

Evidence Gathering Document for SQA Level 8 Professional Developer Award.

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

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

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

Week 2

Unit	Ref	Evidence	
I&T	I.T.5	Demonstrate the use of an array in a program. Take screenshots of: *An array in a program *A function that uses the array *The result of the function running	
		Description:	

```
@fishes = [  
  @fish1 = Fish.new("dot"),  
  @fish2 = Fish.new("stripes"),  
  @fish3 = Fish.new("goldie"),  
  @fish4 = Fish.new("snapper")  
]  
@river = River.new("Forth", @fishes)
```

array of fishes

```
def count_fish
  return @fishes.count
end
```

this is a ruby method using .count which counts number of items in an array

```
def test_number_of_fish
  expected = 4
  actual = @river.count_fish
  assert_equal(expected, actual)
end
```

test to prove method works by calling the count_fish method shown above and proving that it will count the correct number of fish in the array above which is 4.

```
Run options: --seed 44301
```

```
# Running:
```

```
.
```

```
Finished in 0.001044s, 957.8548 runs/s, 957.8548 assertions/s.
```

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

```
.....
```

terminal showing test of method passing

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:	

```
@new_pet = {
  name: "Calum",
  pet_type: :cat,
  breed: "Skinless",
  price: 100
}
```

hash of new pet

```
def pet_name(new_pet)
  return new_pet[:name]
end
```

function returning new pets name by accessing the new_pet key value :name

```
def test_pet_name
  name = pet_name(@new_pet)
  assert_equal("Calum", name)
end
```

test to find out if function works by returning the correct name by comparing it to the expected result and using the method pet_name.

```
→ specs git:(master) x ruby pet_shop_spec.rb
Run options: --seed 53354

# Running:

.

Finished in 0.001022s, 978.4737 runs/s, 978.4737 assertions/s.

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

terminal showing test of function passing

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:	

```
numbers = [1,2,3,4,5]

def add_together_numbers(array)
  total = 0
  for number in array do
    total += number
  end
  return total
end

p add_together_numbers(numbers)
```

array of numbers , function that adds together all the numbers by using a for loop which in turn goes through the array adding each number in turn to the total and then returns the total, p is used to print the total

```
➔ exercise_samples git:(master) ✖ ruby sample.rb
15
➔ exercise_samples git:(master) ✖
```

terminal printing the correct total of numbers

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:	

```
numbers = [4,10,3,8,5]

def sort_numbers(array)
  array.sort
end

p sort_numbers(numbers)
```

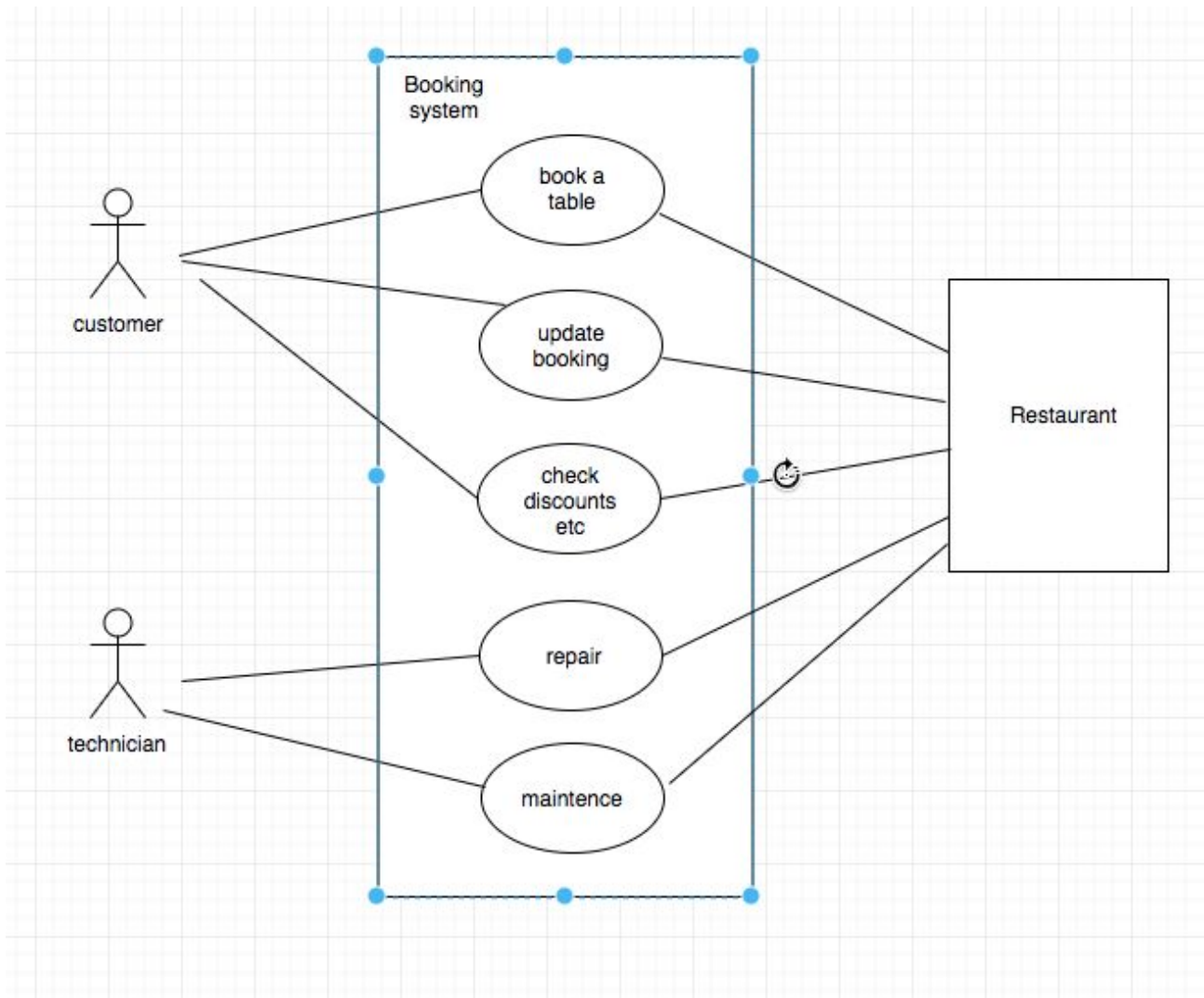
array of numbers, function sorting numbers from lowest to highest by using the Ruby method sort, and then p which prints result of this into terminal

