

**PDA: Software Development
Level 8
Student Evidence Checklist**

Full name	Christopher Murphy
Cohort	G4

The evidence required can be taken from your assignments, homework that you have completed on your own or by creating a specific example for the PDA.

	Unit	Ref.	Evidence	Done
Week 2	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	
			<pre>const RecordCollector = function(name,cash){ this.name = name; this.collection = [] this.cash = cash; }</pre>	
			<pre>RecordCollector.prototype.buy = function (record) { if(this.cash > record.price){ this.cash -= record.price; this.collection.push(record) } };</pre>	

		<pre>it('record collector can buy a record', function(){ recordCollector.buy(record1); assert.strictEqual(recordCollector.cash, 61); assert.deepStrictEqual(recordCollector.collection, [record1]); })</pre>	<ul style="list-style-type: none"> ✓ record store is initially empty ✓ can add records ✓ can show store balance ✓ can print records details ✓ store can list its inventory ✓ store can sell a record ✓ store can show finances ✓ store can view records by genre ✓ record collector can buy a record 	
I & T	I.T 6			
I & T		Static and Dynamic testing task A		

Unit	Ref.	Evidence	Done
I & T	I.T 3	Demonstrate searching data in a program. Take screenshots of: *Function that searches data *The result of the function running	

Week 3

```
public static <T> List<T> getAll(Class classType){
    session = HibernateUtil.getSessionFactory().openSession();
    List<T> results = null;
    Criteria criteria = session.createCriteria(classType);
    results = getList(criteria);
    return results;
}
```

```
List<Team> foundTeams = DBHelper.getAll(Team.class);
```

```
foundTeams = {ArrayList@2167} size = 5
  0 = {Team@2175}
  1 = {Team@2176}
  2 = {Team@2177}
  3 = {Team@2178}
  4 = {Team@2179}
```

