

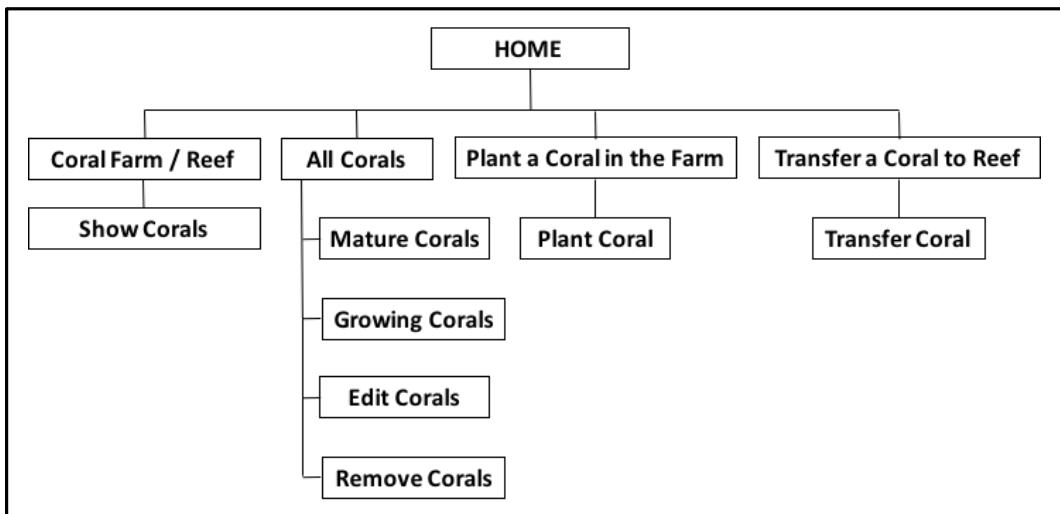
PDA: Software Development Level 8

Student: David Sanchez Rodriguez

EVIDENCE: UNIT P

Ref: P5

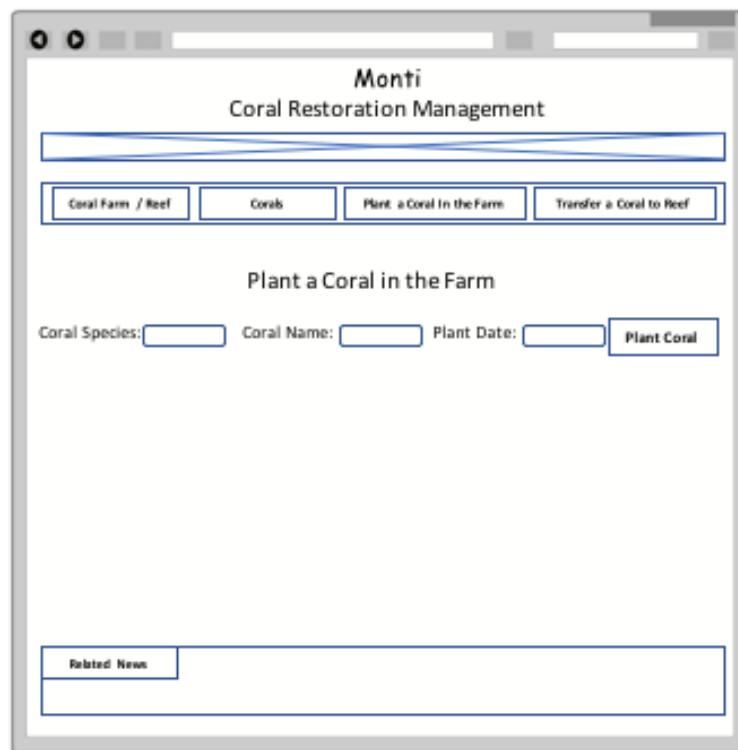
- Create a user sitemap



Ref: P6

- Produce two wireframe designs





Ref: P10

- Example of pseudocode for a function

```
❖ pda.rb

82 # Unit: P - P 10
83 #Pseudocode for a function
84 it("should show the name of the corals in the reef", function(){
85   #test the coral name
86   #test the reef name
87   #verify the save() method is working properly in the Database
88   #verify the update() method is working properly in the Database
89   #the coral-reef relationship is one to many, 'one reef can have many corals'
90   #verify each coral is assigned to one and only one reef
91   #the corals should be listed by transfer date
92 })
```

Ref: P13

- The user inputting something into your program (Updating data)

- 1) The user wants to change the plant date for the coral at the bottom of the table (*Acropora cervicornus*) from “2012 – 11 – 14” to “2012 – 11 – 16” by clicking the button Edit Coral Information

The screenshot shows a web browser window titled "MONTI" with the URL "localhost:4567/coral". The page has a header "Coral Restoration Management" and a sub-header "All the Corals". Below the header is a navigation bar with four buttons: "Coral Farm / Reef", "All Corals" (which is highlighted in red), "Plant a Coral in the Farm", and "Transfer a Coral to Reef". The main content area is titled "All the Corals" and contains a table of coral data. The table has columns: Species, Name, Plant date, Coral status, Location, Edit, and Remove Coral. The data is as follows:

Species	Name	Plant date	Coral status	Location	Edit	Remove Coral
Montipora	Green Montipora	2012-11-14	Mature	Tubbataha	Edit Coral Information	Remove Coral
Euphyllia	Blue Hammer Head	2012-11-14	Mature	Koh Ma	Edit Coral Information	Remove Coral
Sarconphyton	Todstool	2012-11-14	Mature	Molasses	Edit Coral Information	Remove Coral
Acropora cervicornus	Staghorn Coral	2012-11-16	Mature	Ningaloo	Edit Coral Information	Remove Coral

At the bottom of the page, there is a "Related news" section featuring a large image of a coral reef.

- 2) Once the Edit Coral Information is clicked the user is sent to the website view for editing coral information

The screenshot shows a web browser window titled "MONTI" with the URL "localhost:4567/coral/9". The page has a header "Coral Restoration Management" and a sub-header "Edit Coral Information". Below the header is a navigation bar with four buttons: "Coral Farm / Reef", "All Corals", "Plant a Coral in the Farm", and "Transfer a Coral to Reef". The main content area is titled "Edit Coral Information" and contains a form with the following fields:

Coral Species: Coral Name: Plant date:

At the bottom of the page, there is a "Related news" section featuring a large image of a coral reef.

- 3) Once the information is edited, the coral at the bottom of the table appears with the new plant date.

The screenshot shows a web browser window titled "MONTI" with the URL "localhost:4567/coral". The page has a header "Coral Restoration Management" and a sub-header "All the Corals". Below the header is a table with the following data:

Species	Name	Plant date	Coral status	Location	Edit	Remove Coral
Montipora	Green Montipora	2012-11-14	Mature	Tubbataha	Edit Coral Information	Remove Coral
Euphyllia	Blue Hammer Head	2012-11-14	Mature	Koh Ma	Edit Coral Information	Remove Coral
Sarcophyton	Todstool	2012-11-14	Mature	Molasses	Edit Coral Information	Remove Coral
Acropora cervicornis	Staghorn Coral	2012-11-14	Mature	Ningaloo	Edit Coral Information	Remove Coral

Below the table is a large image of a coral reef with various coral species. At the bottom of the page, there is a "Related news" section.