```
→ exercise_samples git:(master) × ruby sample.rb
[3, 4, 5, 8, 10]
→ exercise_samples git:(master) × █
```

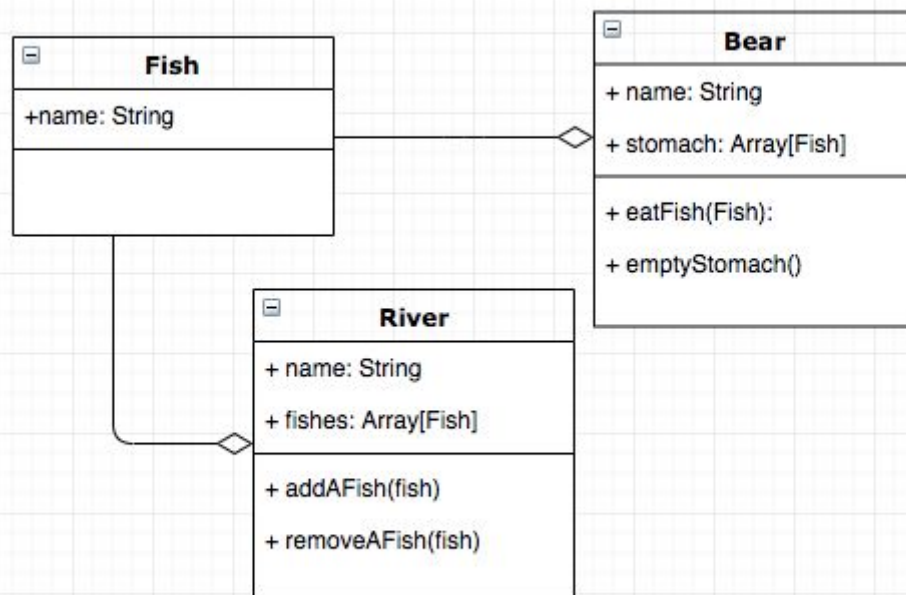
terminal showing result of function working and numbers in low-high order

Week 5 and 6

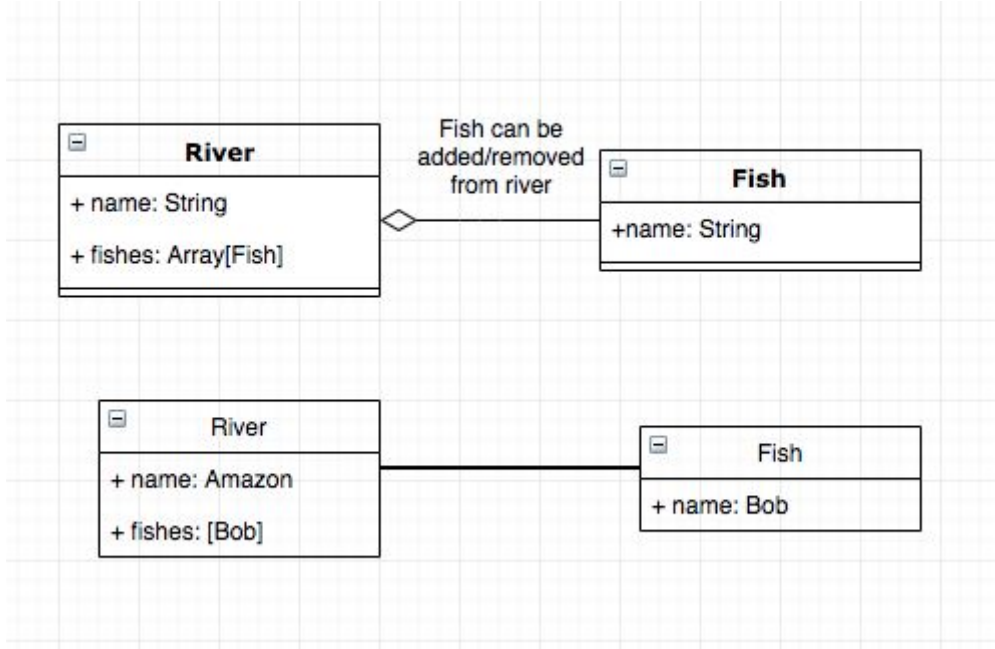
Unit	Ref	Evidence	
A&D	A.D.1	A Use Case Diagram	
		Description: Shows to the left what a customer and technician would need of a booking system and to the right what the company using the system needs from the criteria in the middle.	



Unit	Ref	Evidence	
A&D	A.D.2	A Class Diagram	
		Description: This diagram shows three classes, a Bear, Fish and a River. It shows the instance variables which each class has and the methods which are available to each class and the relationship between the classes.	

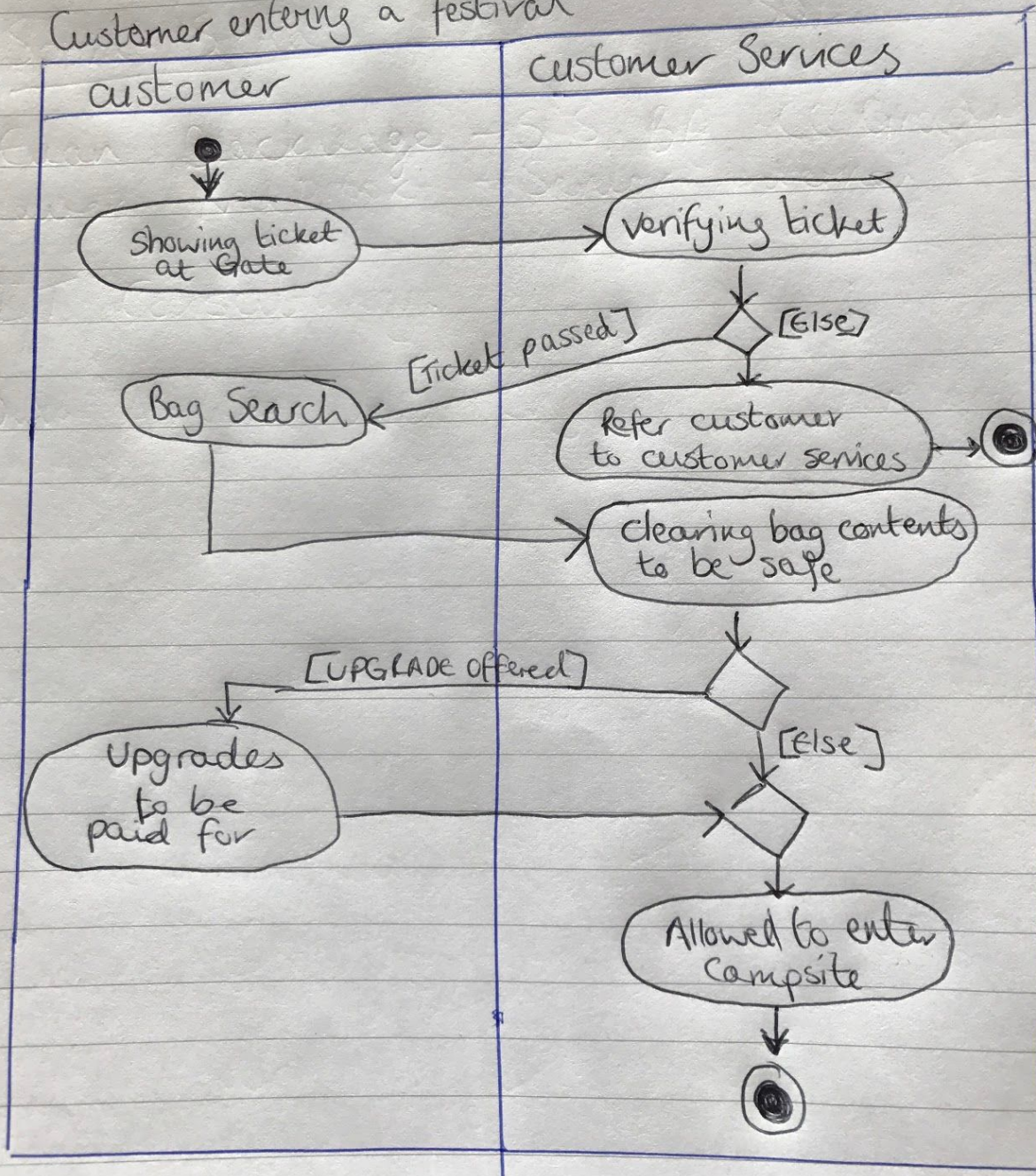


Unit	Ref	Evidence	
A&D	A.D.3	An Object Diagram	
		Description: This is an Object Diagram showing the interaction between 2 instances of the River and Fish classes. The instance variables of each class have been filled in with the details of the objects.	



Unit	Ref	Evidence	
A&D	A.D.4	An Activity Diagram	
		Description: This is an Activity Diagram showing a customer entering a festival. The customer shows their ticket at the gate and if their ticket is valid they get their bag searched, if their bag contents are cleared to be safe they can either opt for an upgrade and pay for it before entering the campsite or can just enter the campsite without upgrading their ticket.	

Customer entering a festival



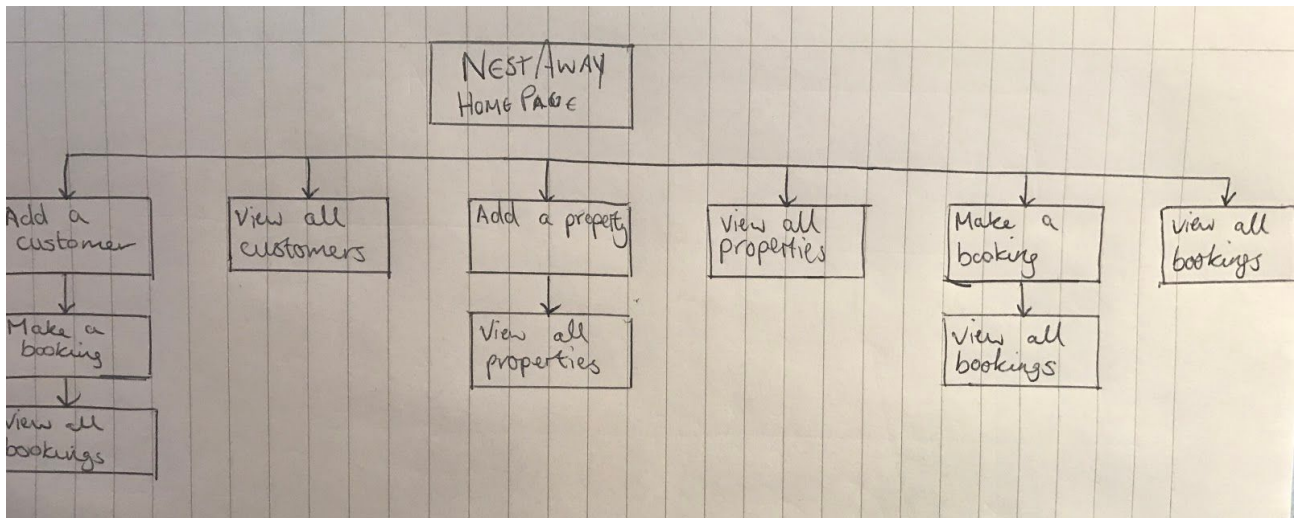
Unit	Ref	Evidence
------	-----	----------

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:

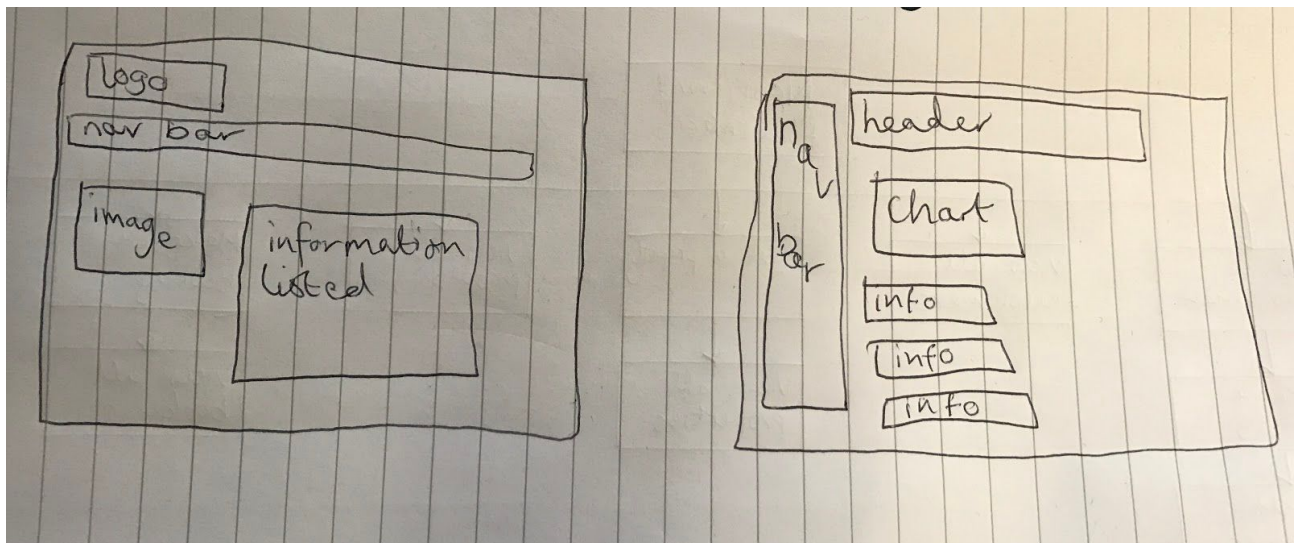
Constraint Category	Implementations Constraints	Solution
Hardware and Software Platforms	User's using different web browsers may not be able to view certain information due to the different formats than what the code was designed for. As this can lead to user's not wanting to use your site due to it not being clear to use or working for their needs.	Make sure testing is done on all platforms to ensure this does not happen.
Performance Requirements	Use of an external server can cause issues if it goes down making the user not able to access or use the website's information that is given from this.	Make sure you choose an the most reliable external server available to you.
Persistent Storage and Transactions	That your local server does not have enough storage space. This can lead to user's not being able to register as customers causing a knock on effect that means they will either give up and go to another competitor or phoning for help which increases costs.	Monitor servers to make sure always enough space for your scope of business. If in need of more purchase further servers to solve this issue.
Usability	User maybe visually impaired and if the website is not made clear and readable with use of a screen reader and given an audio file this won't be a section of the customer market available to the business is completely cut off	By implementing a screen reader and making sure webpage is clear and clean looking making it easier to navigate.

	and not able it be made profit from.	
Budget	The small start budget means would need instant profits quickly or further investment to allow the scope to increase which in turn will increase profits.	Build the base mvp of the project quickly so it can go out on to market and start being used by customers and attracting more potential investors.
Time	Taking too long to produce a working mvp which can could lead to company running out of money paying wages before it is complete to cycle money back into the business.	Make sure a structure plan is made and stuck to to ensure mvp is delivered on time.

Unit	Ref	Evidence	
P	P.5	User Site Map	
		<p>Description: This is a User Site Map showing the user's journey through the NestAway property booking system website. It shows how they can click and move from page to page of the website. This was done so that the website was created using RESTful routes.</p>	



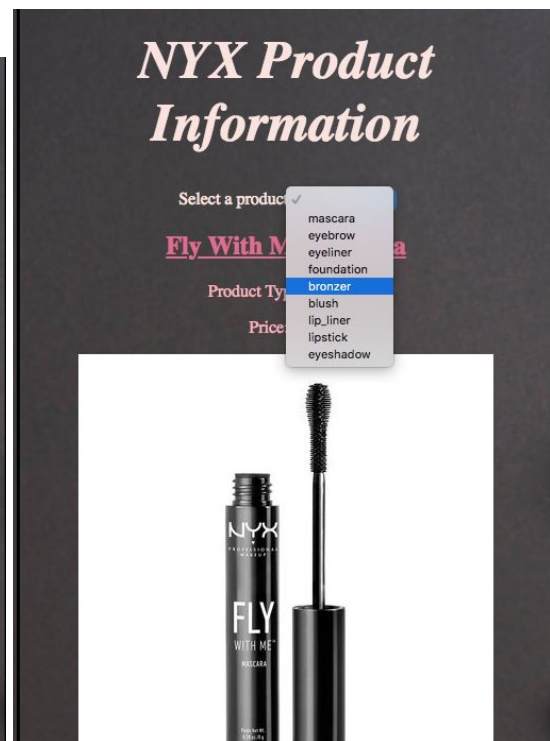
Unit	Ref	Evidence	
P	P.6	2 Wireframe Diagrams	
		Description: Below are 2 wireframes, one on the left is of the NestAway website and the one on the right is of the Stock portfolio website. They show the positions desired when designing the website. They were used to help stick to the initial agreed upon design of the webpages.	



Unit	Ref	Evidence	
P	P.10	Example of Pseudocode used for a method	
		Description: Using pseudocode allows you to write in english everything you would like a method to do to be able find the certain cakes you would like.	