I & T

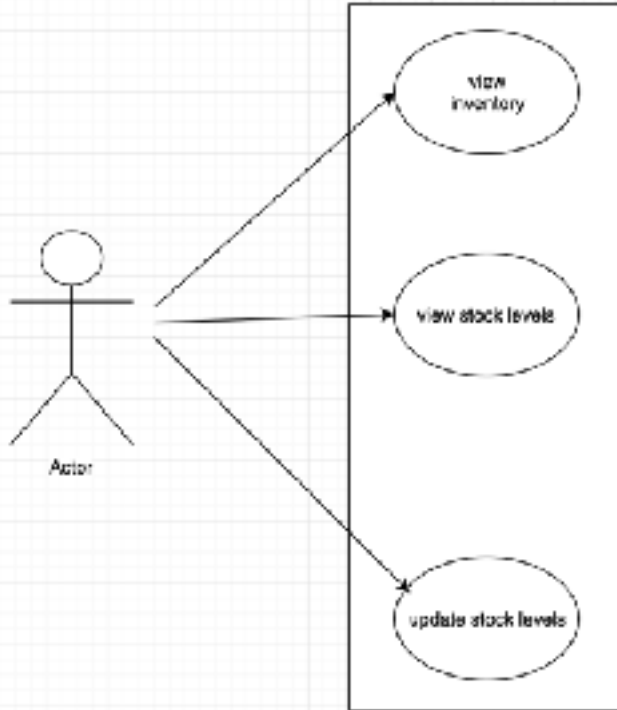
I.T 4

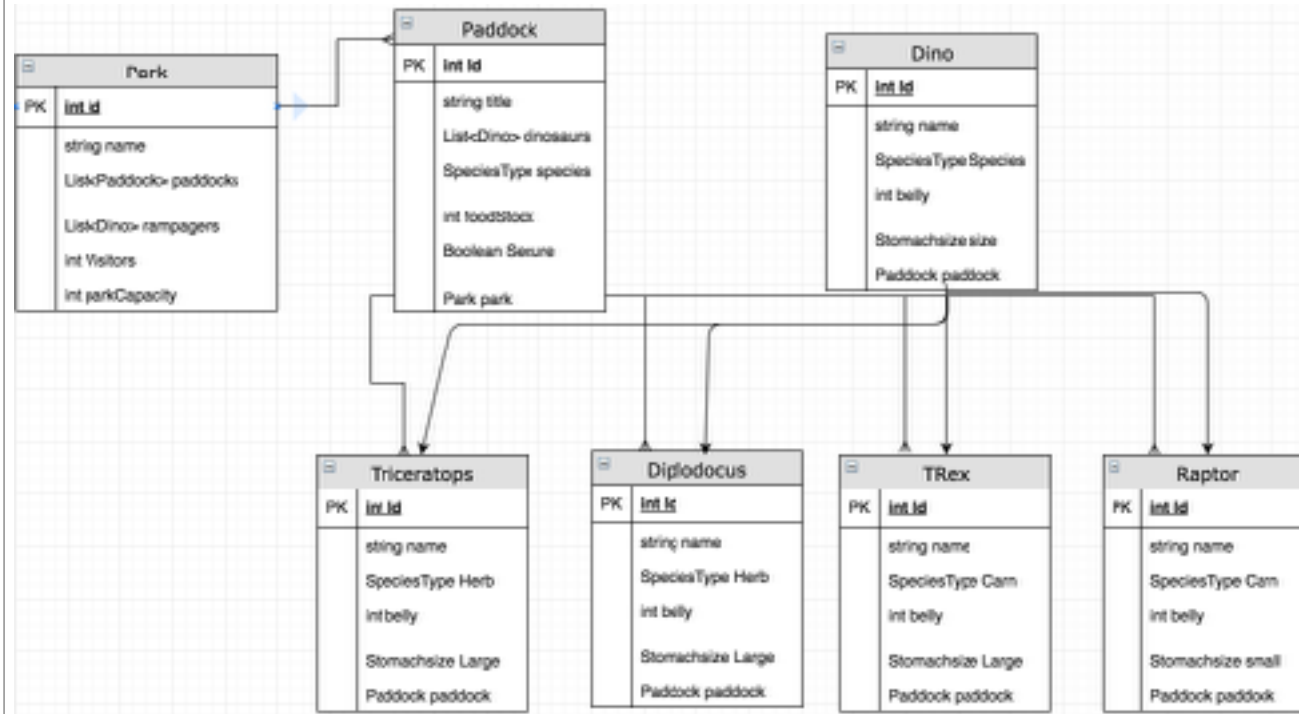
Demonstrate sorting data in a program. Take screenshots of:
 *Function that sorts data
 *The result of the function running

```
public static List<Team> getTeamsInLeague(League league){  
    session = HibernateUtil.getSessionFactory().openSession();  
    List<Team> results = null;  
    Criteria criteria = session.createCriteria(Team.class);  
    criteria.add(Restrictions.eq("league", league));  
    criteria.addOrder(Order.desc("points"));  
    results = getList(criteria);  
    return results;  
}
```

```
List<Team> teamsfoundInPointsOrder = DBHelper.getTeamsInLeague(league);
```

```
teamsfoundInPointsOrder = {ArrayList@2170} size = 5
  0 = {Team@2183}
    id = 5
    name = "Barcelona"
    points = 20
    manager = null
    league = {League@2189}
  1 = {Team@2184}
    id = 4
    name = "Newcastle"
    points = 19
    manager = null
    league = {League@2189}
  2 = {Team@2185}
    id = 1
    name = "soccer united"
    points = 12
    manager = null
    league = {League@2189}
  3 = {Team@2186}
    id = 3
    name = "Man blues"
    points = 7
    manager = null
    league = {League@2189}
  4 = {Team@2187}
    id = 2
    name = "Man reds"
    points = 3
    manager = null
    league = {League@2189}
```

Unit	Ref.	Evidence	Done
A & D	A.D 1	A Use Case Diagram	
		 <pre> graph LR Actor((Actor)) --> UC1((view inventory)) Actor --> UC2((view stock levels)) Actor --> UC3((update stock levels)) </pre> <p>The diagram is a Use Case Diagram. It features a stick figure actor on the left labeled 'Actor'. To the right of the actor is a large rectangular box containing three ovals, each representing a use case. The top oval is labeled 'view inventory', the middle one 'view stock levels', and the bottom one 'update stock levels'. Three arrows originate from the actor's right side and point to the center of each of the three use case ovals.</p>	
A & D	A.D 2	A Class diagram.	