Ref: P14

- Show an interaction with data persistence (Adding new data).

- 1) The user selecting the Coral Species using the drop-down menu.

The screenshot shows a web browser window titled "MONTI" with the URL "localhost:4567/coral/new". The page has a header "Coral Restoration Management" and a sub-header "Plant a Coral in the Farm". On the left, there is a dropdown menu for "Coral Species" with the following options:

- Sarcophyton Sp.
- Lobophytum Sp.
- Euphyllia Sp.
- Catalaphyllia Sp.
- Caulastrea Sp.
- Tracyphyllia Sp.

On the right, there are input fields for "Coral Name" (Common Toadstool Coral), "Plant Date" (02/04/2018), and a "Plant Coral" button. The background features an image of a coral reef with fish.

2) The user selecting the Coral Name using the drop-down menu.

The screenshot shows a web browser window for the MONTI Coral Restoration Management system. The title bar says "MONTI". The URL bar shows "localhost:4567/coral/new". Below the title bar, there's a message "Google Chrome isn't your default browser." and a "Set as default" button. The main content area has a blue header with tabs: "Coral Farm / Reef" (highlighted), "All Corals", "Plant a Coral in the Farm" (highlighted), and "Transfer a Coral to Reef". The background is a photograph of a coral reef with fish. In the center, there's a form with fields: "Coral Species: Sarconphyton Sp.", "Coral Name: Common Toadstool Coral" (with a dropdown arrow), and "Plant Date: 02/04/2018". A "Plant Coral" button is next to the date field. A dropdown menu is open over the "Coral Name" field, listing: Devil's Hand Coral, Hammer Coral, Elegance Coral, Trumpet Coral, and Open Brain Coral. At the bottom of the page, there's a "Related news" section.

3) The user selecting the Date the coral was planted.

The screenshot shows the same MONTI Coral Restoration Management system as the previous one, but the "Coral Name" field now contains "Common Toadstool Coral". The "Plant Date" field shows "02/04/2018". A calendar is open over the "Plant Date" field, displaying April 2018. The calendar grid shows days from Monday to Sunday, with the 2nd highlighted in blue. Other dates are shown in black. Navigation arrows for the month are at the top right of the calendar. The background image of the coral reef is visible behind the calendar.

- 4) Before adding the coral, the “All the Corals” have 5 corals in it.

MONTI
Coral Restoration Management

All the Corals

Species	Name	Plant date	Coral status	Location	Edit	Remove Coral
Acropora cervicornis	Staghorn Coral	2012-11-14	Mature	Ningaloo	Edit Coral Information	Remove Coral
Montipora	Green Montipora	2012-11-14	Mature	Tubbataha	Edit Coral Information	Remove Coral
Euphyllia	Blue Hammer Head	2012-11-14	Mature	Koh Ma	Edit Coral Information	Remove Coral
Sarconphyton	Todstool	2012-11-14	Mature	Molasses	Edit Coral Information	Remove Coral
Another Sarconphyton	Todstool	2012-11-15	Mature	Molasses	Edit Coral Information	Remove Coral

Related news

- 5) After adding the coral, the “All the Corals” have 6 corals in it.

MONTI
Coral Restoration Management

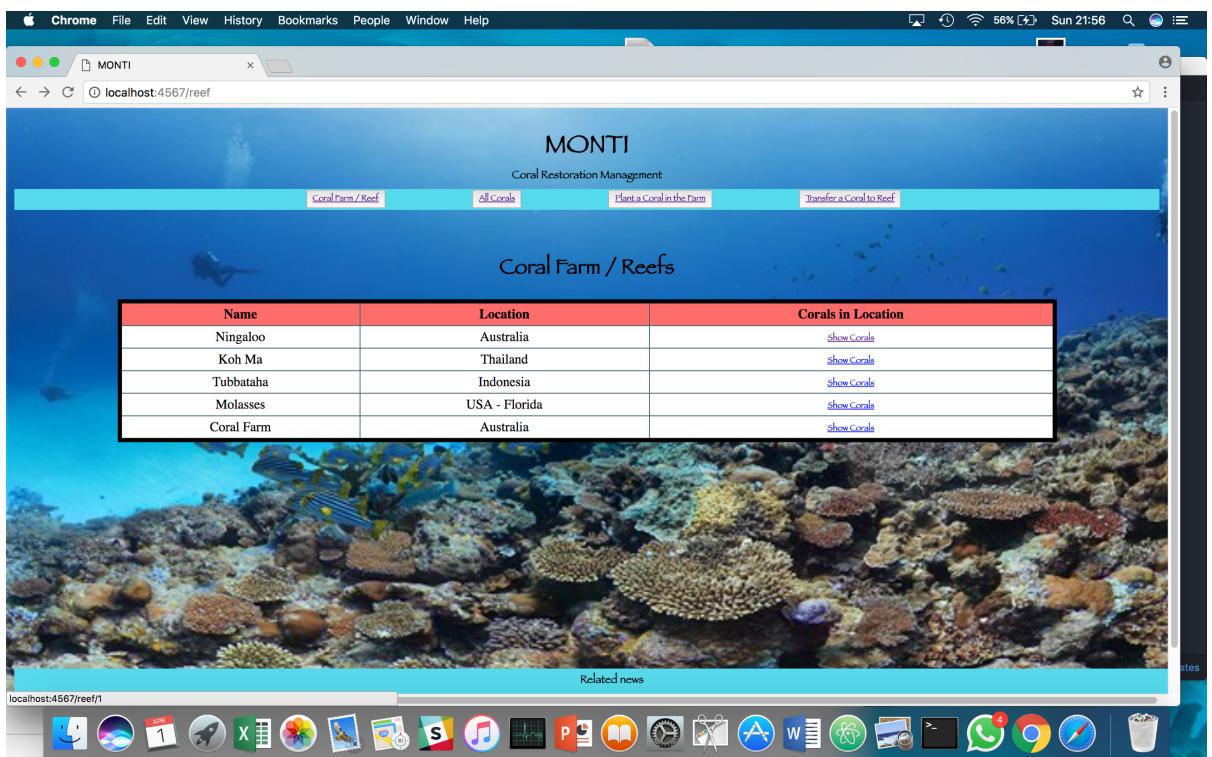
All the Corals

Species	Name	Plant date	Coral status	Location	Edit	Remove Coral
Acropora cervicornis	Staghorn Coral	2012-11-14	Mature	Ningaloo	Edit Coral Information	Remove Coral
Montipora	Green Montipora	2012-11-14	Mature	Tubbataha	Edit Coral Information	Remove Coral
Euphyllia	Blue Hammer Head	2012-11-14	Mature	Koh Ma	Edit Coral Information	Remove Coral
Sarconphyton	Todstool	2012-11-14	Mature	Molasses	Edit Coral Information	Remove Coral
Another Sarconphyton	Todstool	2012-11-15	Mature	Molasses	Edit Coral Information	Remove Coral
Sarconphyton Sp.	Common Toadstool Coral	2018-04-02	Growing	Coral Farm	Edit Coral Information	Remove Coral