```
def find_cakes()
  # get list of cakes
  # filter list of cakes you want
  # return filtered list
end
```

Unit	Ref	Evidence	
P	P.13	<p>Show user input being processed according to design requirements. Take a screenshot of:</p> <ul style="list-style-type: none"> * The user inputting something into your program * The user input being saved or used in some way 	
		<p>Description: The first image on the top left shows the start page with a drop down menu of options of data to be entered and searched by to show the filtered results based on this choice. The second image shows that bronzer has been selected. The third image shows the list of bronzers being displayed.</p>	



NYX Product Information

Select a product: bronzer

Tango With Bronzing Powder

Product Type: bronzer

Price: \$10.0



Unit	Ref	Evidence	
P	P.14	<p>Show an interaction with data persistence. Take a screenshot of:</p> <ul style="list-style-type: none"> * Data being inputted into your program * Confirmation of the data being saved 	
		<p>Description: The first screenshot shows the list displayed on webpage of members in the gym, the second shows the input page for adding a new member and it being entered, the third is showing that the data has been saved by displaying an updated list of the members to include the new information.</p>	

All Members

Create New Member

Member

[Chris Wright](#)

[Bekah Dixon](#)

[leah Hope](#)

[Logan Dunbar](#)

[Home](#)

[Members](#)

[Sessions](#)

[Instructors](#)

[Bookings](#)

First Name:

Last Name:

Create new member

All Members

Create New Member

Member

[Chris Wright](#)

[Bekah Dixon](#)


[leah Hope](#)

[Logan Dunbar](#)

[Sam Smith](#)

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: The first screenshot shows the start page and the second shows the selected output choice in the dropdown and the correct information displayed below accordingly.	

NYX Product Information

Select a product: 

Fly With Me Mascara

Product Type: mascara

Price: \$9.0



NYX Product Information

Select a product:

#LOTD Lip Of The Day Liquid Lip Liner

Product Type: lip_liner

Price: \$7.0



Unit	Ref	Evidence	
P	P.11	Take a screenshot of one of your projects where you have worked alone and attach the Github link.	
		Description: https://github.com/rebekahdixon/gym_project	

[Home](#)
[Members](#)
[Sessions](#)
[Instructors](#)
[Bookings](#)

Gym Admin System



Unit	Ref	Evidence	
P	P.12	Take screenshots or photos of your planning and the different stages of development to show changes.	
		Description: I only did basic planning to decide what each table would contain and what information and functionality I would need to make the system work. I did not stray from this plan throughout the project.	

Gym

Admin System

Member
id
first name
last name
add / delete member

Session
id
time slot
instructor id
name
create new session

Instructor
name
add / delete instructor

Booking
member id
session id
add / delete booking

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:	

