

WfEEK_02

An example of an array in a program

```
# collectionOfNumbers is an array
collectionOfNumbers = [5, 10, 15, 20, 25]
```

A function that uses the array

```
#printNumbers is a function that prints out the array to the console
def printNumbers(collectionOfNumbers)
  puts collectionOfNumbers
end

printNumbers(collectionOfNumbers)
```

The result of the function running

```
[→ week_02 git:(master) ✘ ruby pda.rb
5
10
15
20
25
→ week_02 git:(master) ✘
```

An example of a hash in a program

```
#listOfCountries is a hash of countries and their corresponding regions
listOfCountries = { "Croatia" => "Europe", "Brasil" => "South America", "Uganda" => "Africa"}
```

A function that uses the hash

```
#listOfCountries is a hash of countries and their corresponding regions
listOfCountries = { "Croatia" => "Europe", "Brasil" => "South America", "Uganda" => "Africa" }

#displayListOfCountries is a function that prints out a string with the hash of listOfCountries
def displayListOfCountries(listOfCountries)
  listOfCountries.each do |key, value|
    puts "#{key} is located in #{value}"
  end
end

displayListOfCountries(listOfCountries)
```

The result of the function running

```
[→ week_02 git:(master) ✘ ruby pda.rb
Croatia is located in Europe
Brasil is located in South America
Uganda is located in Africa
→ week_02 git:(master) ✘
```

WEEK_03

An example of a function that searches data

```
#self.find function searches data to find a bottle of whisky by id in database
def self.find( id )
  sql = "SELECT * FROM bottles WHERE id = $1"
  values = [id]
  bottle = SqlRunner.run( sql, values )
  result = Bottle.new( bottle.first )
  return result
end
```

The result of the function running

In the console

```
[1] pry(main)> bottle.find(10)
=> #<Bottle:0x007fa3408a9540 @buy_price=52, @distillery_id=10, @id=10, @name="AILSA BAY", @quantity=50, @sell_price=60, @type="Single Malt">
[2] pry(main)>
```

In the browser

The screenshot shows a web browser window with the URL `localhost:4567/bottles/10`. The page has a header with navigation icons and a title bar. Below the title bar is a navigation menu with links: [Homepage](#), [Stock](#), [Add Stock](#), [Distilleries](#), and [Add Distillery](#). The main content area displays the following information for a bottle of AILSA BAY:

- Brand: AILSA BAY
- Type: Single Malt
- Quantity: 50
- Distillery: Girvan Distillery
- Buy price: 52
- Sell price: 60

At the bottom of the content area is a link: [Return to stock](#).

Function that sorts data

```
#self.all function sorts data to display bottles of whisky by quantity in ascending order
def self.all()
  sql = "SELECT * FROM bottles
  ORDER BY quantity ASC"
  bottle_data = SqlRunner.run(sql)
  bottles = Bottle.map_items(bottle_data)
  return bottles
end
```

The result of the function running

In the console #data is not sorted here

id	name	type	distillery_id	quantity	buy_price	sell_price
10	AILSA BAY	Single Malt	10	50	52	60
11	ARDMORE 20 YEAR OLD 1996	Single Malt	11	55	45	65
13	CHIVAS REGAL 12 YEAR OLD	Blended	13	30	21	34
14	Highland park	Single Malt	12	55	20	40
15	AILSA BAY	Single Malt	14	50	52	60
16	ARDMORE 20 YEAR OLD 1996	Single Malt	15	55	45	65
17	DALMORE CIGAR MALT	Single Malt	16	15	63	85
18	CHIVAS REGAL 12 YEAR OLD	Blended	17	30	21	34
(8 rows)						

In the browser #data is now sorted and ordered by quantity in ascending order with 15 being the smallest quantity in the dataset

[SHOW ORDER](#)

Brand: DALMORE CIGAR MALT
Type: Single Malt
Quantity: 15
Distillery: The Dalmore Distillery
Buy price: £63
Sell price: £85
Stock level: Low (below 20 bottles, order more stock soon!)
Mark up: £22

[UPDATE ORDER](#)

[DELETE DISTILLERY](#)

[SHOW ORDER](#)

Brand: CHIVAS REGAL 12 YEAR OLD
Type: Blended
Quantity: 30
Distillery: Strathisla distillery
Buy price: £21
Sell price: £34
Stock level: Medium (between 20 and 50 bottles)
Mark up: £13

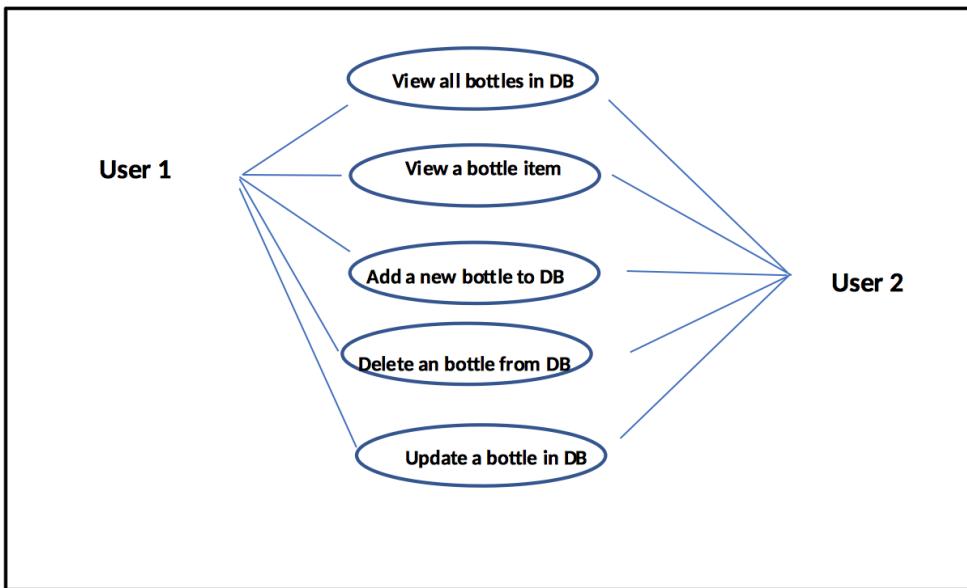
[UPDATE ORDER](#)

[DELETE DISTILLERY](#)