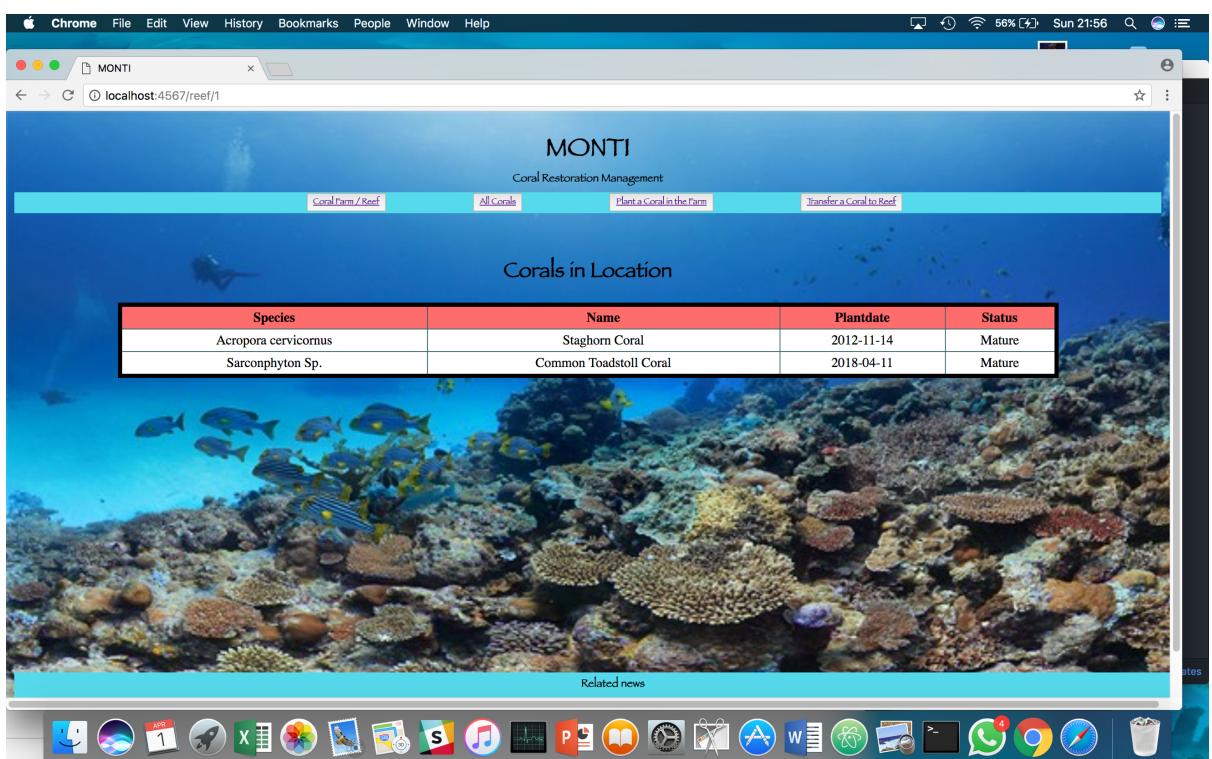
Related news

Ref: P15

- Show the correct output of results and feedback to user:
 - 1) The user requesting information or an action to be performed:
The request is to see all the corals in Ningaloo, Australia

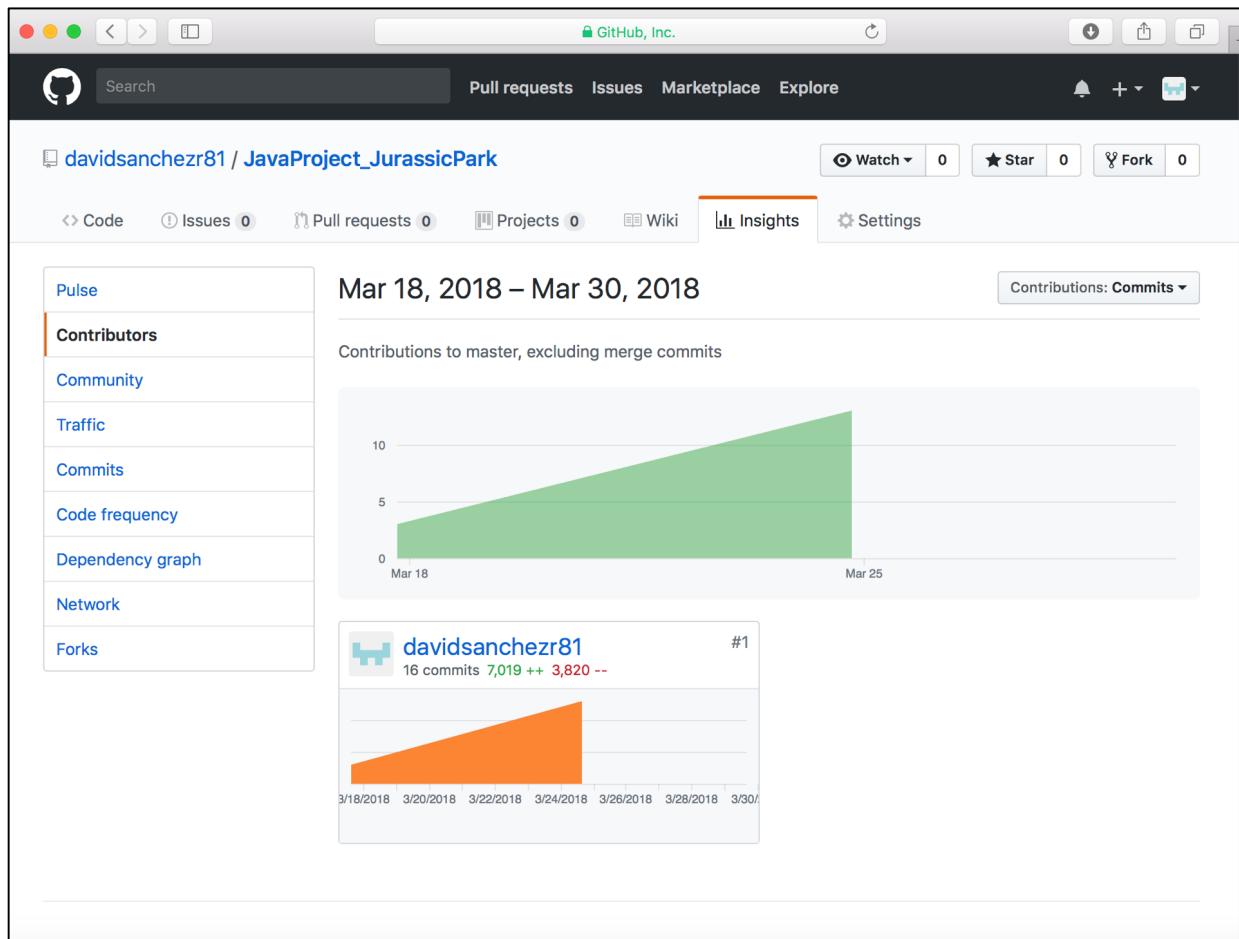


- 2) The user request being processed correctly and demonstrated in the program:
The result of the request the 2 corals currently present in Ningaloo, Australia.