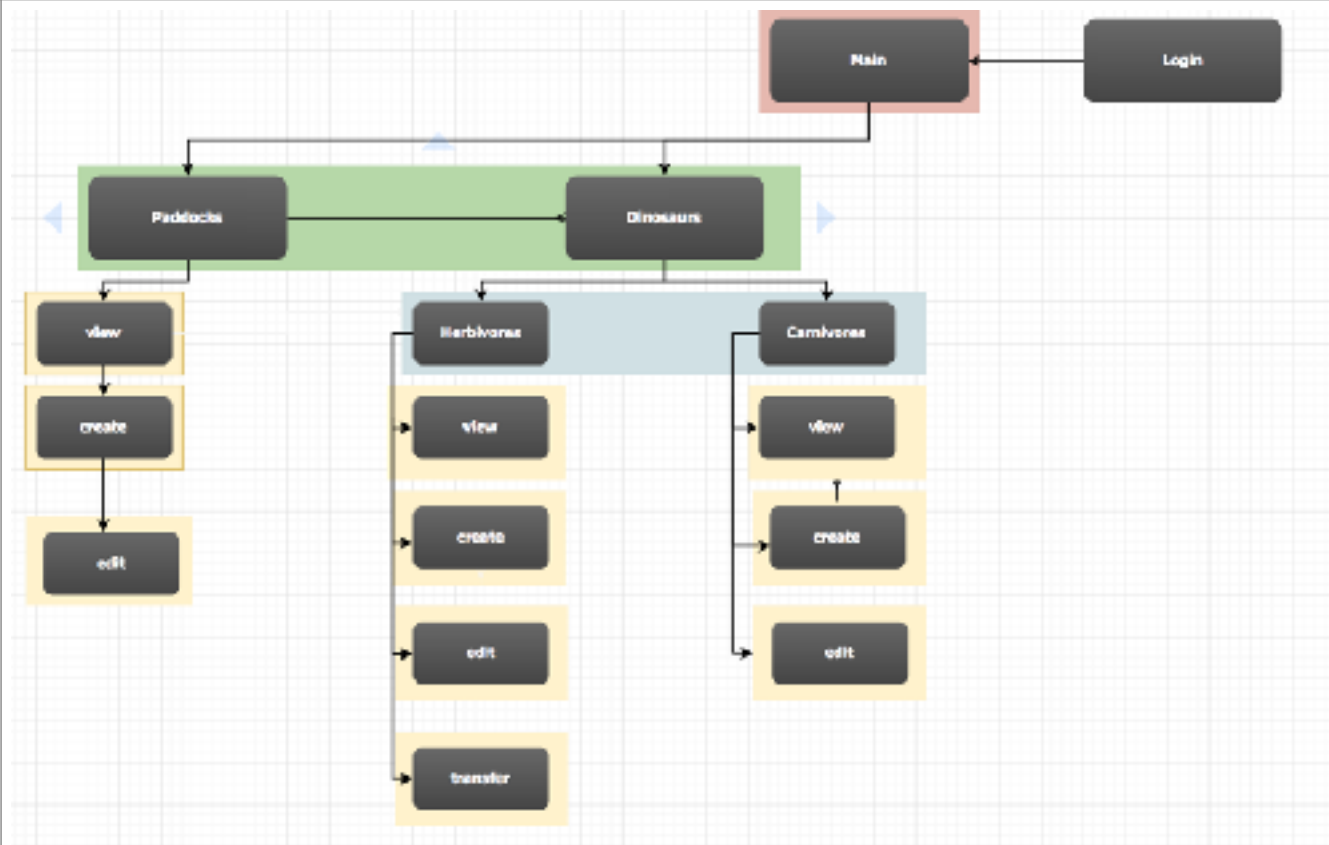


A & D A.D 3 An Object diagram.

A & D A.D 4 An Activity Diagram

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

P P 5 Create a user sitemap.



P

P 6

Produce two wireframe designs.

Paddocks

paddock	Type	Status	view	update
Paddock 1	Herbivore	Food Stock Low!	view	update
paddock 2	Carnivore	Safe	view	update
paddock 3	Carnivore	No Stock! Re-stock immediately	view	update

[Add Paddock](#)

Week 5



P

P 10

Take a screenshot of an example of pseudocode for a function.

```

let newArray = [];
// for each index position in arr.
for(let i in arr){
    // if the index number is less than the number passed in.
    if(i < index){
        // push those elements into the newArray.
        newArray.push(arr[i]);
    }else{
        // then push itemToAdd to the end of the newArray.
        newArray.push(itemToAdd);
        //then push the remaining elements to the end of newArray.
        newArray.push(arr[i]);
    }
}

```

P

P 13

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

- * The user inputting something into your program
- * The user input being saved or used in some way

User clicks new raptor link

Raptors

[New Raptor](#)
[Back](#)

Name	Species	Status	Paddock			
Echo	CARNIVORE	Raging: Check Raptors Paradise food store	Raptors Paradise	View	edit	Delete
Delta	CARNIVORE	Raging: Check Raptors Paradise food store	Raptors Paradise	View	edit	Delete
Charlie	CARNIVORE	Raging: Check Raptors Paradise food store	Raptors Paradise	View	edit	Delete
Blue	CARNIVORE	Raging: Check Raptors Paradise food store	Raptors Paradise	View	edit	Delete