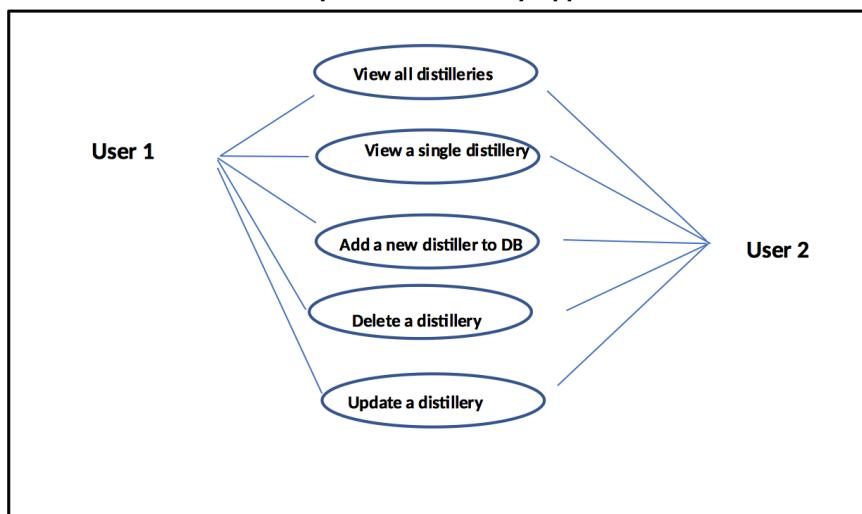
WEEK_05

A use case diagram

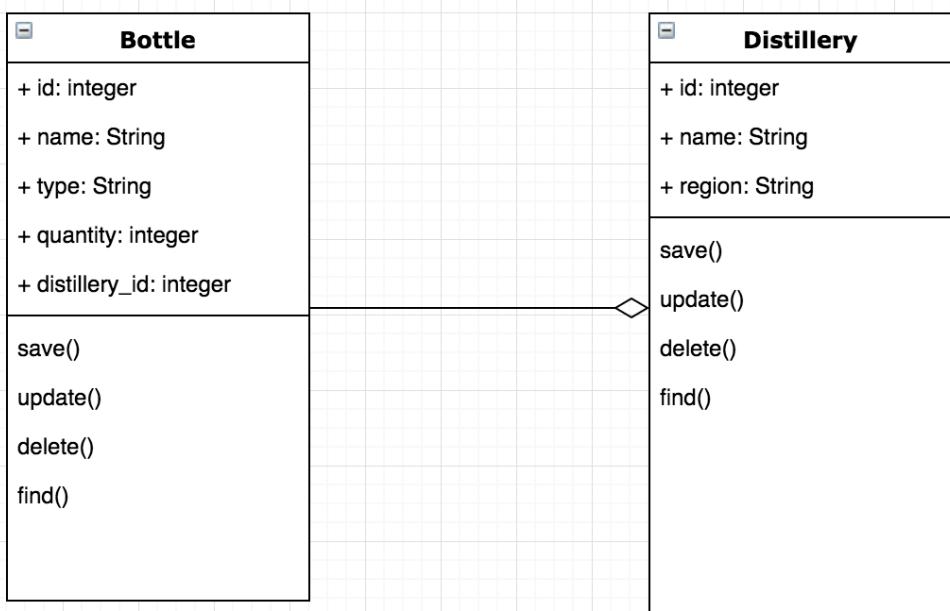
Whisky Villains Inventory App



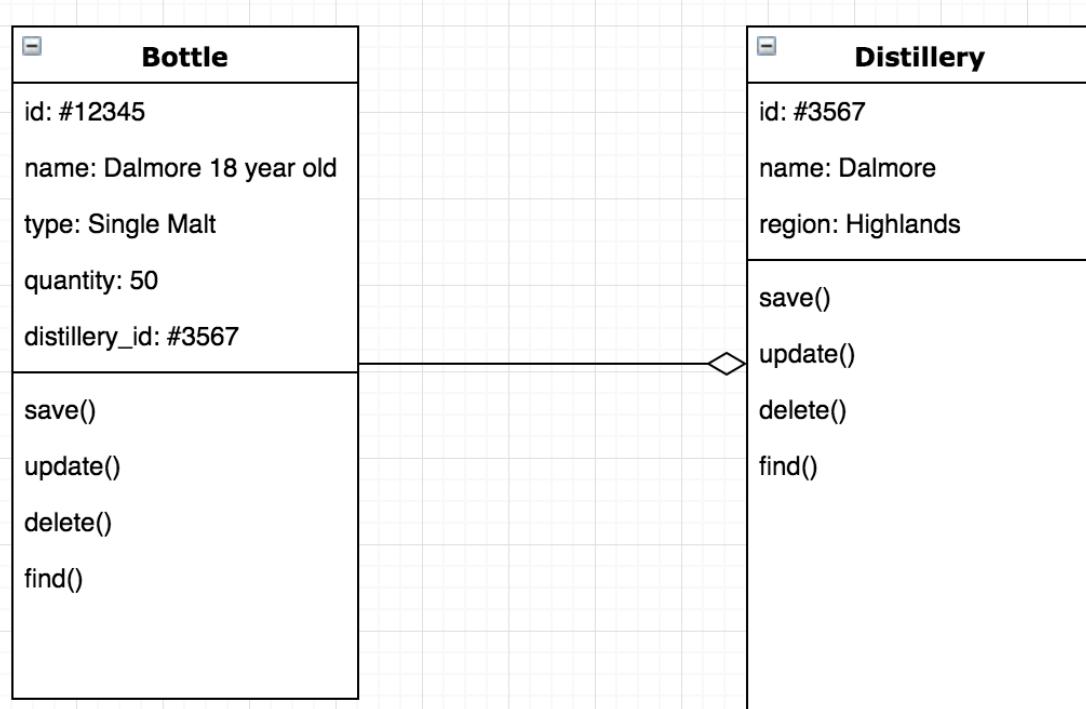
Whisky Villains Inventory App



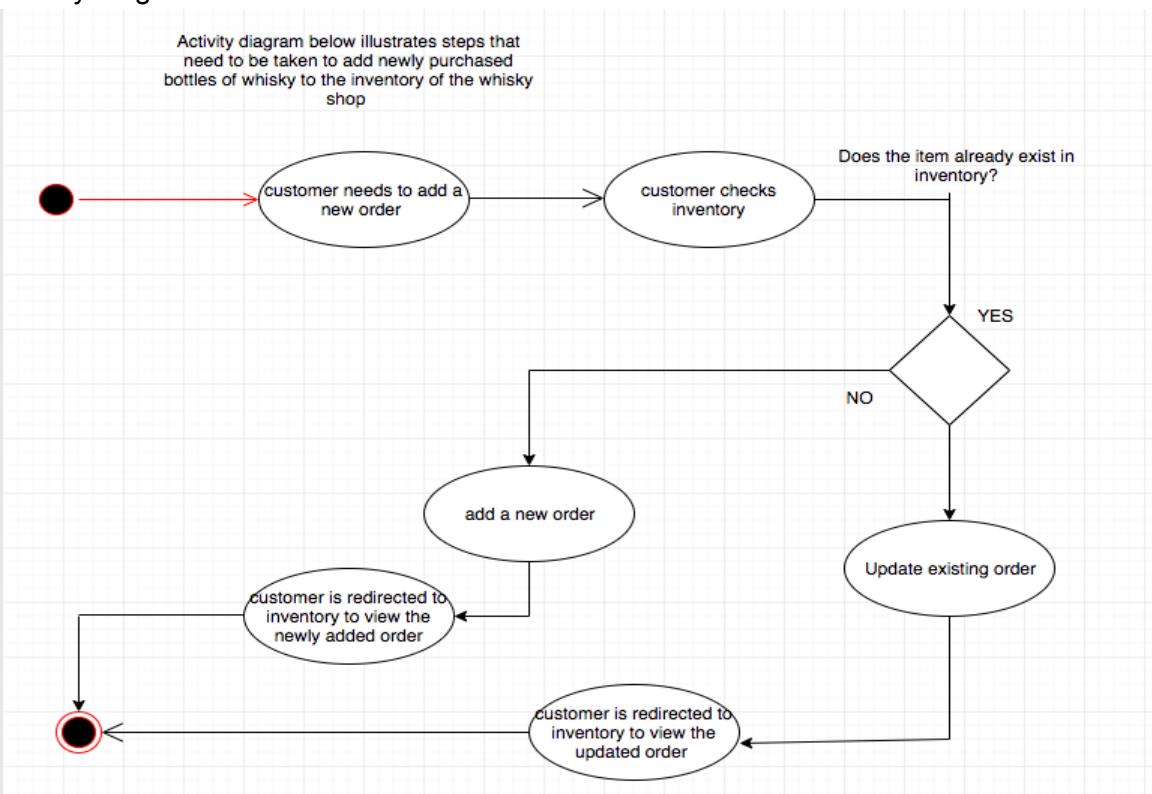
A class diagram



An object diagram



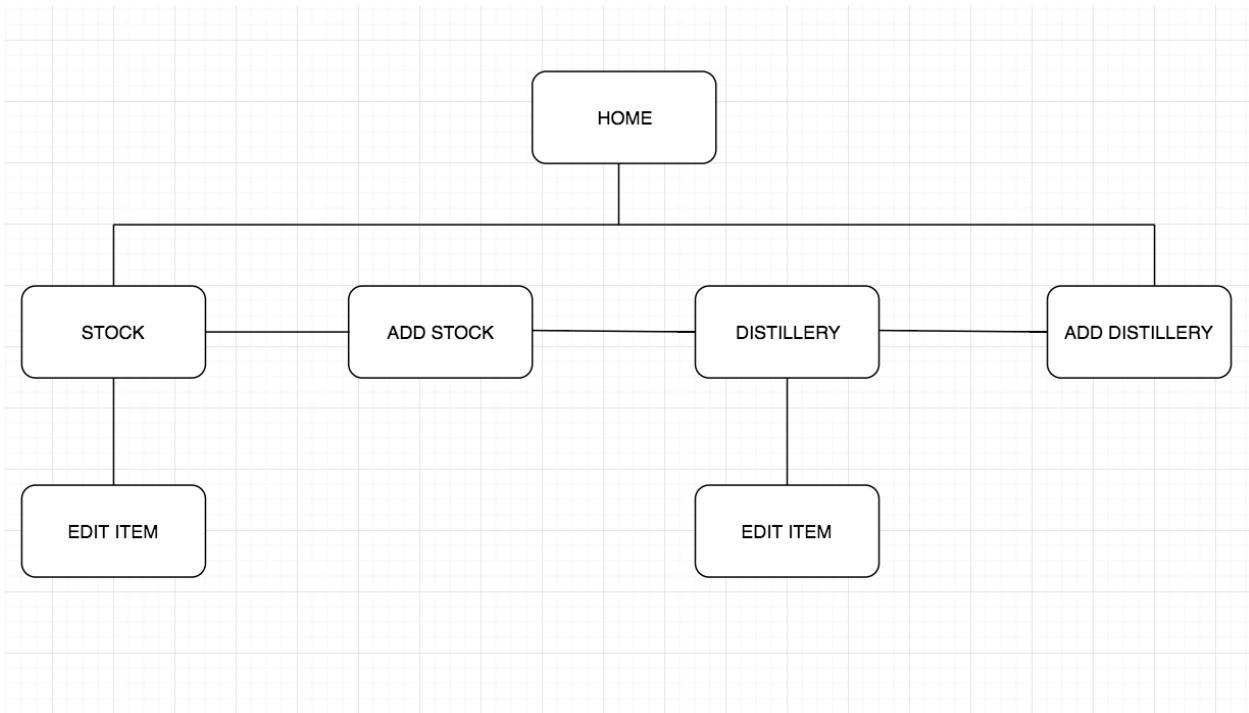
An activity diagram



Implementation constraints plan

Topic	Possible effect of constraint on product	Solution
<i>Hardware and software platforms</i>	Users might not be able to use the whisky inventory app using different types of devices(PC, tablet, phone). This will restrict the usability of the product, limiting access options to the app.	Create a fully responsive app that scales to the viewport of the device the app is accessed on, the look and usability of the app is not restricted.
<i>Performance requirements</i>	The app does not load in every browser or app looks significantly different due to different browsers rendering html & css in different ways.	Use semantic HTML with semantic tags for example <p> for paragraph to make it clear to all browsers that enclosed text between <p> tags refer to paragraphs and must be displayed as such.
<i>Persistent storage and transactions</i>	When the user is using the whisky inventory app, they need to add new whisky bottles to the inventory and that information needs to be stored somewhere for them to review it later. This process of performing CRUD operations on the database must work flawlessly every time to avoid delays and bottlenecks in inventory	All the CRUD operations(CREATE, READ, UPDATE, DELETE) must be thoroughly before releasing the product to the client
<i>Usability</i>	The inventory information must be displayed in a clear, organised and intuitive way in order to avoid user frustration with a non-intuitive confusing layout	Following best practice of UX (user experience) guidelines when designing the layout of the web app
<i>Budgets</i>	Failing to stick to the budget and running out of funds before project completion would lead to an app with limited functionality which will undermine the overall success of the app	Complete the MVP(minimally viable product) first so that even if you run out of budget at some point during the project, you will have a basic functioning app
<i>Time limitations</i>	Failing to stick to deadlines and running out of time before project completion would lead to a half finished app, with undetected bugs present in code due to an incomplete testing cycle	Prioritise planning and perform unit testing throughout the project all completed code is thoroughly tested. As above, prioritise MVP before moving onto extensions.