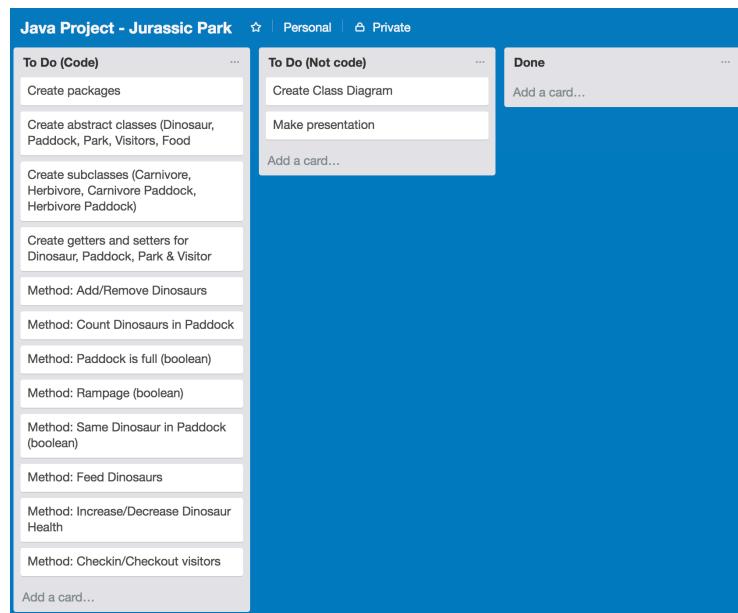
Ref: P11

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

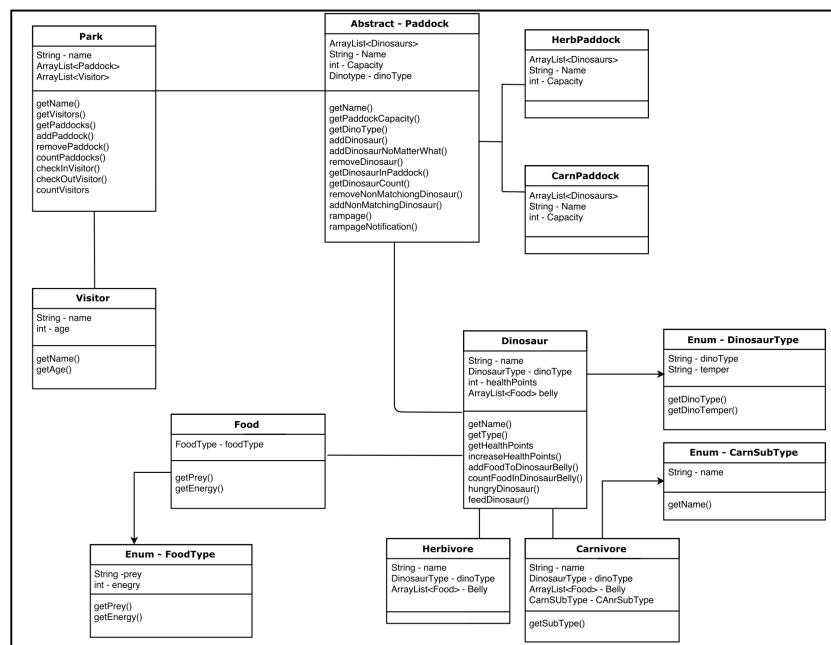


Ref: P12

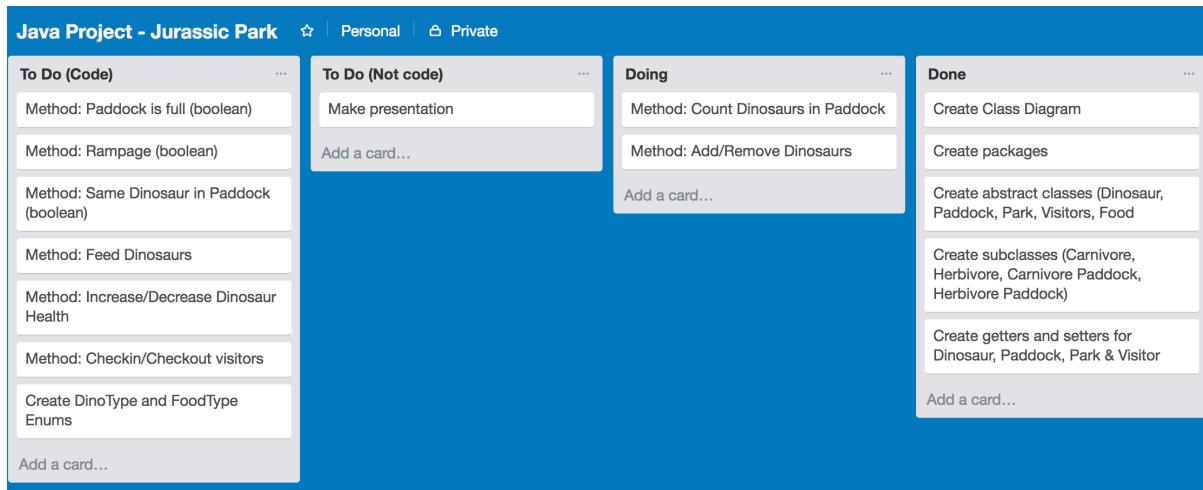
- Take screenshots or photos of your planning and the different stages of development to show changes.
- Starting the Java Project using Trello for planning



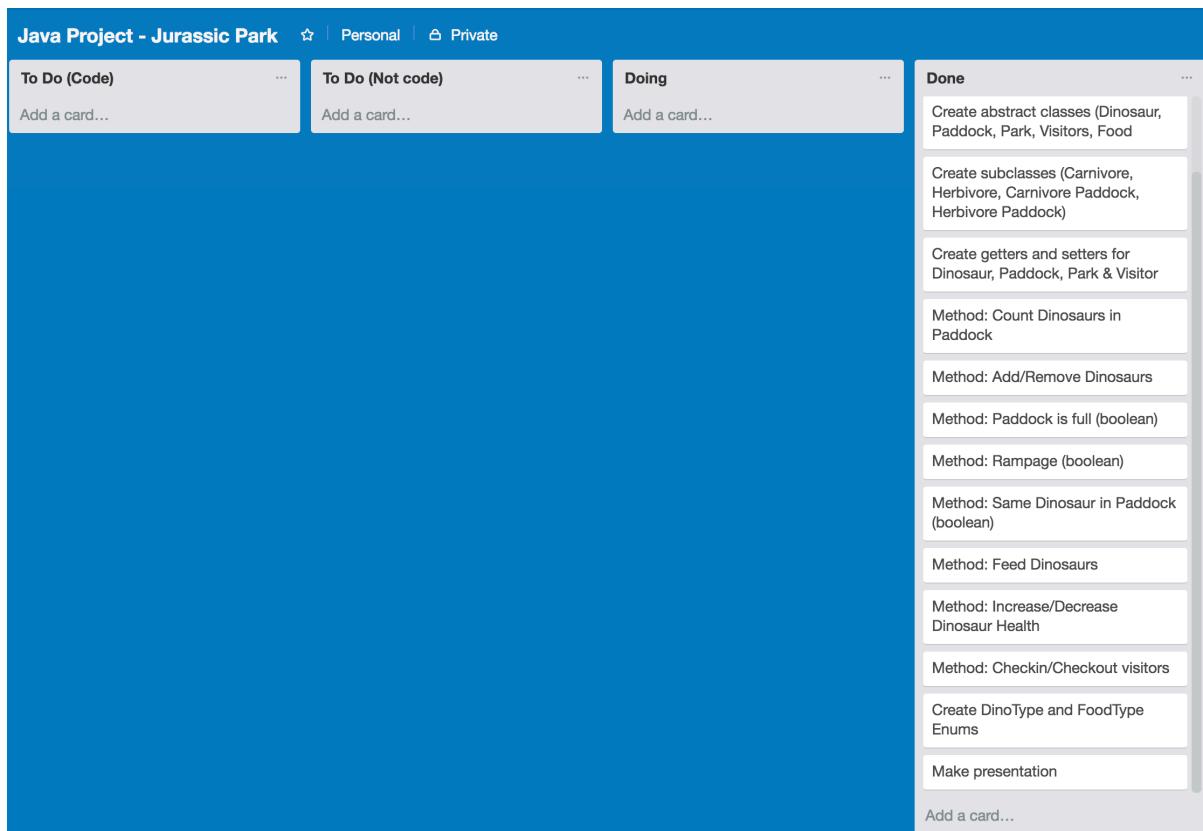
- Class Diagram of Jurassic Park Project



