence** |  |
| **A&D** | A.D.5 | An Inheritance Diagram | |
|  |  | **Description: The class String and Keyboard inherit from instrument the attribute brand.** | |

****

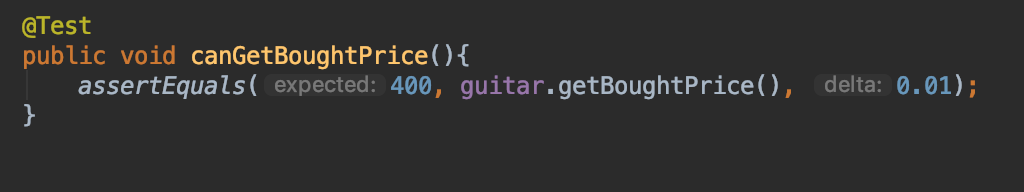
|  |  |  |  |
| --- | --- | --- | --- |
| **Unit** | **Ref** | **Evidence** |  |
| **I&T** | I.T.1 | The use of Encapsulation in a program and what it is doing. | |
|  |  | **Description: The screenshot shows that in a class named Customer, the**  **properties cannot be accessed; encapsulation aspects mean that the class is self-contained and changes are not allowed from outside of the class.** | |

****

|  |  |  |  |
| --- | --- | --- | --- |
| **Unit** | **Ref** | **Evidence** |  |
| **I&T** | I.T.2 | Take a screenshot of the use of Inheritance in a program. Take screenshots of:  \*A Class  \*A Class that inherits from the previous class  \*An Object in the inherited class  \*A Method that uses the information inherited from another class. | |
|  |  | **Description:** In this examples the Class Guitar inherits from Item and on object of Guitar can call the method canGetBoughtPrice() declared in the Item class | |

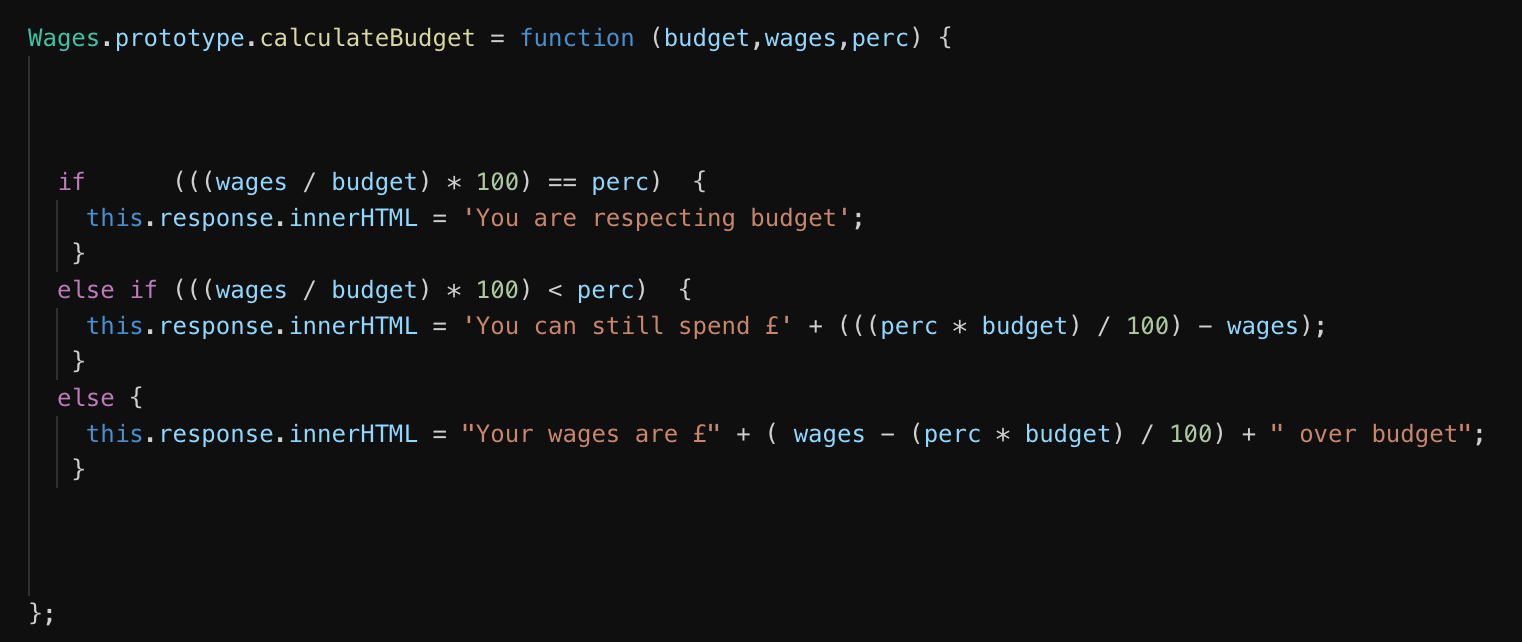
** **

****

****

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

With 3 arguments given: budget, wages and percentage, this algorithm calculates, if the Actual budget invested into wages is on line with budget, over or under.



With 2 arguments given: costPrice and Gross profit to respect this algorithm helps to calculate the menu price for a final item