```
1  const RequestHelper = require('../helpers/request_helper.js');
2  const PubSub = require('../helpers/pub_sub.js');
3
4  const Makeup = function () {
5    this.data = [];
6  };
7
8  Makeup.prototype.getData = function () {
9    const url = `http://makeup-api.herokuapp.com/api/v1/products.json?brand=nyx`;
10   const request = new RequestHelper(url);
11   request.get()
12   .then((makeups) => {
13     this.data = makeups;
14     PubSub.publish('Makeups:all-ready', this.data);
15   })
16 };
17
18
```

```
Makeup.prototype.getDataForProductType = function (makeupType) {
  const url =
    `http://makeup-api.herokuapp.com/api/v1/products.json?brand=nyx&product_type=${makeupType}`;
  const request = new RequestHelper(url);
  request.get()
  .then((makeups) => {
    this.data = makeups;
    PubSub.publish('Products:info-loaded', this.data);
    console.log(makeups);
  });
};

module.exports = Makeup;
```



Unit	Ref	Evidence	
P	P.18	<p>Demonstrate testing in your program. Take screenshots of:</p> <ul style="list-style-type: none"> * Example of test code * The test code failing to pass * Example of the test code once errors have been corrected * The test code passing 	
		<p>Description: The first two screenshots are of the tests that are to run, the third is of the code and commented errors that are wrong with it, the last two are the screenshots of all the tests failing then passing after the code was corrected.</p>	

```

class CardGameTest < MiniTest::Test

  def setup

    @card1 = Card.new("hearts", 9)
    @card2 = Card.new("diamonds", 3)
    @card3 = Card.new("spades", 9)
    @card4 = Card.new( "clubs", 1)

    @cards1 = [@card1, @card2, @card3, @card4]
    @cards2 = []

    @cardgame1 = CardGame.new()
  end

  def test_checkforace__true
    expected = true
    actual = @cardgame1.checkforace(@card4)
    assert_equal(expected, actual)
  end

  def test_checkforace__false
    expected = false
    actual = @cardgame1.checkforace(@card3)
    assert_equal(expected, actual)
  end

  def test_highest_card
    expected = @card1
    actual = @cardgame1.highest_card(@card1, @card2)
    assert_equal(expected, actual)
  end
end

```

```

def test_cards_total__array_with_cards
  expected = "You have a total of 22"
  actual = CardGame.cards_total(@cards1)
  assert_equal(expected, actual)
end

def test_cards_total__array_empty
  expected = "You have a total of 0"
  actual = CardGame.cards_total(@cards2)
  assert_equal(expected, actual)
end

end

```

```

require_relative('card.rb')
class CardGame

  #no def initialize to get info to use
  #no attr reader/writers

  def checkforAce(card)
    #use of captial in Ace should be checkforace
    if card.value = 1
      #needs == not =
      return true
    else
      return false
    end
  end

  def highest_card(card1 card2)
    #should be def not dif
    #missing comma between card1 and card 2
    if card1.value > card2.value
      return card.name
      #no method for name
      #should be card1 not card.name
    else
      card2
    end
  end

  def self.cards_total(cards)
    total
    #no value for total
    for card in cards
      total += card.value
      return "You have a total of" + total
    end
  end
end

```