- 50% of the Java Project Completed



- 100% of the Java Project Completed



- PowerPoint presentation



- Final Java project layout

```

JavaProject > src > main > java > Paddocks > Paddock
Project   ~/codeclan_work/week_01
  .gradle
  .idea
  gradle
  out
  src
    main
      java
        Dinosaur
          Carnivore
          CarnSubType
          Dinosaur
          DinosaurType
          Herbiore
        Paddocks
          CarnPaddock
          Food
          FoodType
          HerbPaddock
          Paddock
        Park
          Park
          Visitor
        resources
      test
        java
          CarnivoreTest
          CarnPaddockTest
          FoodTest
          HerbivoreTest
          HerbPaddockTest
          ParkTest
          VisitorTest
        resources
      build.gradle
      gradlew
  Dinosaur.java
  Paddock.java
  Dinosaur
  Paddock
  Carnivore
  CarnSubType
  Dinosaur
  DinosaurType
  Herbiore
  Paddock
  Park
  Visitor
  resources
  build.gradle
  gradlew
  Event Log
  75:58 LF: UTF-8 Git: master
  
```

Dinosaur.java

```

import Paddocks.Food;
import java.util.ArrayList;

public abstract class Dinosaur {
    private String name;
    private DinosaurType dinoType;
    private ArrayList<Food> belly;
    private int healthPoints;

    public Dinosaur(String name, DinosaurType dinoType, int healthPoint) {
        this.name = name;
        this.dinoType = dinoType;
        this.healthPoints = healthPoints;
        this.belly = new ArrayList<Food>();
    }

    public String sayDinoType(){return "I am a Dinosaur";}

    public String getName() { return this.name; }

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

    public String getType(){ return this.dinoType.getType(); }

    public int getHealthPoints(){ return this.healthPoints; }

    public void setHealthPoints(int healthPoints){ this.healthPoints = healthPoints; }

    public void increaseHealthPoints(Food food){ this.healthPoints += food.getHealthPoints(); }

    public void addFoodToDinosaurBelly(Food food){ this.belly.add(food); }

    public int countFoodInDinosaursBelly(){ return this.belly.size(); }

    public boolean hungryDinosaur() { return this.healthPoints < 5; }

    public void feedDinosaur(Food food) {
        if(hungryDinosaur()){
            addFoodToDinosaurBelly(food);
        }
    }
}
  
```

Paddock.java

```

public int getDinosaursCount() {
    return this.dinosaurs.size();
}

public boolean isPaddockFull() {
    return this.dinosaurs.size() == this.capacity;
}

// TRANSFER HERBIVORE /////////////////////////////////
public ArrayList<Dinosaur> removeNonMatchingDinosaur() {
    ArrayList<Dinosaur> nonMatchingDinosaur = new ArrayList<>();

    for (Dinosaur dinosaur : dinosaurs) {
        if (!dinosaur.getType().equals(getDinoType())) {
            nonMatchingDinosaur.add(dinosaur);
        }
    }
    for (Dinosaur dinosaur : nonMatchingDinosaur) {
        removedDinosaur(dinosaur);
    }
    return nonMatchingDinosaur;
}

public void addNonMatchingDinosaur(ArrayList<Dinosaur> nonMatchingDinosaur) {
    for (Dinosaur dinosaur : nonMatchingDinosaur) {
        if (dinosaur.getType().equals(getDinoType())){
            addDinosaur(dinosaur);
        }
    }
}

// RAMPAGE /////////////////////////////////
public boolean rampage() { return this.dinosaurs.size() > this.capacity; }

public String rampageNotification(){
    if(rampage()){
        return "Rampage Situation! Run for your Life";
    }
    return "All good, no danger so far, keep spending money";
}
  
```

Ref: P18

Demonstrate testing in your program. Take screenshots of:

- Example of test code



The screenshot shows a code editor with two tabs open. The left tab is named 'card.rb' and the right tab is named 'dynamic_testing_spec.rb'. The code in 'card.rb' defines a class 'Card' with methods like 'new', 'rank', and 'suit'. It also defines a class 'CardGame' with methods like 'highest_card', 'cards_total', and 'check_for_ace'. The code in 'dynamic_testing_spec.rb' is a test suite using MiniTest::autorun. It includes setup code, four test methods ('test_check_for_ace_True', 'test_check_for_ace_False', 'test_highest_card', 'test_cards_total'), and assertions for each test.

```
require_relative '../card'
require_relative '../dynamic_testing.rb'
require 'minitest/autorun'

class TestCard < MiniTest::Test

  def setup
    @card1 = Card.new('One', 1)
    @card2 = Card.new('Two', 2)
    @card_game = CardGame.new
    @cards = [@card1, @card2]
  end

  def test_check_for_ace_True
    assert_equal(false, @card_game.check_for_ace(@card1))
  end

  def test_check_for_ace_False
    assert_equal(true, @card_game.check_for_ace(@card2))
  end

  def test_highest_card
    assert_equal(1, @card_game.highest_card(@card2, @card1))
  end

  def test_cards_total
    assert_equal("You have a total of 2", CardGame.cards_total(@cards))
  end
end
```

- The test code failing to pass

```

card.rb | dyn | Testing_exercises — rocco@Davids-MBP — ..ing_exercises — -zsh —
require_relative '../card'
require_relative '../dynamic_testing'
require 'minitest/autorun'

class TestCard < MiniTest::Test

  def setup
    @card1 = Card.new('One', 1)
    @card2 = Card.new('Two', 2)
    @card_game = CardGame.new
    @cards = [@card1, @card2]
  end

  def test_check_for_ace__True
    assert_equal(false, @card_game)
  end

  def test_check_for_ace__False
    assert_equal(true, @card_game)
  end

  def test_highest_card
    assert_equal(1, @card_game.highest_card(@card2, @card1))
  end

  def test_cards_total
    assert_equal("You have a total of 2", CardGame.cards_total(@cards))
  end
end

```

Testing_exercises git:(master) ✘ ruby specs/dynamic_testing_spec.rb

Run options: --seed 43355

Running:

FFFFF

Finished in 0.001179s, 3392.7056 runs/s, 3392.7056 assertions/s.