A site map



Produce two wireframes designs

NinjaMock Project "whisky shop"

export undo redo co

Homepage Stock Add stock Distilleries Add distillery

Stock
Text area

Add stock
Text area

Distilleries
Text area

Add distillery
Text area

FOOTER

Pages: Default group

+ □ + □ -

Homepage Stock Add stock Distillery Add distillery



NinjaMock

Project "whisky shop"

| export | undo | redo

Homepage Stock Add stock Distilleries Add distillery

Current stocklist

Column 1	Column 2	Column 3

Column 1	Column 2	Column 3

Column 1	Column 2	Column 3

FOOTER

Pages: Default group

+ □

+ □ ▾

Homepage Stock Add stock Distillery Add distillery

Take a screenshot of an example of pseudocode for a function

```
// transferDinosaur function needs to transfer a dinosaur between two paddocks  
  
// pass in three arguments to this function: which dino needs to be moved,  
// the paddock that the dino needs to vacate and the paddock that it needs to move to  
  
// remove the dino from the paddock that it needs to vacate  
💡  
// add the dino to the paddock that it needs to move to
```

Show user input being processed according to design requirements. Take a screenshot of:

The user inputting something into your program

A screenshot of a web-based inventory tool. At the top, there is a navigation bar with links: 'Homepage', 'Stock', 'Add stock', 'Distilleries', and 'Add distillery'. Below the navigation bar, a section titled 'New Distillery' contains two input fields: 'Distillery Name' (containing 'Glengoyne Distillery') and 'Region of Distillery' (containing 'Highlands', which is highlighted with a yellow background). A button labeled 'Add Distillery' is located to the right of the region field. At the bottom of the page, a copyright notice reads '(c) 2018 Whisky Villains Inventory Tool'.

A new distillery was successfully added!

A screenshot of the same web-based inventory tool. The top navigation bar and the 'New Distillery' form are identical to the first screenshot. However, the main content area now displays a bold success message: 'A new distillery was successfully added!'. Below this message is a link 'Return to distilleries'. At the bottom of the page, the copyright notice '(c) 2018 Whisky Villains Inventory Tool' is visible.

The user input being saved

Homepage	Stock	Add stock	Distilleries	Add distillery
Name of distillery Region of distillery				
The Dalmore Distillery	Highland	Edit item Delete item		
Strathisla distillery	Highland	Edit item Delete item		
GLENFIDDICH	Dufftown	Edit item Delete item		
Bowmore	Islay	Edit item Delete item		
Jura	Highlands	Edit item Delete item		
Glengoyne Distillery	Highlands	Edit item Delete item		

(c) 2018 Whisky Villains Inventory Tool

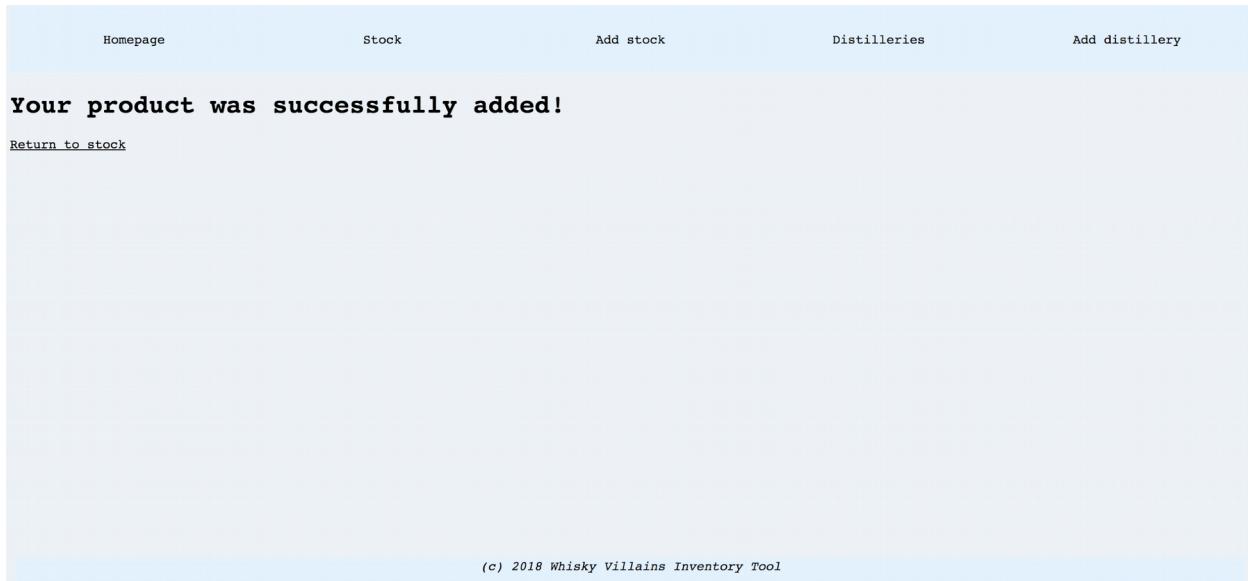
Glengoyne Distillery can be seen at the bottom of the list which corresponds to last item saved to database

Show an interaction with data persistence. Take a screenshot of:

Data being inputted into your program

Homepage	Stock	Add stock	Distilleries	Add distillery	
Distillery	Brand	Type	Quantity	Buy Price	Sell Price(£)
Ardmore distillery	<input type="text" value="Ardmore 20 Year Old"/>	<input type="text" value="Select type"/> Single Malt	<input type="text" value="35"/>	<input type="text" value="50"/>	<input type="text" value="70"/>
Add Stock					

(c) 2018 Whisky Villains Inventory Tool



Confirmation of the data being saved
In the browser

Homepage		Stock		Add stock			Distilleries		Add distillery	
Distillery	Brand	Type	Quantity	Buy price	Sell price	Stock level		Mark up		
Girvan Distillery	AILSA BAY	Single Malt	50	52	60	High (more than 50 bottles)	8	Edit item	Delete item	
Ardmore distillery	ARDMORE 20 YEAR OLD 1996	Single Malt	55	45	65	High (more than 50 bottles)	20	Edit item	Delete item	
The Dalmore Distillery	DALMORE CIGAR MALT	Single Malt	15	63	85	Low (below 20 bottles, order more stock soon!) 22		Edit item	Delete item	
Strathisla distillery	CHIVAS REGAL 12 YEAR OLD	Blended	30	21	34	Medium (between 20 and 50 bottles)	13	Edit item	Delete item	
Ardmore distillery	Ardmore 20 Year Old	Single Malt	35	50	70	Medium (between 20 and 50 bottles)	20	Edit item	Delete item	

(c) 2018 Whisky Villains Inventory Tool

In the terminal

```

→ db git:(master) ✘ psql inventory
psql (10.0)
Type "help" for help.

inventory=# SELECT * FROM bottles;
   id |          name           |      type       | distillery_id | quantity | buy_price | sell_price
---+-----+-----+-----+-----+-----+-----+
   1 | AILSA BAY             | Single Malt | 1 | 50 | 52 | 60
   2 | ARDMORE 20 YEAR OLD 1996 | Single Malt | 2 | 55 | 45 | 65
   3 | DALMORE CIGAR MALT    | Single Malt | 3 | 15 | 63 | 85
   4 | CHIVAS REGAL 12 YEAR OLD | Blended | 4 | 30 | 21 | 34
   5 | Ardmore 20 Year Old    | Single Malt | 2 | 35 | 50 | 70
(5 rows)

inventory=#

```

Show the correct output of results and feedback to user. Take a screenshot of:

The user requesting information or an action to be performed

User clicks on stock link in the navigation bar and expects to be taken to inventory page

Stock
Go to this page to check the quantity of stock you have in your inventory, delete and/or update stock.

Add stock
Go to this page to add a new product to your inventory.

Distilleries
Go to this page to check what distilleries you are currently sourcing your whisky from.

Add distillery
Go to this page to add a new distillery to your list of distilleries.