Park Visitors : 0

Inputs the name of the new raptor and paddock and clicks save

SearchPath: 6547raptors/index

home
Logged in as: Admin_jags
Logout

New Raptor

Name: Paddock: Save
Cancel

Park Visitors : 0

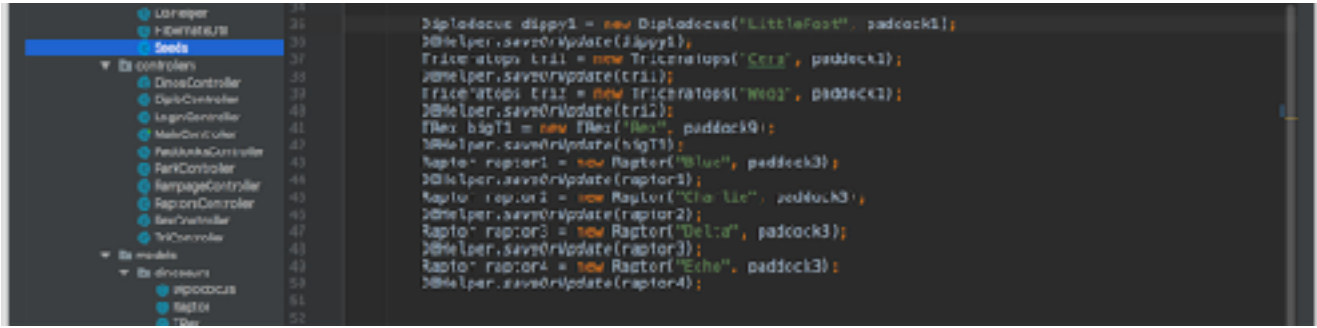
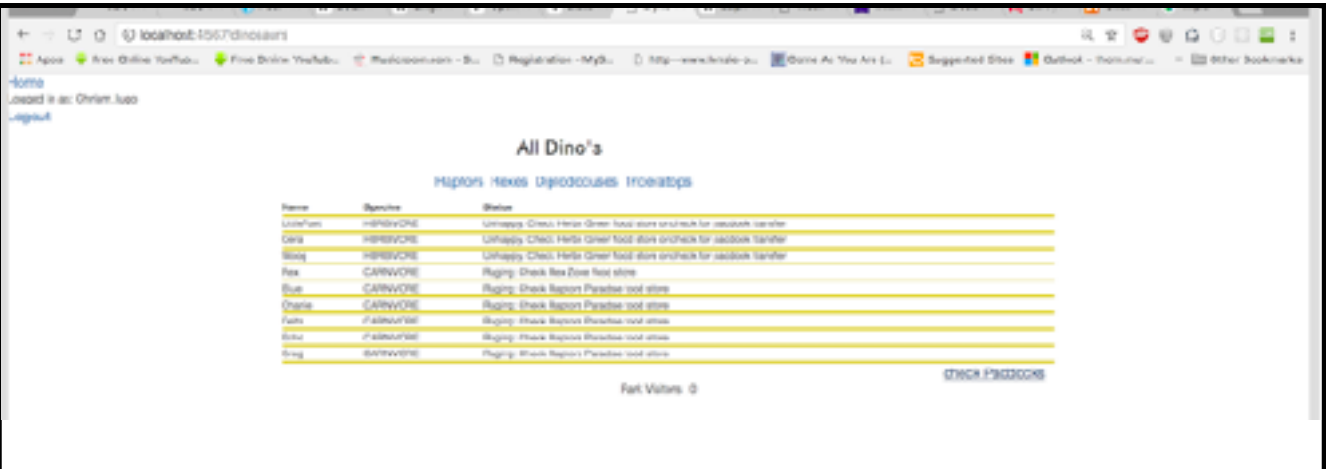
The input is saved to the database which is confirmed when viewed on raptors page.

Raptors

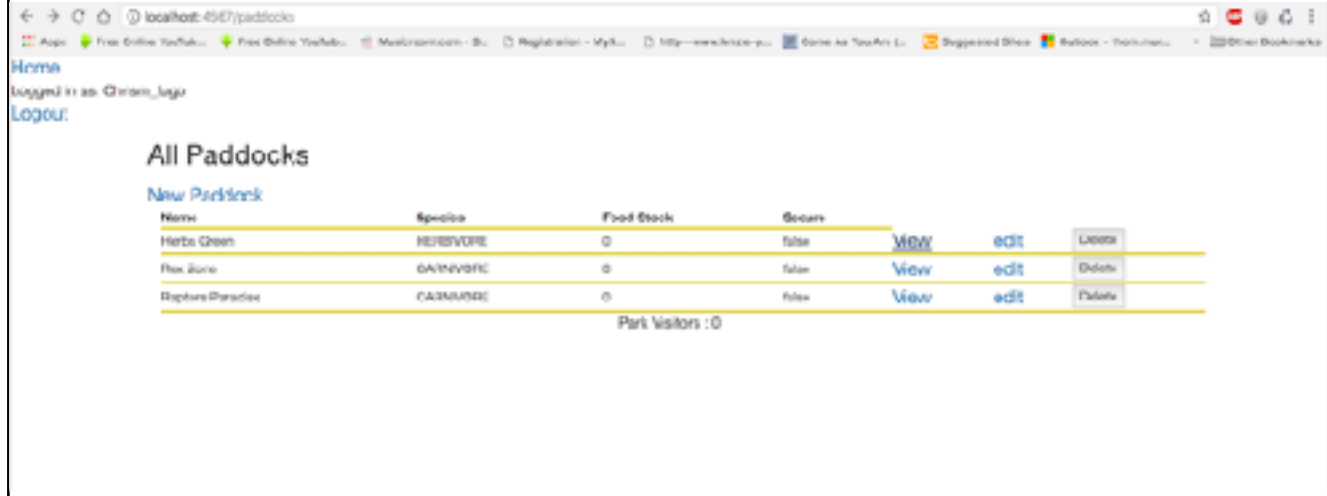
[New Raptor](#)
[Back](#)

Name	Species	Status	Paddock			
Greg	CARNIVORE	Raging: Check Raptors Paradise food store	Raptors Paradise	View	edit	Delete
Echo	CARNIVORE	Raging: Check Raptors Paradise food store	Raptors Paradise	View	edit	Delete
Delta	CARNIVORE	Raging: Check Raptors Paradise food store	Raptors Paradise	View	edit	Delete
Charlie	CARNIVORE	Raging: Check Raptors Paradise food store	Raptors Paradise	View	edit	Delete
Blue	CARNIVORE	Raging: Check Raptors Paradise food store	Raptors Paradise	View	edit	Delete