```
require_relative('card.rb')
class CardGame

  def checkforace(card)
    if card.value == 1
      return true
    else
      return false
    end
  end

  def highest_card(card1, card2)
    if card1.value > card2.value
      return card1
    else
      return card2
    end
  end

  def self.cards_total(cards)
    total = 0
    for card in cards
      total += card.value
    end
    return "You have a total of " + total.to_s
  end
end
```


→ PDA_Static_and_Dynamic_Task_A git:(master) ✖ ruby card_spec.rb

Run options: --seed 19645

Running:

FFFFF

Finished in 0.027370s, 182.6818 runs/s, 182.6818 assertions/s.

1) Failure:

CardGameTest#test_checkforace__true [card_spec.rb:24]:

Expected: false

Actual: true

2) Failure:

CardGameTest#test_cards_total__array_empty [card_spec.rb:48]:

Expected: "You have a total of 2"

Actual: "You have a total of 0"

3) Failure:

CardGameTest#test_checkforace__false [card_spec.rb:30]:

Expected: true

Actual: false

4) Failure:

CardGameTest#test_highest_card [card_spec.rb:36]:

--- expected

+++ actual

@@ -1 +1 @@

-#<Card:0XXXXXX @suit="diamonds", @value=3>

+#<Card:0XXXXXX @suit="hearts", @value=9>

5) Failure:

CardGameTest#test_cards_total__array_with_cards [card_spec.rb:42]:

Expected: "You have a total of 21"

Actual: "You have a total of 22"

5 runs, 5 assertions, 5 failures, 0 errors, 0 skips

→ PDA_Static_and_Dynamic_Task_A git:(master) ✖

→ PDA_Static_and_Dynamic_Task_A git:(master) ✖ ruby card_spec.rb

Run options: --seed 34339

Running:

.....

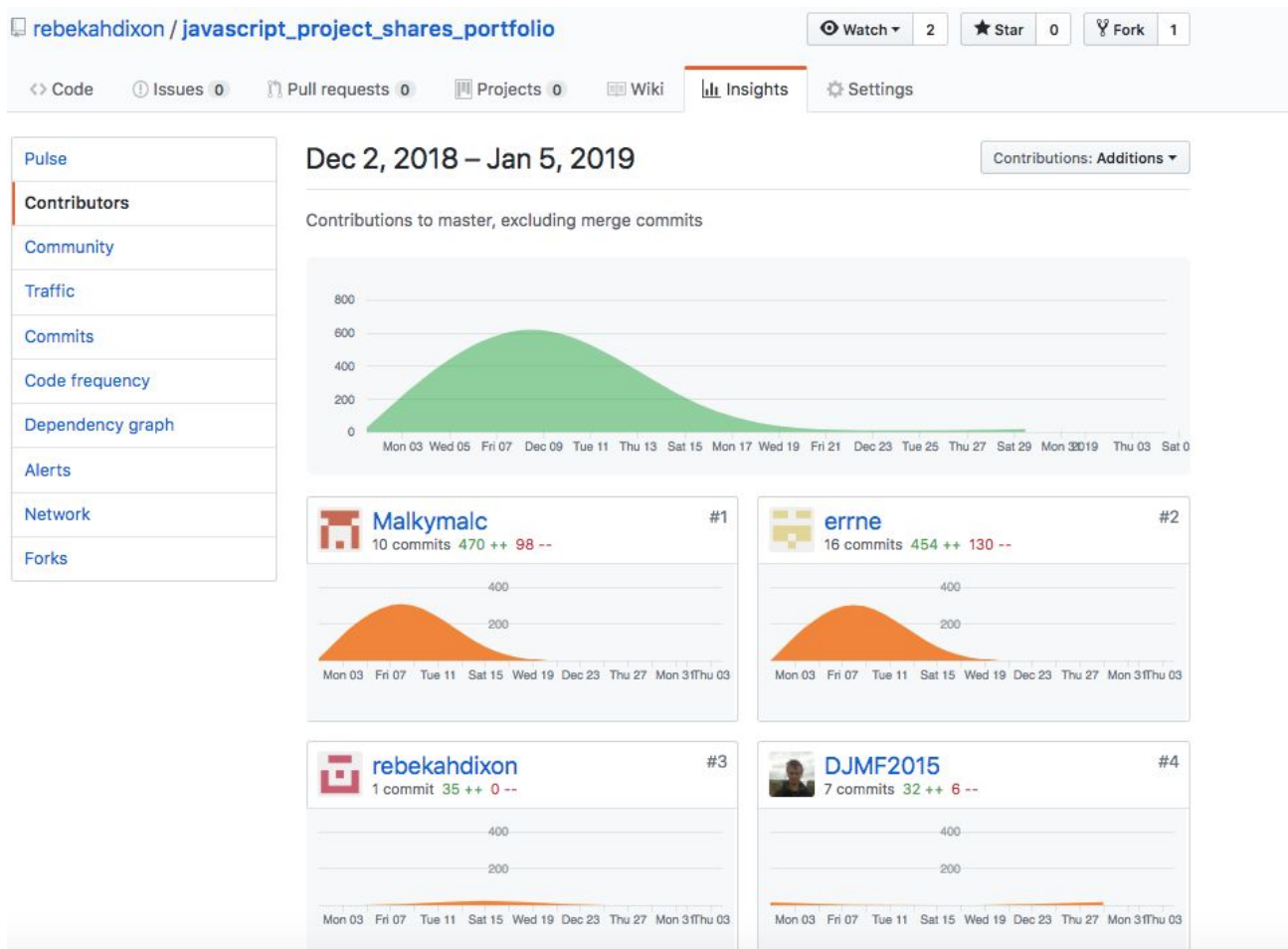
Finished in 0.001541s, 3244.6466 runs/s, 3244.6466 assertions/s.

5 runs, 5 assertions, 0 failures, 0 errors, 0 skips

→ PDA_Static_and_Dynamic_Task_A git:(master) ✖

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: I created the master branch from my laptop and we worked on others to complete the project. Which was to build a Shares Portfolio.	



Unit	Ref	Evidence	
P	P.2	Take a screenshot of the project brief from your group project.	
		Description:	

Shares Portfolio Project

Purpose

A local trader has come to you with a portfolio of shares. She wants to be able to analyse it more effectively. She has a small sample data set to give you and would like you to build a Minimum Viable Product that uses the data to display her portfolio so that she can make better decisions.

MVP

A user should be able to:

view total current value. view individual and total performance trends. retrieve a list of share prices from an external API and allow the user to add shares to her portfolio. View a chart of the current values in her portfolio.

Extensions

Change in price to also be shown in different colours depending on the change. Made it useable on mobile aswell as web.

Unit	Ref	Evidence	
P	P.3	Provide a screenshot of the planning you completed during your group project, e.g. Trello MOSCOW board.	
		Description: Every day we would write up tasks on a whiteboard and divide them up, set a sprint time and meet make to discuss where we all were work wise.	

[3.30 pm] 12/12/18

Things To Do

- Total Portfolio Value - Malcolm
- Edit [Add to Portfolio] button - Bekah
- Render ^{Pie} Charts - Ernest
- Round & Amounts - David

MVP

Extensions Bekah