(c) 2018 Whisky Villains Inventory Tool

The user request being processed correctly and demonstrated in the program

Homepage		Stock		Add stock		Distilleries		Add distillery	
Distillery	Brand	Type	Quantity	Buy price	Sell price	Stock level		Mark up	
Girvan Distillery	AILSA BAY	Single Malt	50	52	60	High (more than 50 bottles)	8	Edit item	Delete item
Ardmore distillery	ARDMORE 20 YEAR OLD 1996	Single Malt	55	45	65	High (more than 50 bottles)	20	Edit item	Delete item
The Dalmore Distillery	DALMORE CIGAR MALT	Single Malt	15	63	85	Low (below 20 bottles, order more stock soon!)	22	Edit item	Delete item
Strathisla distillery	CHIVAS REGAL 12 YEAR OLD	Blended	30	21	34	Medium (between 20 and 50 bottles)	13	Edit item	Delete item
Ardmore distillery	Ardmore 20 Year Old	Single Malt	35	50	70	Medium (between 20 and 50 bottles)	20	Edit item	Delete item

(c) 2018 Whisky Villains Inventory Tool

Clicking on “stock” link in navigation bar takes the user to the inventory page displayed in screenshot above

WEEK_6

Demonstrate the use of Polymorphism in a program

```

1 package Behaviors;
2
3 public interface IFly {
4     public String fly();
5 }

```

Pterosaur(flying dinosaurs) abstract class implements IFly interface. As it is an abstract class all its subclasses implement the IFly interface too.

```

1 package FlyingDinos;
2
3 import Behaviors.IFly;
4 import Dinos.Dinosaur;
5
6 public abstract class Pterosaur implements IFly {
7     private String name;
8     private int speed;
9     private int attackValue;
10    private int wingSpan;
11    private String kind;
12
13    public Pterosaur(String name, int speed, int attackValue, int wingSpan, String kind) {
14        this.name = name;
15        this.speed = speed;
16        this.attackValue = attackValue;
17        this.wingSpan = wingSpan;
18        this.kind = kind;
19    }
20
21    public String fly(){
22        return "I can fly!";
23    }
}

```

In the Park class, “IFly” objects are placed into an ArrayList called ‘predators’ to separate them from non flying dinosaurs who, in this scenario, are their prey.

```

1 import Dinos.Dinosaur;
2 import Paddocks.Paddock;
3 import Paddocks.CarnivorePaddock;
4 import Paddocks.HerbivorePaddock;
5 import Behaviors.IFly;
6
7
8 import java.util.ArrayList;
9
10 public class Park {
11
12     private int parkCapacity;
13     private String name;
14     private ArrayList<Paddock> paddocks;
15     private ArrayList<Visitor> visitors;
16     private HerbivorePaddock herbivorePaddock;
17     private HerbivorePaddock herbivorePaddock2;
18     private CarnivorePaddock carnivorePaddock;
19     private Park park;
20     private Visitor visitor;
21     private ArrayList<IFly> predators;
22
23
24     public Park(int parkCapacity) {
25         this.parkCapacity = parkCapacity;
26         this.name = "Ace Ventura";
27         this.predators = new ArrayList<IFly>();
28         this.paddocks = new ArrayList<Paddock>();
29         this.visitors = new ArrayList<Visitor>();
30         this.herbivorePaddock = new HerbivorePaddock( name: "HerbPaddock" , capacity: 10 );
31         this.carnivorePaddock = new CarnivorePaddock( name: "CarnPaddock" , capacity: 15 );
32     }
33
34     public void addToPredatorWatchList(IFly predator) {
35         this.predators.add(predator);
36     }
37
38     public void removeFromPredatorWatchList(IFly predator) {
39         this.predators.remove(predator);
40     }

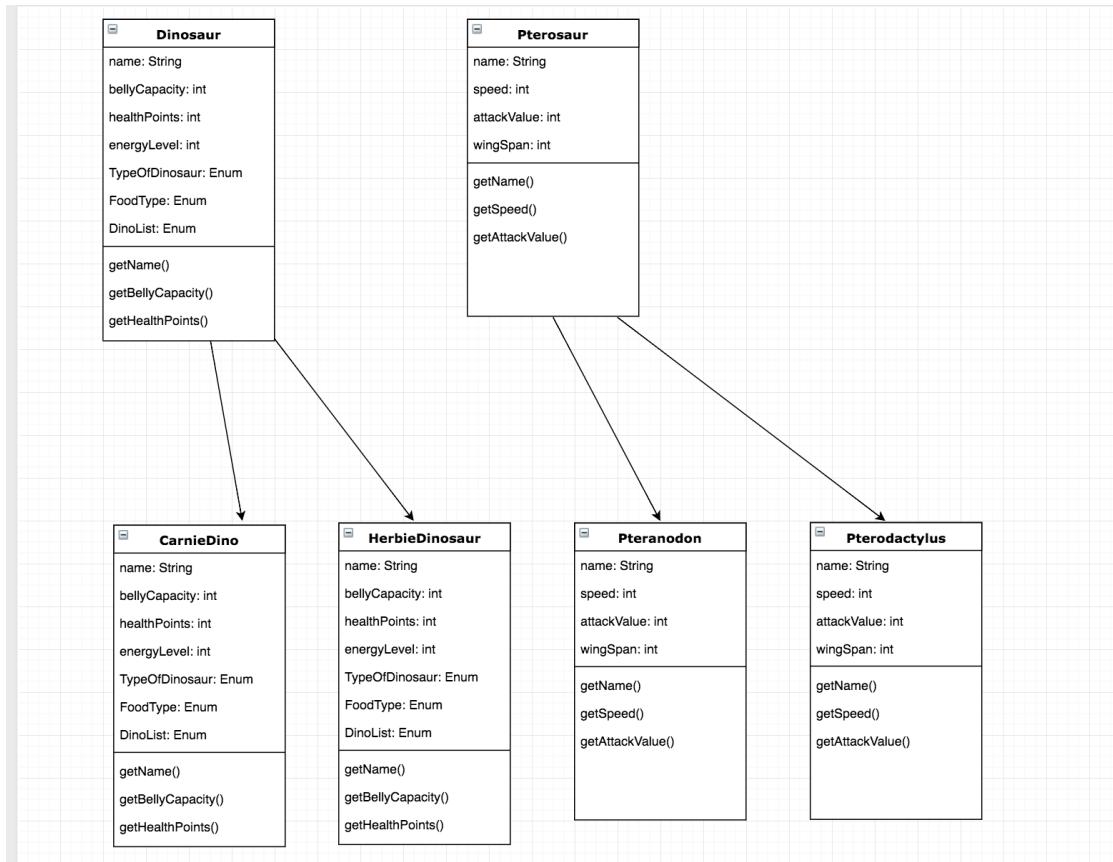
```

`addToPredatorWatchList()` adds `IFly` objects to an `ArrayList` of predators.

`removeFromPredatorWatchList()` removes `IFly` objects from the `ArrayList` of predators.

WEEK_9

Below is an inheritance diagram showing two abstract classes(Dinosaur & Pterosaur) and their respective subclasses



Take an example of encapsulation in a program

```

public abstract class Pterosaur implements IMove {
    private String name;
    private int speed;
    private int attackValue;
    private int wingSpan;
    private String kind;

    public Pterosaur(String name, int speed, int attackValue, int wingSpan, String kind) {
        this.name = name;
        this.speed = speed;
        this.attackValue = attackValue;
        this.wingSpan = wingSpan;
        this.kind = kind;
    }

    public String move(){
        return "I can fly!";
    }

    public String getName() {
        System.out.println("I am " + this.name + ", a flying reptile of type " + this.kind + " and I am here to kick s");
        return name;
    }

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

    public String getKind() { return kind; }

    public int getSpeed() { return speed; }

    public int getAttackValue() {
        System.out.println("I am packing heat and my attack value is " + attackValue + " health points that my enemies");
        return attackValue;
    }
}
  
```

Take a screenshot of the use of Inheritance in a program. Take screenshots of:

Abstract class

```

1 package Dinos;
2
3 import Behaviors. IMove;
4
5 import java.util.ArrayList;
6
7 public abstract class Dinosaur implements IMove {
8
9     private String name;
0     private TypeOfDinosaur type;
1     private ArrayList<Food> belly;
2     private int bellyCapacity;
3     private int healthPoints;
4     private int energyLevel;
5     private Food food;
6     private FoodType foodType;
7     private DinoList dinoKind;
8
9     public Dinosaur(String name, TypeOfDinosaur type, int bellyCapacity, int healthPoints, int energyLevel, DinoList dinoKind) {
0         this.name = name;
1         this.belly = new ArrayList<>();
2         this.type = type;
3         this.bellyCapacity = bellyCapacity;
4         this.healthPoints = healthPoints;
5         this.energyLevel = energyLevel;
6         this.dinoKind = dinoKind;
7     }
8
9     public String move(){
0         return "I am a fast walker but not much of a runner";
1     }
2
3     public String getName() { return this.name; }
4
5     public void setName(String name) { this.name = name; }
6
7     public int getHealthPoints() { return healthPoints; }
8
9     public void setHealthPoints(int healthPoints) { this.healthPoints = healthPoints; }
0
1     public int getEnergyLevel() { return this.energyLevel; }
2
3     public void setEnergyLevel() { this.energyLevel = energyLevel; }
4
5     public TypeOfDinosaur getType() { return this.type; }
6
7 }

```

Subclass that inherit from abstract class “Dinosaur”

```

1 package Dinos;
2
3 public class HerbieDino extends Dinosaur {
4
5     public HerbieDino(String name, TypeOfDinosaur type, int bellyCapacity, int healthPoints, int energyLevel, DinoList dinoKind) {
6         super(name, type, bellyCapacity, healthPoints, energyLevel, dinoKind);
7     }
8
9 }

```

An object in the inherited class

```

package DinoTests;
import ...
public class TestHerbivoreDino {
    HerbieDino herbieDino;
    TypeOfDinosaur typeOfDinosaur;
    Food food;
    FoodType foodType;
    DinoList dinoKind;

    @Before
    public void before() {
        food = new Food(foodType.PLANTS);
        herbieDino = new HerbieDino( name: "Steve", typeOfDinosaur.Herbivore, bellyCapacity: 10, healthPoints: 15, energyLevel: 10, dinoKind: Dinos.Spinosaurus);
    }
}