Park Visitors : 0

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>This diagram shows a seeds file which on launch of the application stores this set of dinosaurs to the database</p>  <p>Here we can see the data is persisted.</p> 	
P	P 15	<p>Show the correct output of results and feedback to user. Take a screenshot of:</p> <ul style="list-style-type: none"> * The user requesting information or an action to be performed * The user request being processed correctly and demonstrated in the program 	

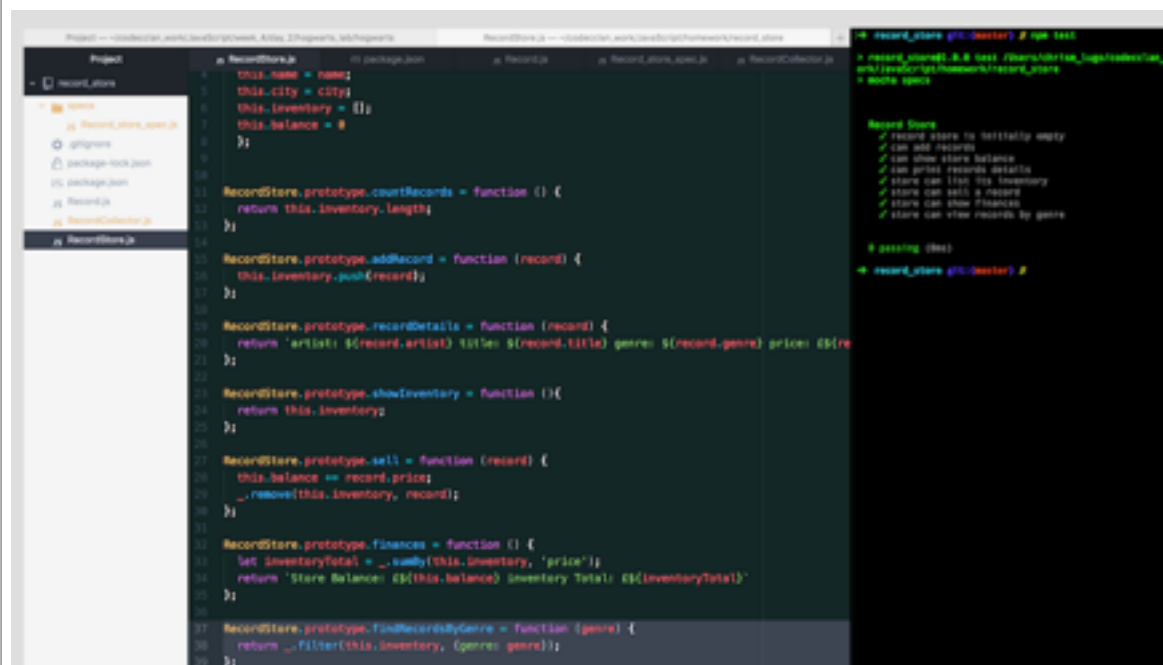
User clicks view to see details for Herbs Green

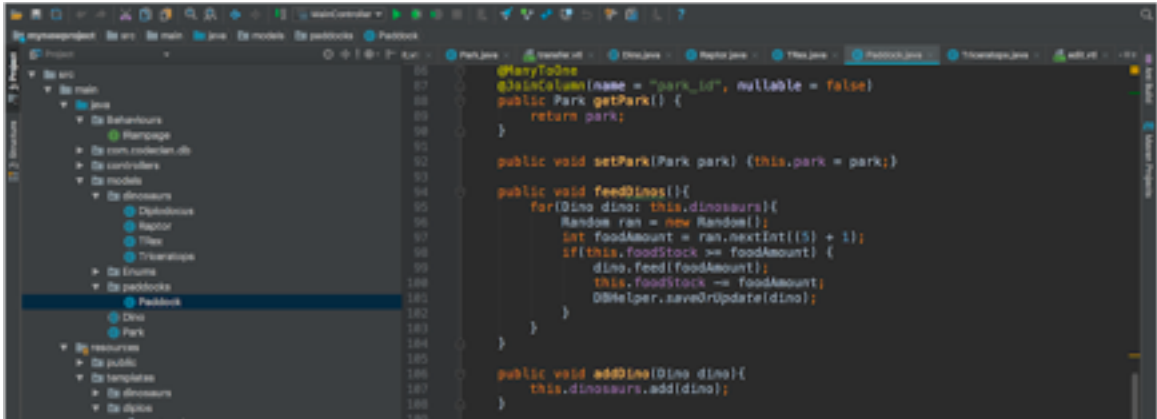


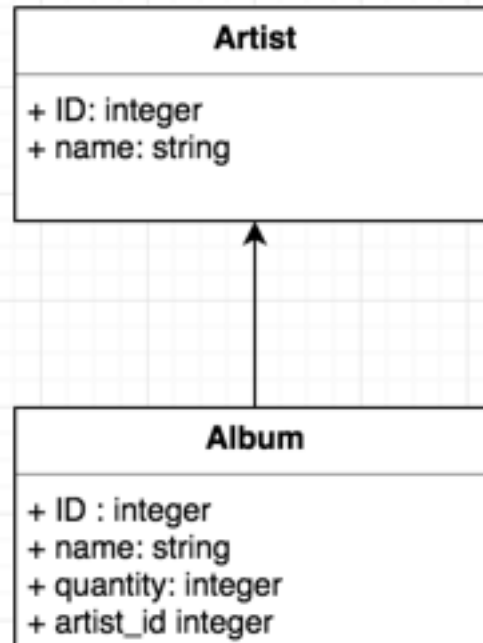
Link Navigates to paddock page for Herb's Green