- Format Change displays [port list, stock list, tabular summary] for + ad - (green) (red)
- Add ^{Hi} day ^{low} info to sidebar & mobile nav - Line Charts - Ernest

Malcolm

Charts

£7.686666

£7.68

Unit	Ref	Evidence	
P	P.4	Write an acceptance criteria and test plan.	
		This shows some of the main features the user would want from the website and how it was tested to see if it would work.	

Acceptance Criteria	Expected Result	Pass/Fail
A user is able to view all customers.	Select "customers" in nav bar then be shown a list of all customers.	Pass
A user is able to view all properties.	Select "properties" in nav bar then be shown a list of all properties.	Pass

A user is able to add a customers.	Enter information about a customer signing up then save this information into the database.	Pass
A user is able to make a booking.	Select a customer from a drop down list of all customers, enter details to filter all properties for available ones according to the filter criteria, select a property from this list produced and then save this information to the database.	Pass

Unit	Ref	Evidence	
P	P.7	Produce two system interaction diagrams (sequence and/or collaboration diagrams).	
		Description: The first collaboration diagram is of the steps a customer makes in entering a festival. The second collaboration is of the steps a customer makes buying a drink in the wine bar.	

Initialization

~~:customer~~

1: show ticket at gate
to event staff

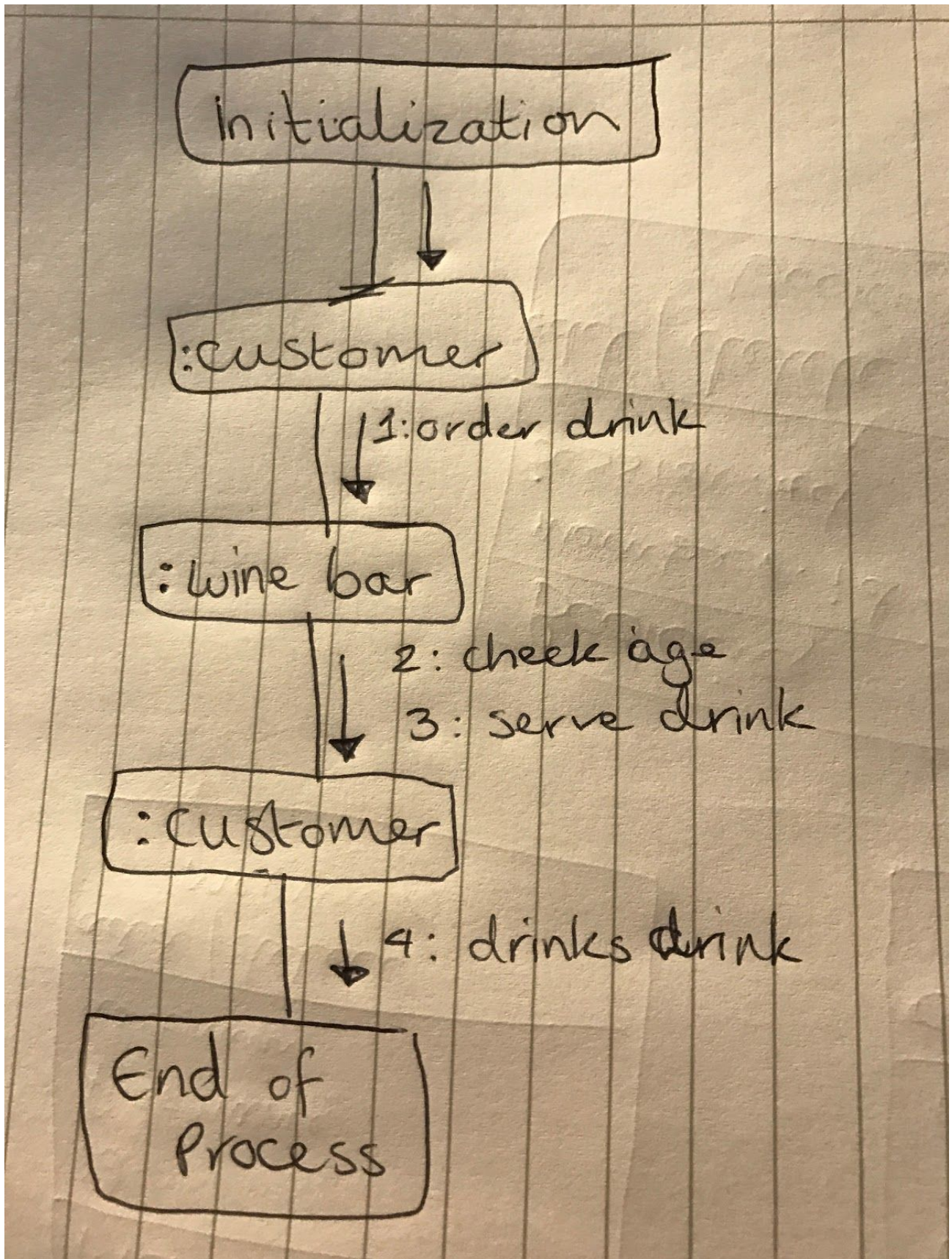
:event staff

2: verify ticket
3: bag search
4: offer upgrade

:customer

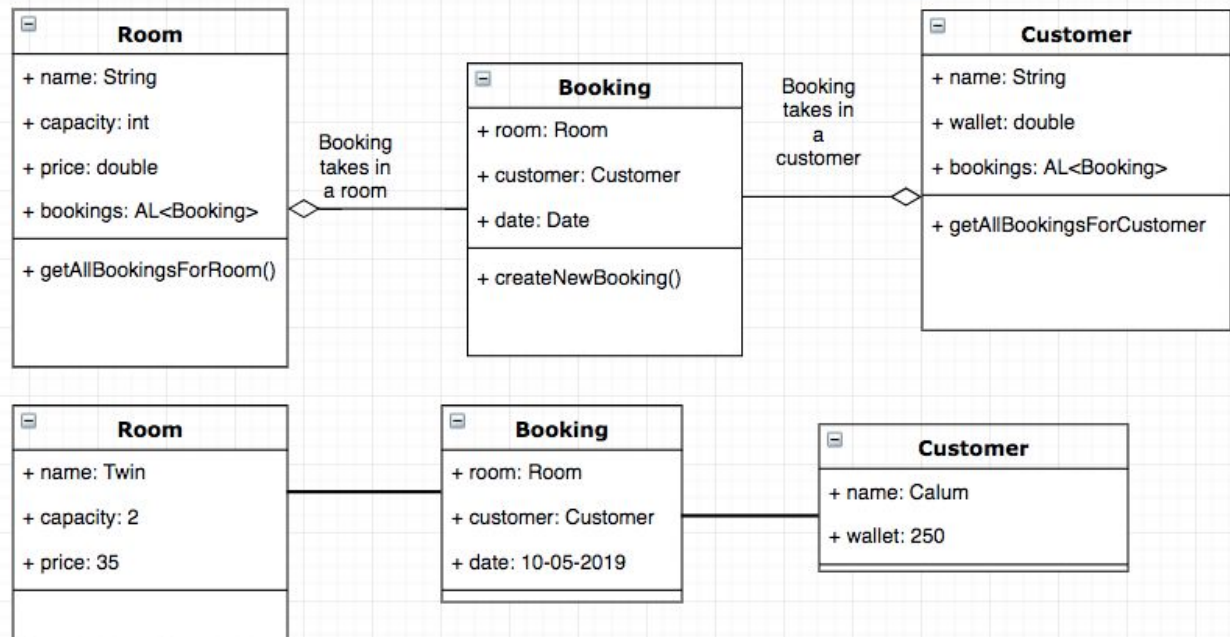
5: take upgrade
6: enter campsite.

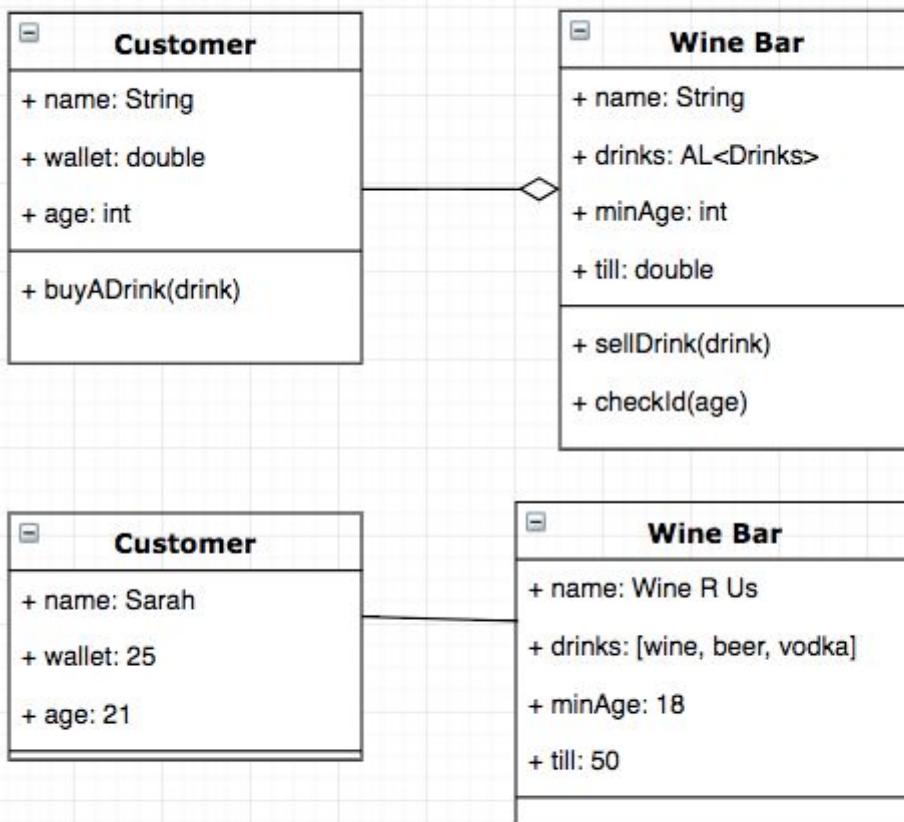
End of Process



Unit	Ref	Evidence	
P	P.8	Produce two object diagrams.	

Description: These are Object Diagrams, the first showing the interaction between 3 instances of the Room, Booking and Customer classes. The second showing the interaction between Customer and Wine Bar classes. The instance variables of each class have been filled in with the details of the objects.





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

Bug/Error	Solution	Date
First name was not appearing on user's information displayed.	Added firstName to the parameters being sent through and it then displayed on screen.	2/2/2019
Spring would not run due to this error: org.hibernate.TransientPropertyValueException	This error is corrected by making sure that the instance of the Class was saved to the database which it turns out it was not.	3/2/2019
null pointer exception error again within Spring.	Corrected the mistake of not having 'new'ed up the list in the model constructor	3/2/2019

Week 12

Unit	Ref	Evidence	
I&T	I.T.7	The use of Polymorphism in a program and what it is doing.	
		Description: The first screenshot shows that the Shop takes in and ArrayList of ISell as its stock and that any object that implements the ISell interface can be added to this ArrayList. The second and third screenshots show the Item class which creates many kinds of Item with a change of ItemType to know what kind of item it is and the fourth then shows the Instrument class which passes down from inheritance to its children the ISell. Finally, a screenshot of the ISell interface.	