```

A method that uses the information inherited from another class

```

7  public abstract class Dinosaur implements IMove {
8
9      private String name;
0      private TypeOfDinosaur type;
1      private ArrayList<Food> belly;
2      private int bellyCapacity;
3      private int healthPoints;
4      private int energyLevel;
5      private Food food;
6      private FoodType foodType;
7      private DinoList dinoKind;
8
9      public Dinosaur(String name, TypeOfDinosaur type, int bellyCapacity, int healthPoints, int energyLevel, DinoList dinoKind) {
0          this.name = name;
1          this.belly = new ArrayList<>();
2          this.type = type;
3          this.bellyCapacity = bellyCapacity;
4          this.healthPoints = healthPoints;
5          this.energyLevel = energyLevel;
6          this.dinoKind = dinoKind;
7      }
8
9      public String getName() { return this.name; }
2
3      public void setName(String name) { this.name = name; }

```

getName and setName are both methods on Dinosaur abstract class

```

Q  public class TestHerbivoreDino {
    HerbieDino herbieDino;
    TypeOfDinosaur typeOfDinosaur;
    Food food;
    FoodType foodType;
    DinoList dinoKind;

    @Before
    public void before() {
        food = new Food(foodType.PLANTS);
        herbieDino = new HerbieDino( name: "Steve", typeOfDinosaur.Herbivore, bellyCapacity: 10, healthPoints: 15, energyLevel: 10, dinoKind: 2 );
    }

    @Test
    public void canGetName() { assertEquals( expected: "Steve", herbieDino.getName()); }

    @Test
    public void canSetName(){
        herbieDino.setName("Dave");
        assertEquals( expected: "Dave", herbieDino.getName());
    }
}

```

getName and setName are methods inherited from Dinosaur abstract class

Take a screenshot of one of your projects where you worked alone and attach the Github link

<https://github.com/arthurandreev/full-stack-project-in-ruby>

No description, website, or topics provided.

Branch: master ▾ New pull request Create new file Upload files Find file Clone or download ▾

arthurandreev changed heading on homepage

File	Description	Date
db	basic html and css added to style it up	3 months ago
models	refactored the views and added styling in css	10 days ago
public	refactored views	10 days ago
views	changed heading on homepage	10 days ago
README.md	Project brief	17 days ago
app.rb	refactored the views and added styling in css	10 days ago

<https://github.com/arthurandreev/full-stack-project-in-ruby/graphs/contributors>

Pulse

Contributors

Community

Traffic

Commits

Code frequency

Dependency graph

Network

Forks

Feb 25, 2018 – May 24, 2018

Contributions to master, excluding merge commits

Contributions: Commits ▾

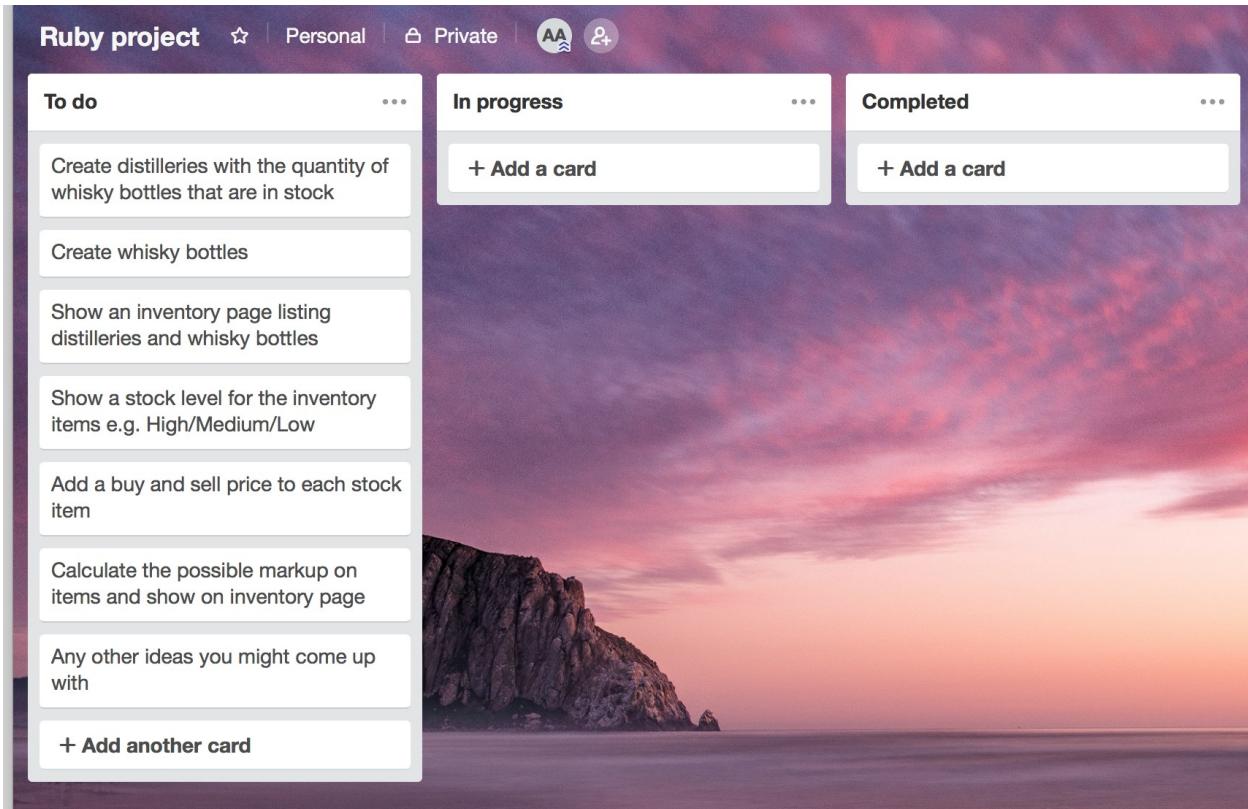
arthurandreev #1

9 commits 356 ++ 234 --

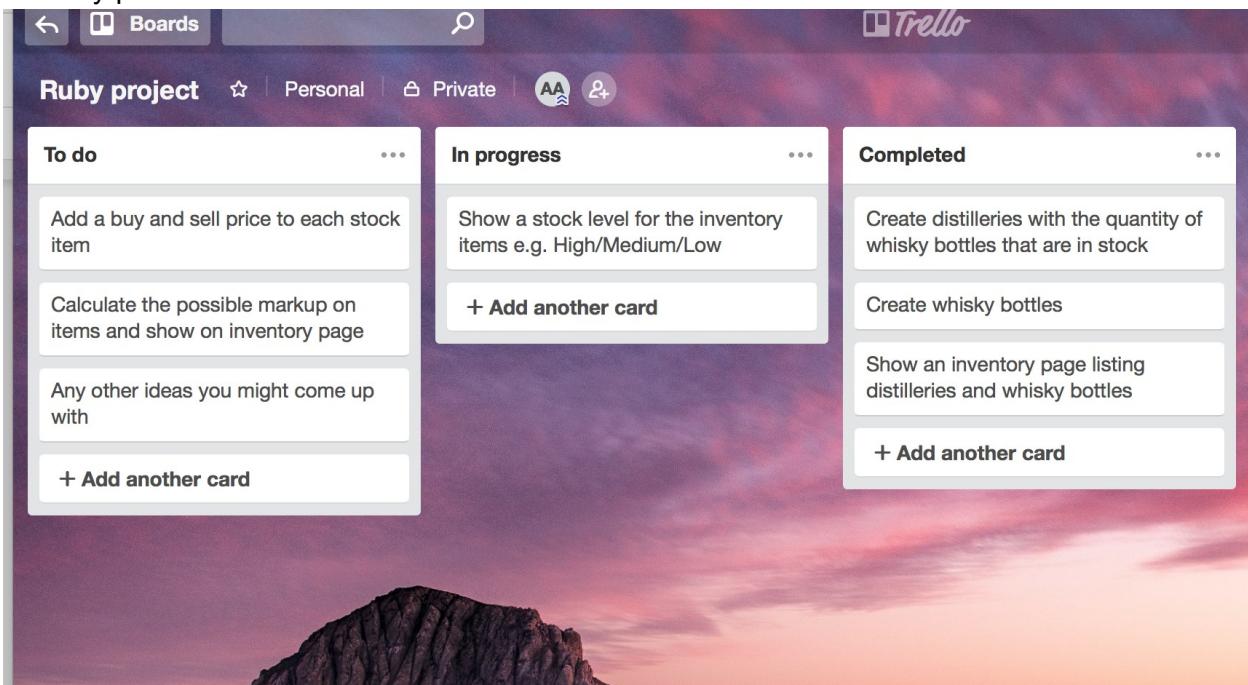
<https://github.com/arthurandreev/full-stack-project-in-ruby>

Take screenshots or photos of your planning and the different stages of development to show changes