	P	P 18	Demonstrate testing in your program. Take screenshots of: <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	
--	---	------	---	--



Unit	Ref.	Evidence	Done
I & T	I.T 7	Demonstrate the use of Polymorphism in a program.	
			
A & D	A.D 5	An Inheritance Diagram	



I & T

I.T 1

Take a screenshot of an example of encapsulation in a program.

```

package models.paddocks;

import javax.persistence.*;

@Entity
@Table(name="paddocks")
public class Paddock {
    private int id;
    private String name;
    private List<Dinosaur> dinosaurs;
    private SpeciesType species;
    private int foodStock;
    private boolean paddockSecure;
    private Park park;

    public Paddock() {}

    public Paddock(String name, SpeciesType species, Park park) {
        this.name = name;
        this.species = species;
        this.foodStock = 0;
        this.dinosaurs = new ArrayList<>();
        this.paddockSecure = true;
        this.park = park;
    }
  
```

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. 	
-------	-------	--	--

Week 7

This Employee Class contains all the functionality for all types of employee

```
1 public abstract class Employee {
2     private String name;
3     private String niNumber;
4     private double salary;
5
6     public Employee(String name, String niNumber, double salary) {
7         this.name = name;
8         this.niNumber = niNumber;
9         this.salary = salary;
10    }
11
12    public String getName() {
13        return name;
14    }
15
16    public String getNiNumber() {
17        return niNumber;
18    }
19
20    public double getSalary() {
21        return salary;
22    }
23
24    public void setName(String name) {
25        if (!name.equals(null)) {
26            this.name = name;
27        }
28    }
29 }
```

The Manager class shown here inherits from the Employee class.

```
1 public class Manager extends Employee {
2     private String deptName;
3
4     public Manager(String name, String niNumber, double salary, String deptName) {
5         super(name, niNumber, salary);
6         this.deptName = deptName;
7     }
8
9     public String getDeptName() {
10        return deptName;
11    }
12 }
```

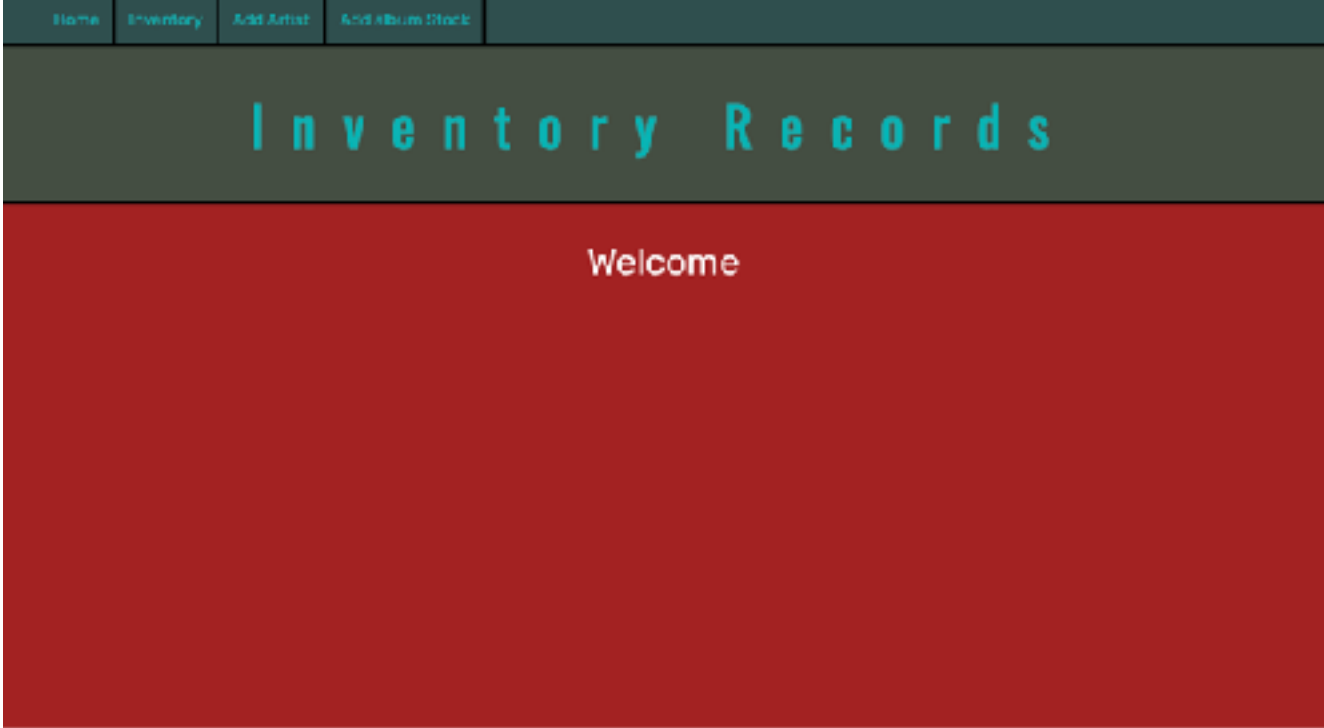
The test file here demonstrates the manager Class has inherited the methods of its parent class.

```
1 public class ManagerTest {
2     Manager manager;
3
4     @Before
5     public void before() {
6         manager = new Manager("Stephanie McIntyre", "JR3627198", 10000.00, "DevOps");
7     }
8
9     @Test
10    public void hasName() {
11        assertEquals("Stephanie McIntyre", manager.getName());
12    }
13
14    @Test
15    public void hasNiNumber() {
16        assertEquals("JR3627198", manager.getNiNumber());
17    }
18
19    @Test
20    public void hasSalary() {
21        assertEquals(10000.00, manager.getSalary());
22    }
23 }
```