```
public class Shop {  
    private String name;  
    private ArrayList<ISell> stock;  
    private double till;  
  
    public Shop(String name, double till) {  
        this.name = name;  
        this.stock = new ArrayList<>();  
        this.till = 100;  
    }  
  
    public void setTill(double till) { this.till = till; }  
  
    public String getName() { return name; }  
  
    public ArrayList<ISell> getStock() { return stock; }  
  
    public void addStockItem(ISell stockItem){  
        stock.add(stockItem);  
    }  
}
```

```
import behaviours.ISell;

public abstract class Item implements ISell {

    private ItemType type;
    protected double priceSell;
    protected double priceBuy;

    public Item(ItemType type, double priceSell, double priceBuy) {
        this.type = type;
        this.priceSell = priceSell;
        this.priceBuy = priceBuy;
    }

    public double getPriceSell() { return priceSell; }

    public double calculateMarkUp() { return priceSell - priceBuy; }
}
```

```
public enum ItemType {
    DRUMSTICKS("Drumsticks"),
    GUITARSTRINGS("Guitar Strings"),
    SHEETMUSIC("Sheet Music");

    private final String name;

    ItemType(String name) {
        this.name = name;
    }
}
```

```
public abstract class Instrument extends Item implements ISell, IPlay {
    private String color;
    private String material;
    private FamilyType family;
    private KindType type;

    public Instrument(ItemType type, double priceSell, double priceBuy) { super(type, priceSell, priceBuy); }

    public String getColor() { return color; }

    public String getMaterial() { return material; }

    public FamilyType getFamily() { return family; }

    public KindType getType() { return type; }

    public double getPriceBought() { return priceBuy; }

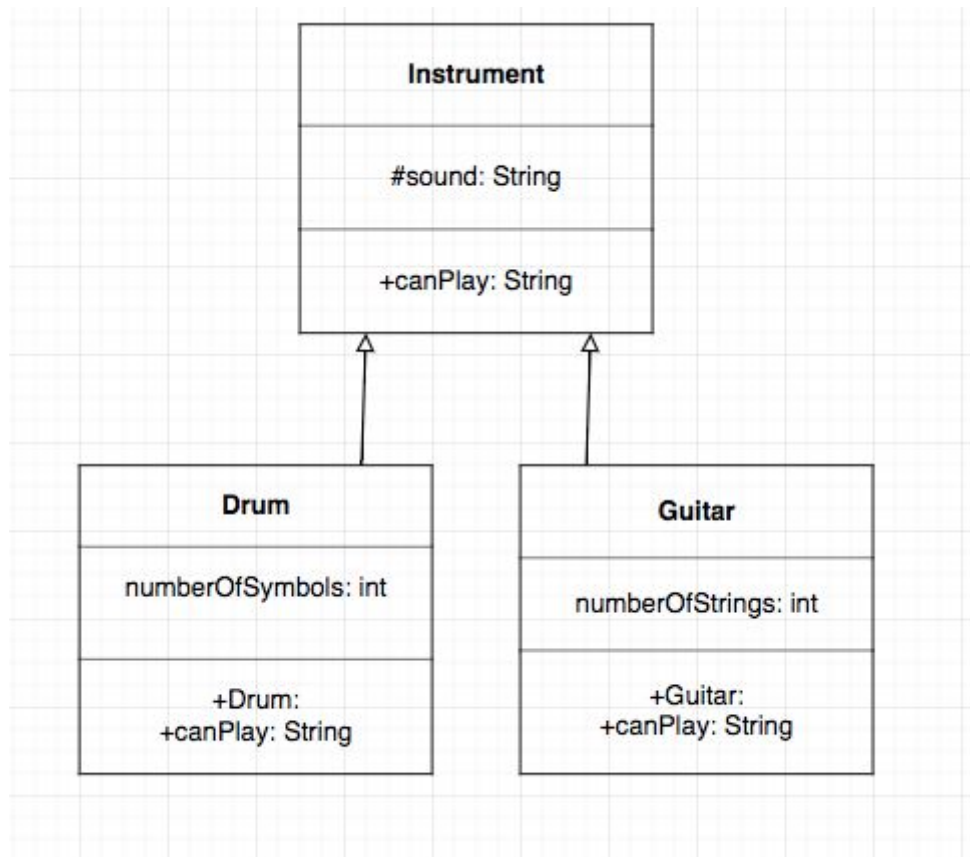
    public double getPriceSell() { return priceSell; }