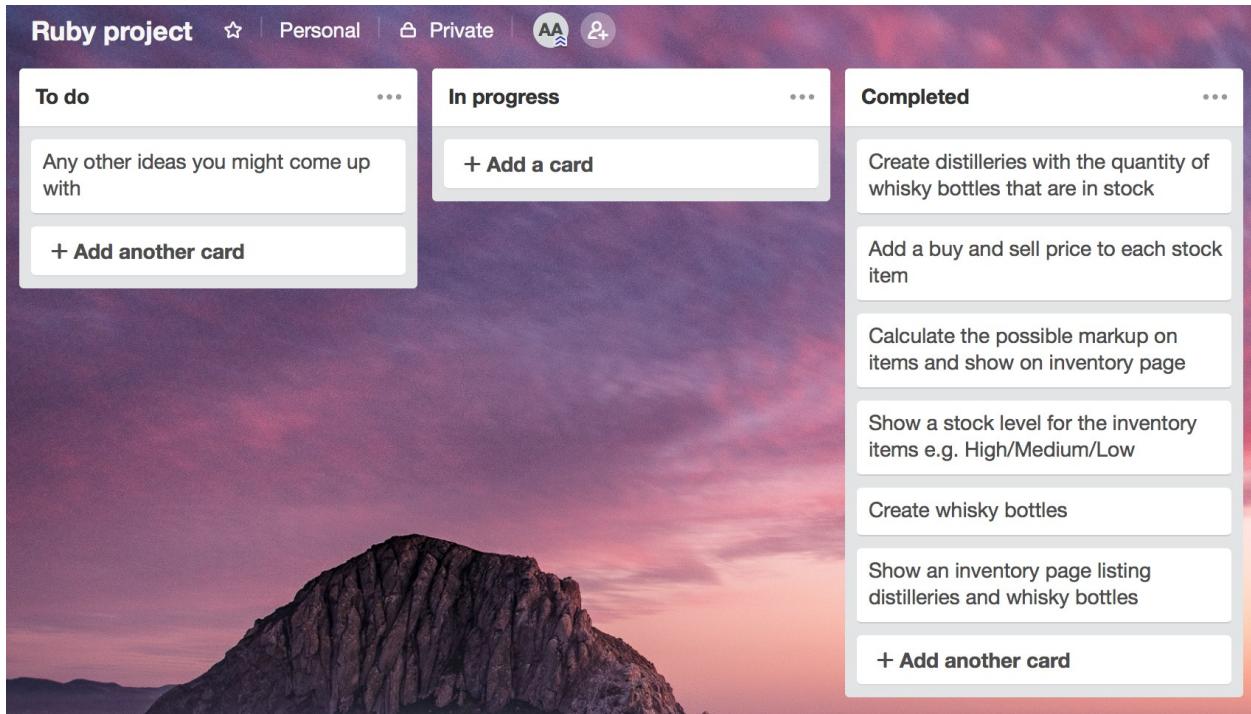
Start of project



Halfway point



Completed project



WEEK_11

Demonstrate testing in your program. Take screenshots of:

Example of test code

```
card.rb

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
```

```

require_relative('card.rb')
class CardGame

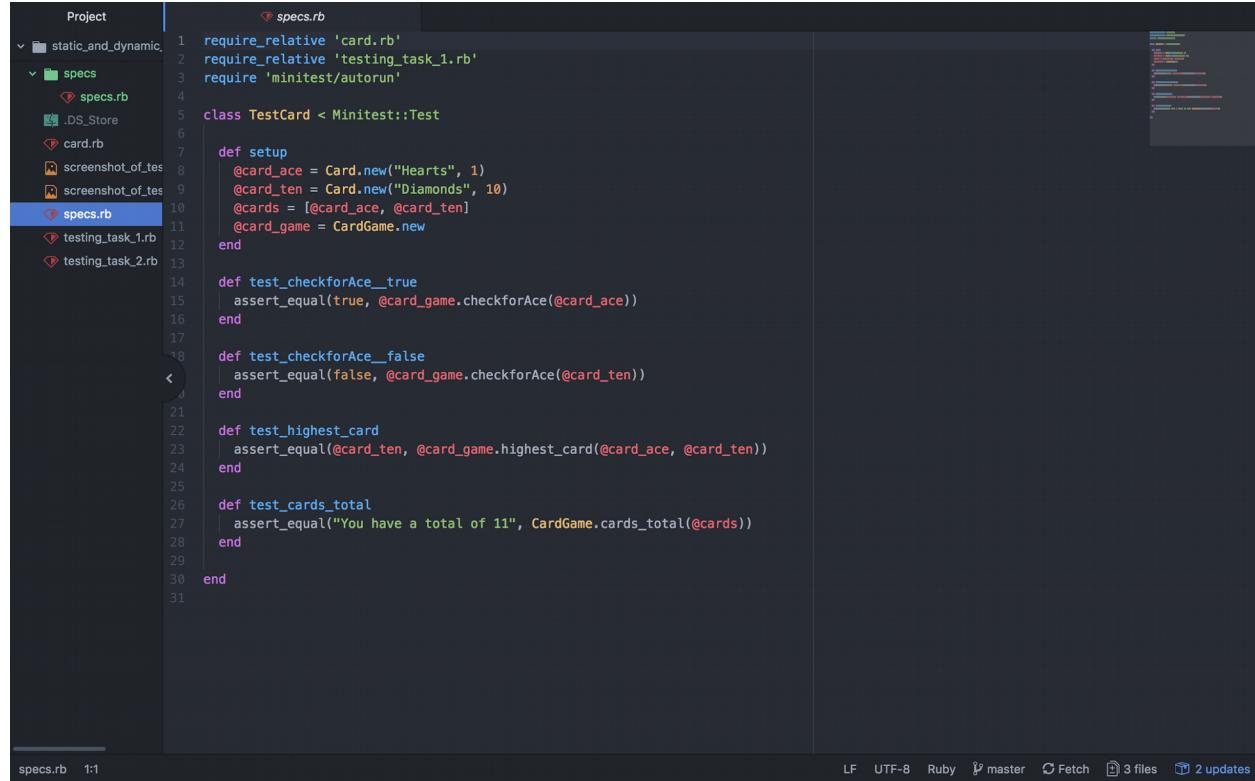
  def checkforAce(card)
    if card.value == 1 #card.value should be followed by a double equal sign
      return true
    else
      return false
    end
  end

  #dif should say def as def is the required keyword used to define a method
  #card1 and card2 must be separated with a comma
  def highest_card(card1, card2)
    if card1.value > card2.value
      return card1.name #there is no variable called 'name' in this class, it should say 'card1' instead
    else
      card2
    end
  end
end

def self.cards_total(cards)
  total #total must be set to 0 here as it will be set to null by default
  for card in cards
    total += card.value
  ##return statement should be outside of the loop after the first end keyword. There should be a space between "of" and
    return "You have a total of" + total
  end
end
#missing an end keyword here to end the class

```

Specs file



The screenshot shows a code editor interface with a dark theme. On the left is a file tree for a project named 'static_and_dynamic'. The 'specs' directory contains several files: 'spec_helper.rb', 'spec.rb', 'CardGame_spec.rb', 'testing_task_1.rb', and 'testing_task_2.rb'. The 'spec.rb' file is currently open in the editor.

```

Project
└ static_and_dynamic
  └ specs
    spec_helper.rb
    spec.rb
    CardGame_spec.rb
    screenshot_of_tes
    screenshot_of_tes
    spec.rb
    testing_task_1.rb
    testing_task_2.rb

```

The content of the 'spec.rb' file is as follows:

```

require_relative 'spec_helper'
require_relative 'testing_task_1.rb'
require 'minitest/autorun'

class TestCard < Minitest::Test

  def setup
    @card_ace = Card.new("Hearts", 1)
    @card_ten = Card.new("Diamonds", 10)
    @cards = [@card_ace, @card_ten]
    @card_game = CardGame.new
  end

  def test_checkforAce_true
    assert_equal(true, @card_game.checkforAce(@card_ace))
  end

  def test_checkforAce_false
    assert_equal(false, @card_game.checkforAce(@card_ten))
  end

  def test_highest_card
    assert_equal(@card_ten, @card_game.highest_card(@card_ace, @card_ten))
  end

  def test_cards_total
    assert_equal("You have a total of 11", CardGame.cards_total(@cards))
  end
end

```

The code editor has syntax highlighting for Ruby and includes a status bar at the bottom with file information: 'spec.rb 1:1', 'LF', 'UTF-8', 'Ruby', 'master', 'Fetch', '3 files', and '2 updates'.

The test code failing to pass

```
+ static_and_dynamic_tasks git:(master) ✘ ruby specs.rb
specs.rb:2:in `require_relative': /Users/user/codeclan_work/my_pda_folder/static_and_dynamic_tasks/testing_task_1.rb:27: syntax error, unexpected keyword_end, expecting end-of-inp
ut (SyntaxError)
    from specs.rb:2:in `<main>'
+ static_and_dynamic_tasks git:(master) ✘
```

Example of the test code once errors have been corrected

```
1  require_relative('card.rb')
2  class CardGame
3
4
5      def checkforAce(card)
6          if card.value == 1
7              return true
8          else
9              return false
10         end
11     end
12
13     def highest_card(card1, card2)
14         if card1.value > card2.value
15             return card1
16         else
17             card2
18         end
19     end
20
21     def self.cards_total(cards)
22         total = 0
23         for card in cards
24             total += card.value
25         end
26         return "You have a total of #{total}"
27     end
28   end
29
```

The test code passing

```
[→ static_and_dynamic_tasks git:(master) ✘ ruby specs.rb
Run options: --seed 62710
```

Running:

....

Finished in 0.001419s, 2819.6416 runs/s, 2819.6416 assertions/s.