Run ManagerTest

All 6 tests passed - 7ms

- hasName
- hasNiNumber
- hasSalary
- hasDeptName
- canSetName
- canSetNiNumber
- canSetSalary

P	P 11	Take a screenshot of one of your projects where you have worked alone and attach the Github link.	
		 <p>https://github.com/Chrismlugo/Project-1</p>	
P	P 12	Take screenshots or photos of your planning and the different stages of development to show changes.	


User Needs

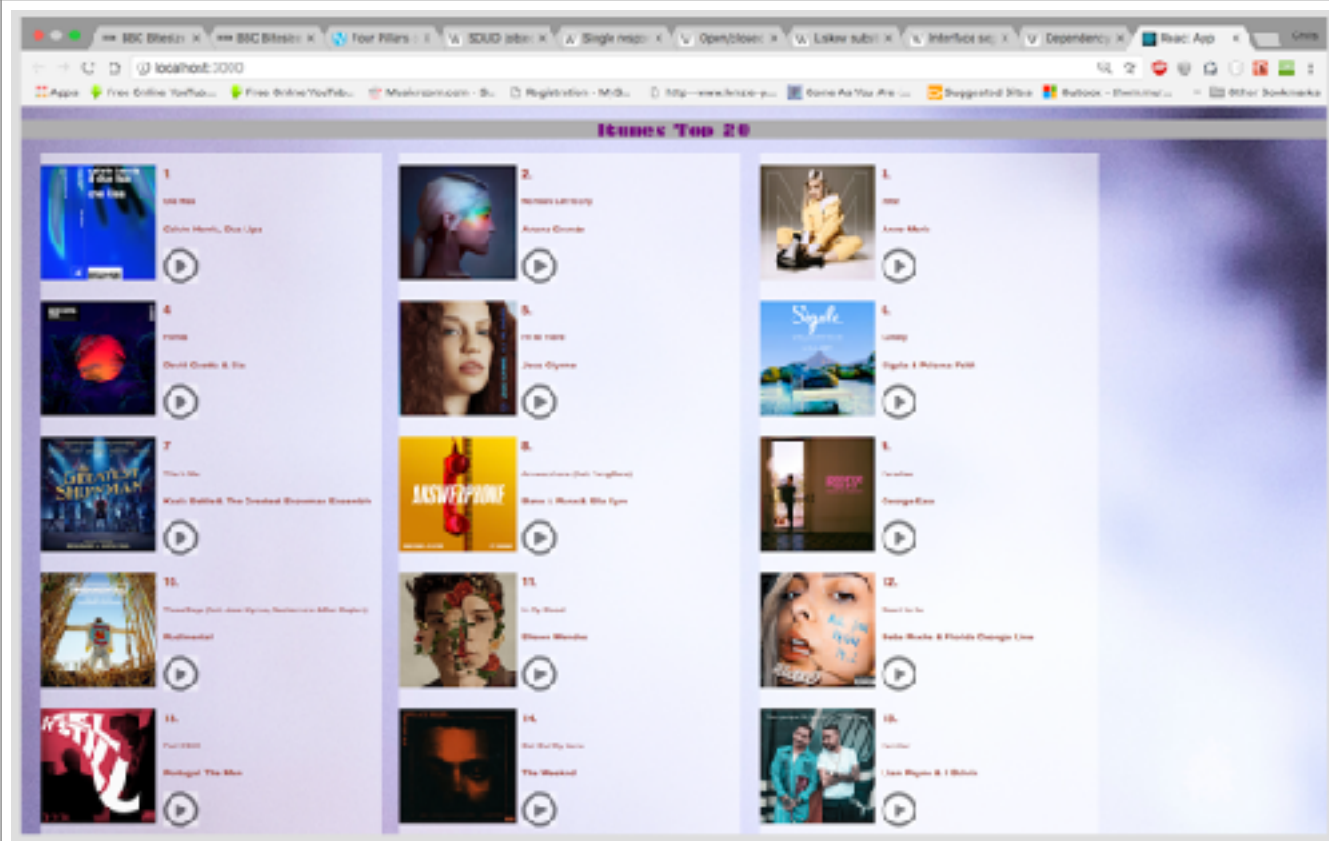
As a	I want to	So that
Business owner	Have a stock management app	I can view, update and add stock.
Person who isn't Too familiar with the latest tech	Have a straightforward Layout that's familiar.	Be able to easily understand and use the app.
Manager of a team	Have an app which is easy to use	My staff can easily use it.