    public double calculateMarkUp() { return priceSell - priceBuy; }
}
```

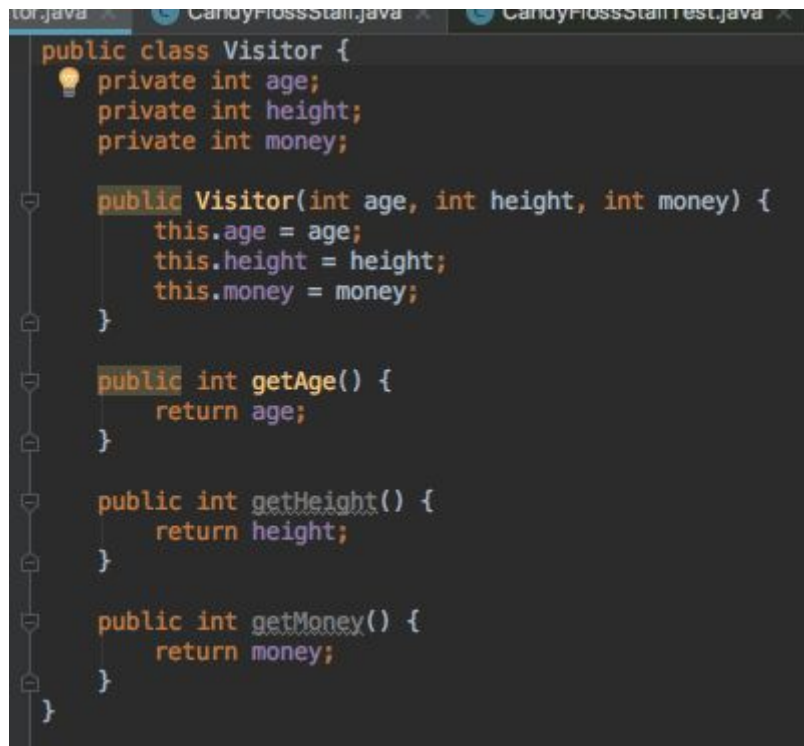
```
public interface ISell {
    public double calculateMarkUp();
}
```

Unit	Ref	Evidence	
------	-----	----------	--

A&D	A.D.5	An Inheritance Diagram
		Description: The Drum and Guitar classes both inherit sound as an instance variable and the canPlay method from the Instrument Class.



Unit	Ref	Evidence	
I&T	I.T.1	The use of Encapsulation in a program and what it is doing.	
		Description: Encapsulation affects how you can access an attribute. In this example money, age and height are defined as “private” which means you cannot read or set them to new values. To do these things you need to define the methods <code>getMoney</code> , <code>getAge</code> and <code>getHeight</code> . Writing these methods allow you to access the information of the money, age and height of the visitor.	



```

public class Visitor {
    private int age;
    private int height;
    private int money;

    public Visitor(int age, int height, int money) {
        this.age = age;
        this.height = height;
        this.money = money;
    }

    public int getAge() {
        return age;
    }

    public int getHeight() {
        return height;
    }

    public int getMoney() {
        return money;
    }
}

```

Unit	Ref	Evidence	
I&T	I.T.2	<p>Take a screenshot of the use of Inheritance in a program. Take screenshots of:</p> <ul style="list-style-type: none"> *A Class *A Class that inherits from the previous class *An Object in the inherited class *A Method that uses the information inherited from another class. 	
		<p>Description: The super-class here is Player, from this the Fighter class inherits all the attributes and methods the Player class holds. Then the Knight class inherits again all the attributes and methods the Fighter class contains and also the ones Fighter inherits from player as well. The last screen shot shows the object knight which was created from the Knight class and shows it has all the attributes and methods passed from Player to Fighter then the two combined and passed to Knight by the tests shown.</p>	

```

public abstract class Player {
    protected double healthpoints;
    private String name;
    protected double coins;

    public Player(double healthpoints, String name, double coins) {
        this.healthpoints = healthpoints;
        this.name = name;
        this.coins = coins;
    }

    public double getCoins() {
        return coins;
    }

    public void setCoins(double coins) {
        this.coins = coins;
    }

    public double getHealthpoints() { return healthpoints; }

    public void setHealthpoints(double healthpoints) { this.healthpoints = healthpoints; }

    public String getName() { return name; }

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

```

```

import players.Player;

public abstract class Fighter extends Player {
    private WeaponType weapon;

    public Fighter(double healthpoints, String name, double coins, WeaponType weapon) {
        super(healthpoints, name, coins);
        this.weapon = weapon;
    }

    public WeaponType getWeapon() { return weapon; }

    public void setWeapon(WeaponType weapon) { this.weapon = weapon; }
}

```



```
import fighters.Fighter;

public class Knight extends Fighter {
    // private double healthpoints;

    public Knight(double healthpoints, String name, double coins, WeaponType weapon) {
        super(healthpoints, name, coins, weapon);
    }

    public void buyArmour(){
        this.healthpoints += 20;
        setCoins(coins -= 30);
    }
}
```

```
public class KnightTest {

    Knight knight;

    @Before
    public void before() { knight = new Knight( healthpoints: 100, name: "Soltaire", coins: 50, WeaponType.SWORD); }

    @Test
    public void canGetArmour(){
        knight.buyArmour();
        assertEquals( expected: 120, knight.getHealthpoints(), delta: 0.01);
        assertEquals( expected: 20, knight.getCoins(), delta: 0.01);
    }
}
```

Unit	Ref	Evidence	
P	P.9	Select two algorithms you have written (NOT the group project). Take a screenshot of each and write a short statement on why you have chosen to use those algorithms.	
		<p>Description: In the first screenshots it shows an algorithm which takes in a selected newenemy and in turn compares each enemy in the ArrayList enemys to check if there is one inside it, if there is it will return it if not it will return null; I chose this as it is a could example of a java for loop.</p> <p>In the second screenshot it shows the fightEnemy algorithm, which is given an enemy and a player, it then gets both of these objects attribute of healthPoints and compares them, if players health points are greater than the enemys then they win else the enemy wins and the player loses, if the player wins it will also have 40 coins added to it and if the player loses its healthPoints are set to 0. I picked this as I was proud of the logic I have written here.</p>	

```
public EnemyType getEnemy(EnemyType newenemy){
    for (EnemyType enemy : enemys){
        if (newenemy == enemy){
            return enemy;
        }
    }
    return null;
}
```

```
public String fightEnemy(EnemyType enemy, Player player) {  
    double playerHP = player.getHealthpoints();  
    double eHP = enemy.getPoints();  
    double playerC = player.getCoins();  
    if ( playerHP > eHP ){  
        player.setHealthpoints(playerHP -= eHP);  
        player.setCoins(playerC += 40);  
        return "Player Wins";  
    }  
    else {  
        player.setHealthpoints(0);  
        return "Enemy Wins. Player is out of HP";  
    }  
}
```