1) Failure:
TestCard#test_check_for_ace__False [specs/dynamic_testing_spec.rb:20]:
Expected: true
Actual: false

2) Failure:
TestCard#test_check_for_ace__True [specs/dynamic_testing_spec.rb:16]:
Expected: false
Actual: true

3) Failure:
TestCard#test_highest_card [specs/dynamic_testing_spec.rb:24]:
Expected: 1
Actual: 2

4) Failure:
TestCard#test_cards_total [specs/dynamic_testing_spec.rb:28]:
Expected: "You have a total of 2"
Actual: "You have a total of 3"

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

- Example of the test code once errors have been corrected

```

↳ card.rb | ↳ dynamic_testing_spec.rb
require_relative '../card'
require_relative '../dynamic_testing.rb'
require 'minitest/autorun'

class TestCard < MiniTest::Test

  def setup
    @card1 = Card.new('One', 1)
    @card2 = Card.new('Two', 2)
    @card_game = CardGame.new
    @cards = [@card1, @card2]
  end

  def test_check_for_ace__True
    assert_equal(true, @card_game.check_for_ace(@card1))
  end

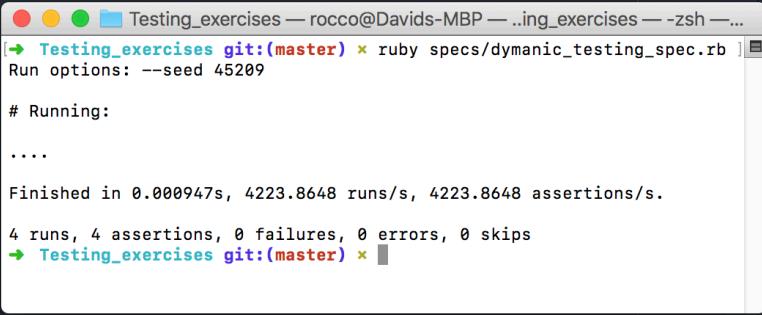
  def test_check_for_ace__False
    assert_equal(false, @card_game.check_for_ace(@card2))
  end

  def test_highest_card
    assert_equal(2, @card_game.highest_card(@card2, @card1))
  end

  def test_cards_total
    assert_equal("You have a total of 3", CardGame.cards_total(@cards))
  end
end

```

- The test code passing



The screenshot shows a terminal window with two tabs: 'card.rb' and 'dymanic_testing_spec.rb'. The 'dymanic_testing_spec.rb' tab is active, displaying Ruby test code. The terminal output shows the test results:

```

Testing_exercises git:(master) ✘ ruby specs/dymanic_testing_spec.rb
Run options: --seed 45209

# Running:
.....
Finished in 0.000947s, 4223.8648 runs/s, 4223.8648 assertions/s.

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

```

The test code in 'dymanic_testing_spec.rb' includes methods for setup, checking for ace, finding the highest card, and calculating the total cards.

```

require_relative '../card'
require_relative '../dynamic_testing.rb'
require 'minitest/autorun'

class TestCard < MiniTest::Test

  def setup
    @card1 = Card.new('One', 1)
    @card2 = Card.new('Two', 2)
    @card_game = CardGame.new
    @cards = [@card1, @card2]
  end

  def test_check_for_ace_True
    assert_equal(true, @card_game.check_for_ace(@card1))
  end

  def test_check_for_ace_False
    assert_equal(false, @card_game.check_for_ace(@card2))
  end

  def test_highest_card
    assert_equal(2, @card_game.highest_card(@card2, @card1))
  end

  def test_cards_total
    assert_equal("You have a total of 3", CardGame.cards_total(@cards))
  end
end