4 runs, 4 assertions, 0 failures, 0 errors, 0 skips

```
→ static_and_dynamic_tasks git:(master) ✘ █
```

WEEK_12

Show an API being used within your program. Take a screenshot of:

The code that uses or implements the API

```
1 const apiBeerArray = [];
2
3 document.addEventListener('DOMContentLoaded', () => {
4   const url = 'https://api.punkapi.com/v2/beers';
5   makeRequest(url, requestComplete);
6
7   const selectBeerDropDown = document.querySelector('#beer-selector');
8   selectBeerDropDown.addEventListener('change', handleSelectBeer);
9 });
10
11 const makeRequest = function (url, callback) {
12   const request = new XMLHttpRequest();
13   request.open('GET', url);
14   request.send();
15   request.addEventListener('load', callback);
16 };
17
18 const requestComplete = function () {
19   if (this.status !== 200) return;
20   const jsonString = this.responseText;
21   const beers = JSON.parse(jsonString);
22   beers.forEach((beer) => {
23     apiBeerArray.push(beer);
24   });
25   populateBeerDropDown();
26};
```

The API being used by the program whilst running

BrewDog collection

Select a beer to view:

BrewDog collection

Beer name: Devine Rebel (w/ Mikkeller)

- ABV : 12.5%
- "Oak-aged Barley Wine."



BrewDog collection

Trashy Blonde

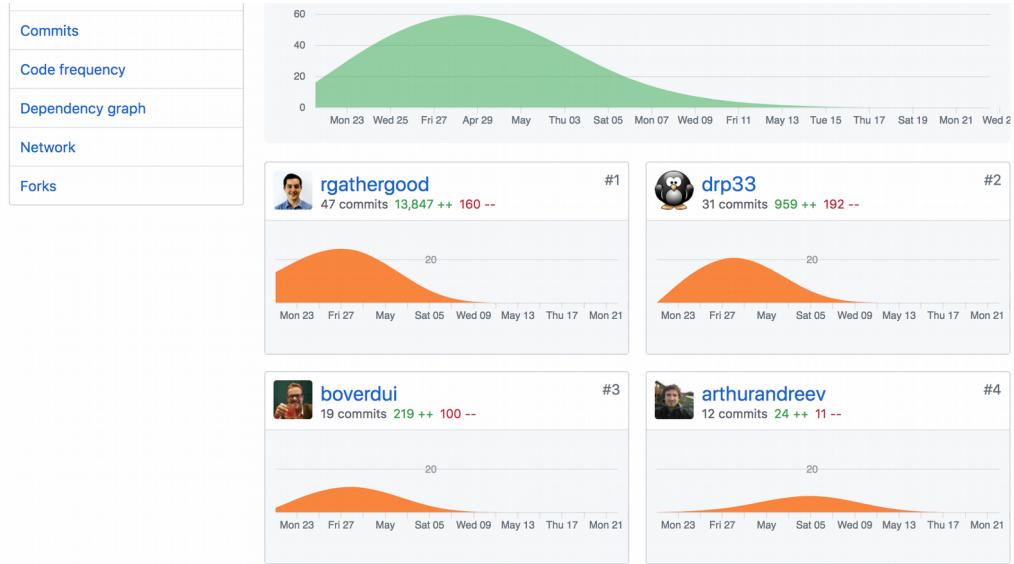


Beer name: Trashy Blonde

- ABV : 4.1%
- "You Know You Shouldn't"



Take a screenshot of contributor's page on GitHub from your group project to show the team you worked with



Take a screenshot of the project brief from your group project

Vex Yourself - Flag Quiz - Group Project

Week 14 Group Project

Educational App

The BBC are looking to improve their online offering of educational content by developing some interactive apps that display information in a fun and interesting way.

Your task is to make an MVP to put forward to them - this may only be for a small set of information, and may only showcase some of the features to be included in the final app. You might use an API to bring in content or a database to store facts. The topic of the app is your choice, but here are some suggestions you could look into:

- Interactive timeline, e.g. of the history of computer programming
- Interactive map of a historical event - e.g. World War 1, the travels of Christopher Columbus

MVP

- Display some information about a particular topic in an interesting way
- Have some user interactivity using event listeners, e.g. to move through different sections of content

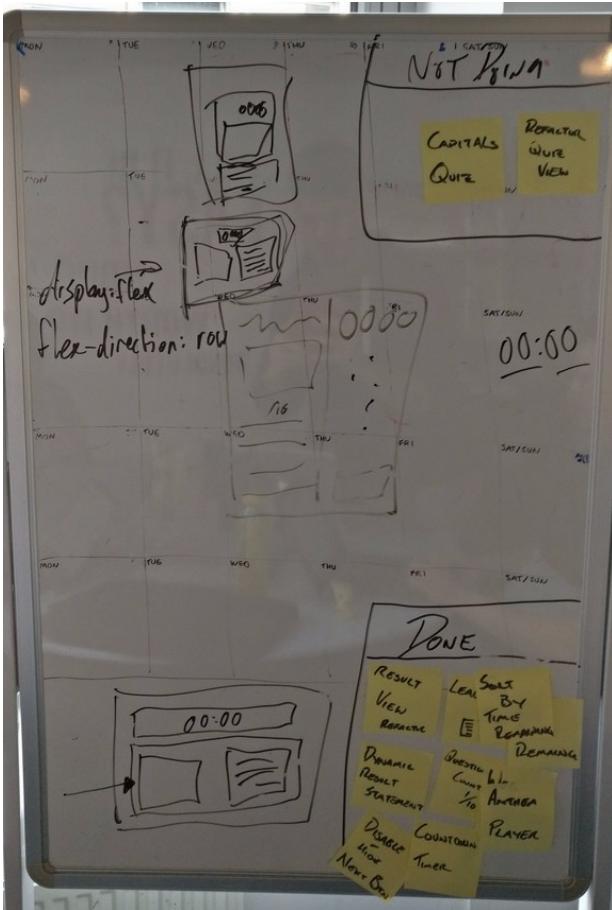
How to run

To run our app you'll have to do the following:

- Clone this repo

In your terminal run:

Provide a screenshot of the planning you completed during your group project



Write an acceptance criteria and test plan

Acceptance criteria	Expected result/output	Pass/Fail
A user is able to access all stock items in inventory	A full stock-list is displayed when link to 'Stock' is clicked	Pass
A user is able to edit a stock item in inventory	A new page is displayed which allows the user to edit the selected stock item when "Edit item" button is clicked	Pass
A user is able to delete a stock item in inventory	A selected item is deleted when the user clicks "Delete item" button	Pass
A user is able to add a new item to inventory	A new page is displayed which allows the user to add a new stock item when "Add stock" link is clicked	Pass
A user is able to add a new distillery to the list of distilleries	A new page is displayed which allows the user to add a new distillery when "Add distillery" link is clicked	Pass

Produce two system interaction diagrams(sequence and/or collaboration diagrams)

Diagram 1

Collaboration diagram of Dinosaur transfer system

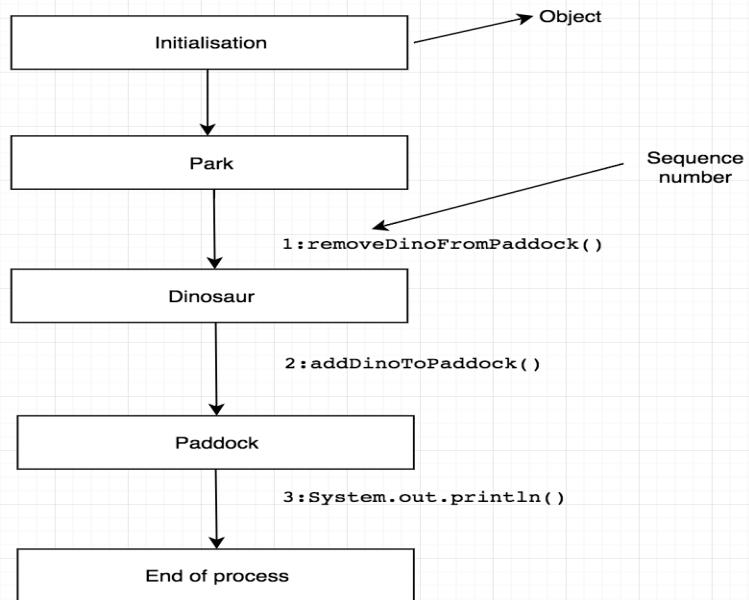
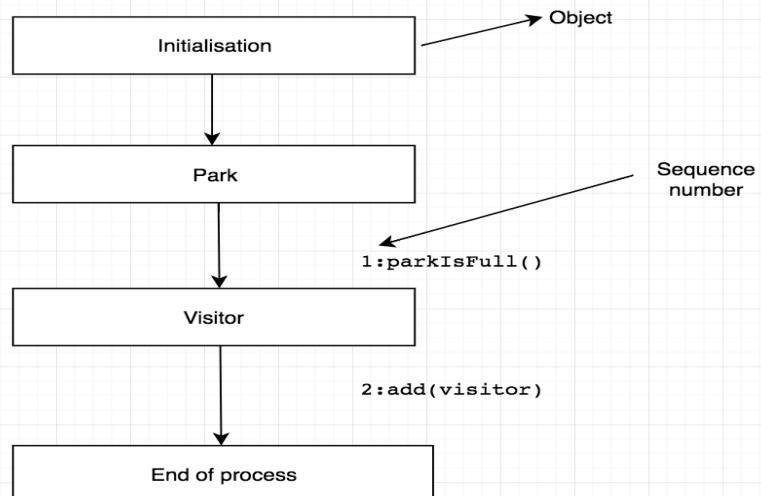