```

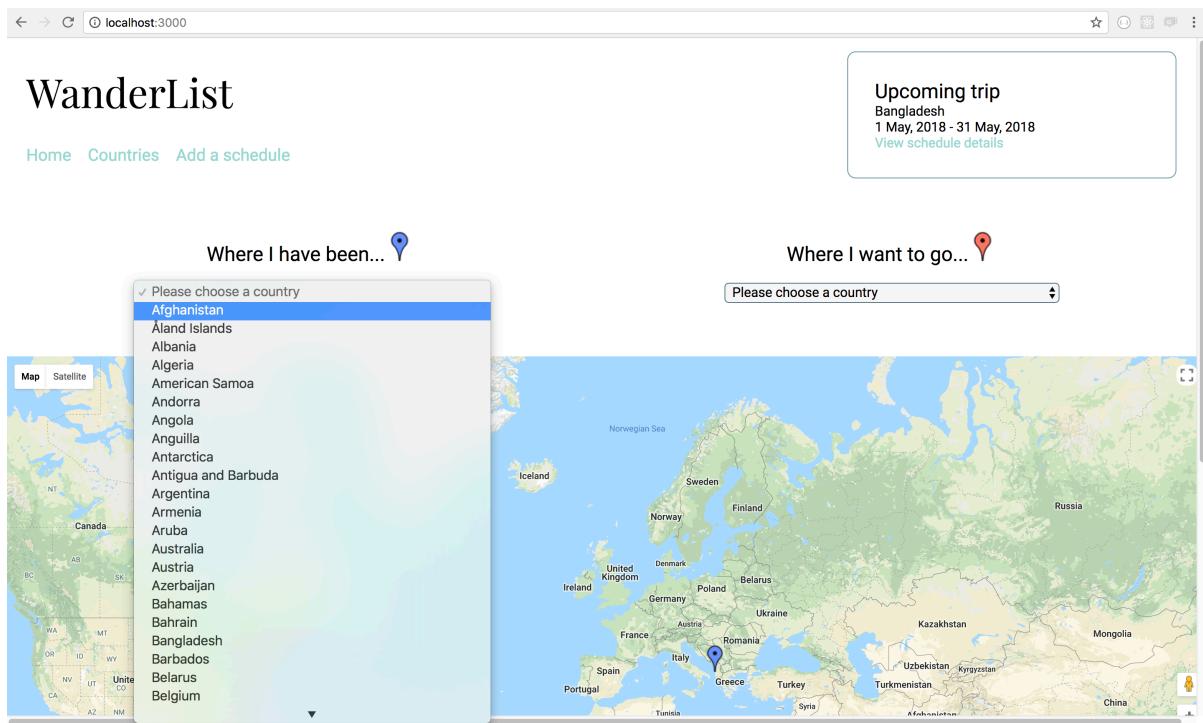
Ref: P16

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

- The code that uses or implements the API

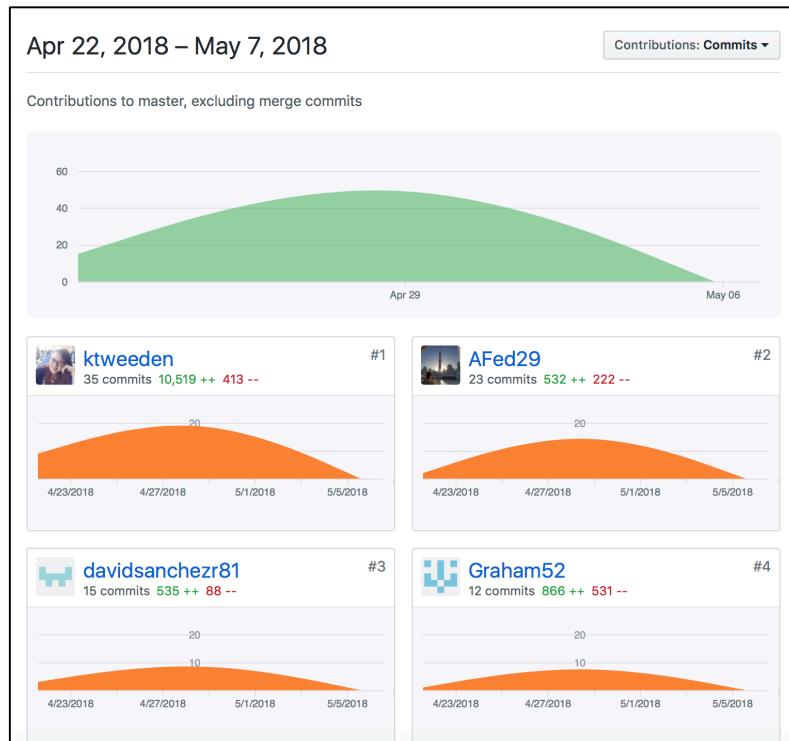
```
1 const Request = require('../helpers/request.js');
2
3 const Countries = function() {
4     this.allCountries = [];
5     this.visitedCountries = [];
6     this.toVisitCountries = [];
7     this.url = 'https://restcountries.eu/rest/v2/all';
8 }
9
10 Countries.prototype.getData = function(onComplete) {
11     const request = new Request(this.url);
12     request.get((response) => {
13         this.allCountries = response;
14         onComplete(response);
15     });
16 };
17
18 Countries.prototype.findIfCountryAlreadyInVisited = function (inputCountry) {
19     return this.visitedCountries.some((country) => {
20         return country.name === inputCountry.name;
21     });
22 };
23
24 Countries.prototype.findIfCountryAlreadyInToVisit = function (inputCountry) {
25     return this.toVisitCountries.some((country) => {
26         return country.name === inputCountry.name;
27     });
28 };
29
30 module.exports = Countries;
```

- The API being used by the program whilst running



Ref: P1

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



Ref: P2

Take a screenshot of the project brief from your group project

The screenshot shows a README.md file for a 'Travel Dashboard' project. The file content includes:

Travel Dashboard

Travel enthusiasts want to be able to track the countries they have visited and where they want to travel to in the future. Use an existing API, or create a new one, to display information about countries, travel plans, and other relevant information.

MVP

- Display countries for users to select from
- Allow users to add countries to a "countries I have been to" list
- Allow users to add countries to a "countries I would like to visit" list

Possible extensions

- Display both lists on a map
- Display currency and languages for a selected country
- Allow users to create a schedule for an upcoming trip

A project made in week 14 of [CodeClan](#) by Alex, David, Graham and Kate.

Ref: P3

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

The screenshot shows a GitHub project board for the 'travel_dashboard' repository. The board has three columns: 'To Do', 'Doing', and 'Done'. Each column contains several cards representing tasks or issues. The 'To Do' column has 14 items, the 'Doing' column has 1 item, and the 'Done' column has 26 items. The 'Doing' card is highlighted with a red border. The 'Done' column has a '+ Add column' button. The top navigation bar shows the repository name 'ktweedden / travel_dashboard' and various GitHub navigation links.

Ref: P4

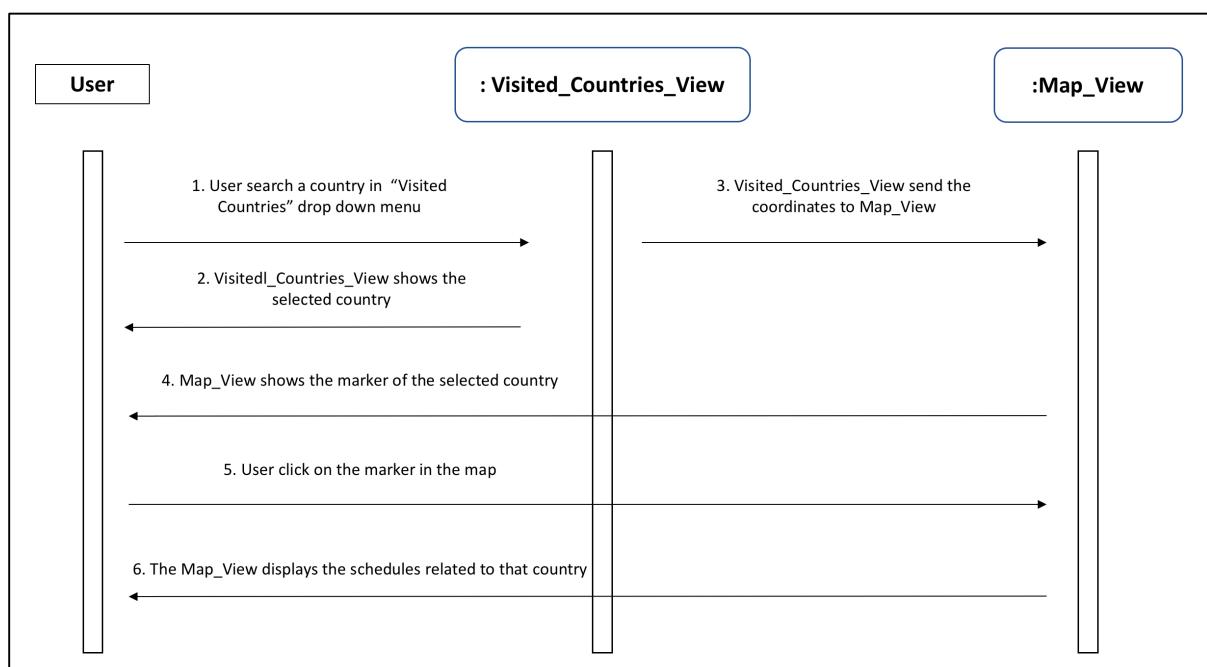
Write an acceptance criteria and test plan.

Acceptance Criteria	Expected Result / Output	Pass / Fail
The user is able to see the countries he previously visited	The list of the countries visited by the user is displayed	Pass
The user is able to see the marker in the map over every country he visited	The markers appears in the map once the user selects a country from the dropdown menu	Pass
The user is able to see the schedules on "trip date order"	The schedules are displayed on "trip date order"	Pass
The user is able to see photos/notes from the schedule	The user should be able to see photos and notes from the previous trips	Fail

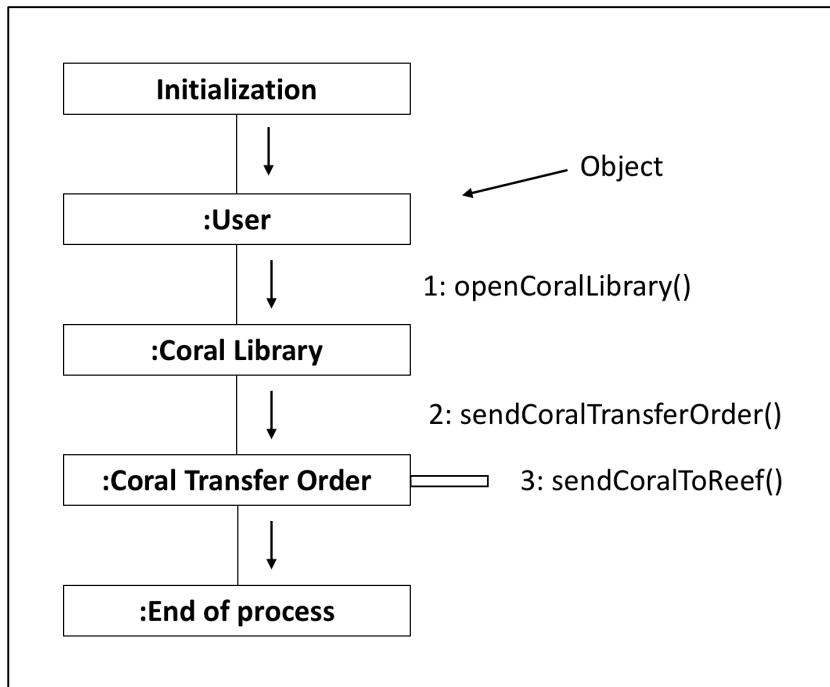
Ref: P7

Produce two system interaction diagrams:

- Sequence diagram

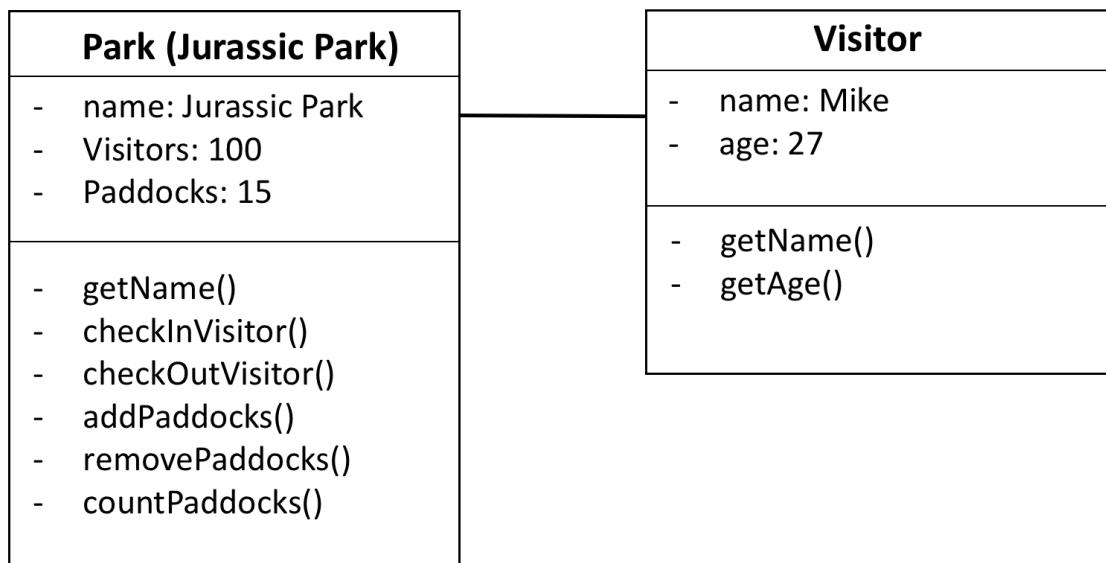


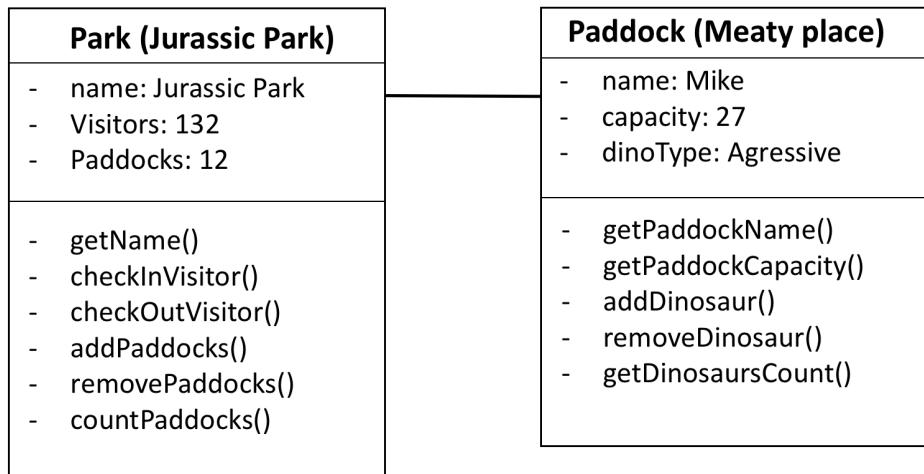
- Collaboration diagram



Ref: P8

Produce two object diagrams.





Ref: P9

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) feedDinosaur()

The reason this method was chosen is because the dinosaurs needed to be fed. However, this method is composed by 3 different methods (hungryDinosaur(), addFoodToDinosaurBelly(), increaseHealthPoints()).

The first method *hungryDinosaur()* would check if the animal is hungry giving a true or false result based on the current health level of the dinosaur. Therefore, if the animal is hungry the second method *addFoodToDinosaurBelly()* would run and the number of units of food eaten would be added to the belly. Finally, the method *increaseHealthPoints()* would increase the health level of the animal. Each food eaten would have a different health value.

```

public void feedDinosaur(Food food) {
    if(hungryDinosaur()){
        addFoodToDinosaurBelly(food);
        increaseHealthPoints(food);
    }
}

```

2) removeNonMatchingDinosaur()

The reason this method was chosen is because it was necessary to remove those dinosaurs of a different type from the paddock under evaluation.

First of all, an array for those dinosaurs to be removed from the paddock is created, *nonMatchingDinosaur*.

The first for loop would take those different dinosaurs from the paddock and would add them to the array previously created.

The second for loop would remove form the paddock the dinosaurs added to the array.

This method will return the array.

```
public ArrayList<Dinosaur> removeNonMatchingDinosaur() {  
    ArrayList<Dinosaur> nonMatchingDinosaur = new ArrayList<>();  
  
    for (Dinosaur dinosaur : dinosaurs) {  
        if (!dinosaur.getType().equals(getDinoType())) {  
            nonMatchingDinosaur.add(dinosaur);  
        }  
    }  
    for (Dinosaur dinosaur : nonMatchingDinosaur){  
        removeDinosaur(dinosaur);  
    }  
    return nonMatchingDinosaur;  
}
```

Ref: P17

Produce a bug tracking report

Expected result	1st Testing Attempt		2nd Testing Attempt
The user must be able to see all the countries in the list			Passed
The schedules should be shown based in trip date			Passed
The marker for "Visited" countries should have a different color than the "To visit" countries	Failed	In the "map_wrapper.js" file two different icons were created and linked to google icon .png images (Blue Marker for Visited countries ; Red Marker for To Visit countries)	Passed
The schedules should appear in a different page	Failed	One "bundle.js" file for each new page was created	Passed
Schedules cannot overlap	Failed	"If statements" added to avoid adding new schedules in dates that other schedules already exist	Passed