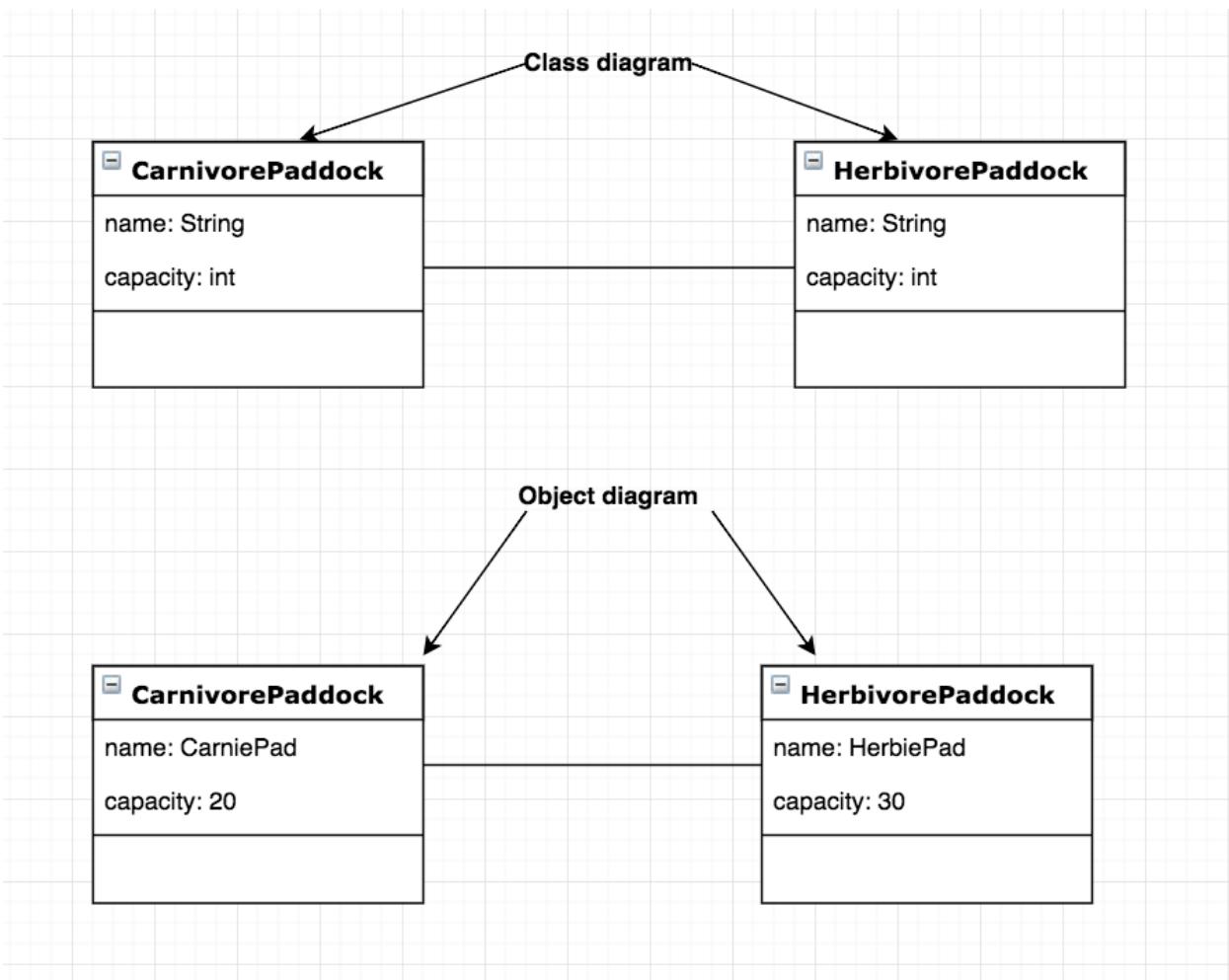
Diagram 2

Collaboration diagram of Visitor checking-in system

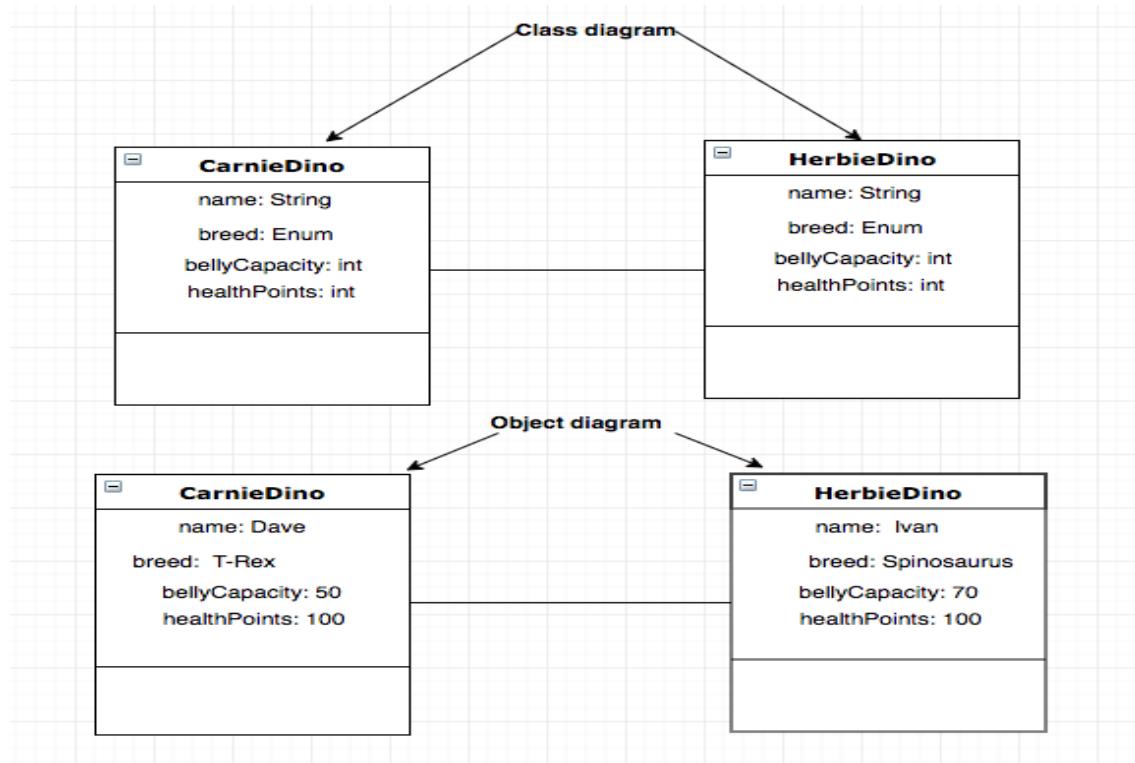


Produce two object diagrams

Object diagram 1



Object diagram 2



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

1) addDinosaurToPaddock()

```
public void addDinosaurToPaddock(Dinosaur dinosaur) {  
    // switch case statement that calls getType method on dinosaur object to access TypeOfDinosaur Enum  
    switch (dinosaur.getType()) {  
        case Herbivore:  
            herbivorePaddock.addDinoToPaddock(dinosaur);  
  
        case Carnivore:  
            carnivorePaddock.addDinoToPaddock(dinosaur);  
    }  
}
```

This method adds different types of dinosaurs (herbivore/carnivore) in separate paddocks as they can't reside in the same paddock. I chose this method because I needed to write an algorithm for checking what type of dinosaur has been passed in to my method and then add this dinosaur object to the correct paddock. I used a switch/case statement here which is passed in the getType() method that returns the type of dinosaur object passed in (herbivore or carnivore which is stored in an Enum). These are the only two types that exist in the Enum hence there are two corresponding cases for Herbivore and Carnivore. addDinoToPaddock() method adds the dinosaur object to the correct paddock depending on whether dinosaur is herbivore or carnivore.

2) transferDinosaur()

```
public void transferDinosaur(Dinosaur dinoToMove, Paddock paddockToMoveFrom, Paddock paddockToMoveTo){  
    paddockToMoveFrom.removeDinoFromPaddock(dinoToMove);  
    paddockToMoveTo.addDinoToPaddock(dinoToMove);  
    System.out.println("Dino has been transferred to a new paddock!");  
}
```

I chose this method because I needed to write an algorithm that would allow me to transfer dinosaurs between paddocks.

This method takes in three parameters: dinosaur, paddockToMoveFrom and paddockToMoveTo.

First step in this algorithm is calling removeDinoFromPaddock method on paddockToMoveFrom and passing it our dinosaur object. This will remove our dinosaur from paddock.

Second step is calling addDinoToPaddock method on paddockToMoveTo and passing it our dinosaur object. This will add our dinosaur to a new paddock.

Finally a message will be printed out to the console that will read "Dino has been transferred to a new paddock".

Produce a bug tracking report

Expected result	1st testing attempt		2nd testing attempt
The user must be able to start a new quiz by clicking on “start new quiz” button			Passed
The countdown timer must activate as soon as “play” button is clicked			Passed
The user must be able to enter their name which will be saved alongside his quiz score	Failed	An empty array that was created outside of newQuiz() function was initially set to ‘const’ which prevented it from being modified. Changed the empty array to ‘let’ which rectified this error and allowed the empty array to take in result values such as scores	Passed
The user must be able to see countdown timer in the following format- 00:00	Failed	Couldn’t get the time left to display with “:” in between minutes and seconds as was required. Added split(“：“) method to timer.display which separated time strings with “:” that worked perfectly	Passed