User Journey

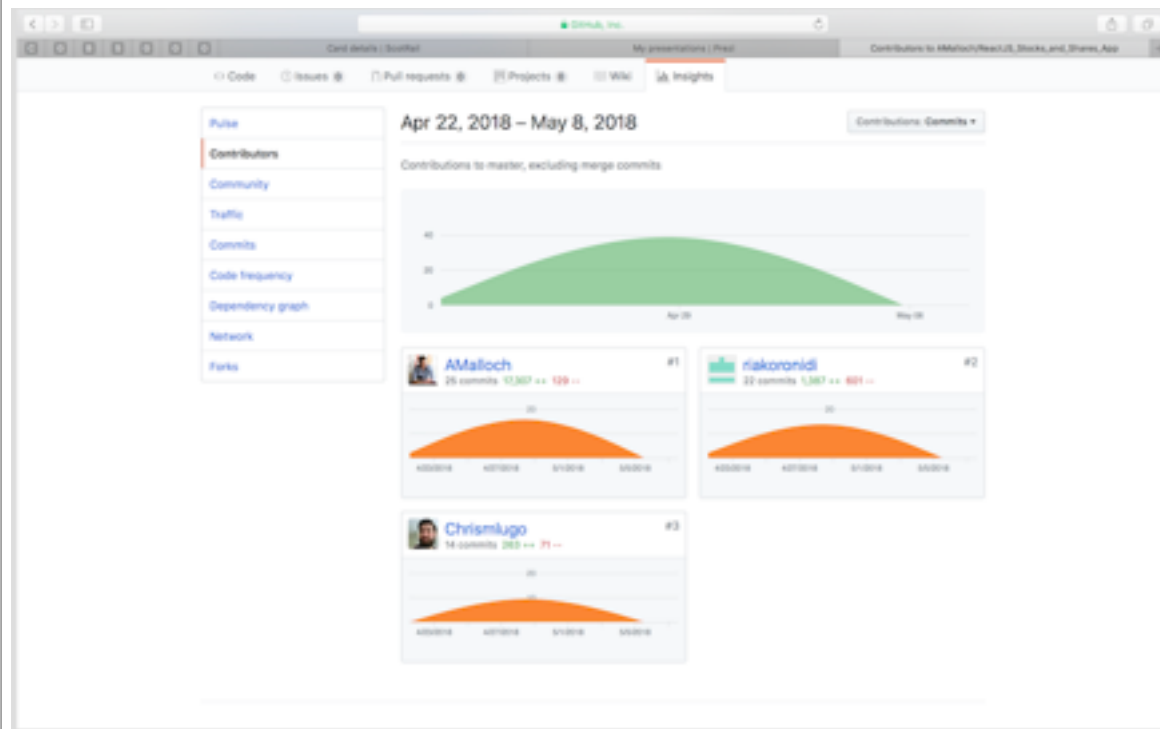
Action 1	Action 2	Action 3	Action 4	Action 5	Action 6
George views the home screen and would like to check the stores current inventory	He clicks link to inventory and views current stock	George wants to know the availability of a particular album	The album is showing low stock availability. He would like to update the inventory with new stock	George adds the new stock to the system and would like to confirm the update.	George submits the form and would like to view the inventory to confirm stock has been updated
System 1	System 2	System 3	System 4	System 5	System 6
Home page should have a link to navigate to the inventory page	Searches artists database and albums database and displays all details on screen.	Albums have a quantity. Can call a method with logic to determine high/med/low/out of stock.	link next to each entry will navigate to an edit screen with a pre-populated form.	Form will include a quantity bar which will be a number input.	The submit button will update the album quantity and redirect to the inventory page.

Unit	Ref.	Evidence	Done
I & T		Unit, integration and acceptance testing task B	

Week 11			https://github.com/Chrismlugo/calculator_unit_and_integration_testing_mocha_selenium	
	P	P 16	<p>Show an API being used within your program. Take a screenshot of:</p> <ul style="list-style-type: none"> * The code that uses or implements the API * The API being used by the program whilst running 	
			 <pre> componentDidMount(){ fetch("https://itunes.apple.com/gb/rss/topsongs/limit=20/json") .then(response => response.json()) .then(json => this.setState({songs: json.feed.entry})); } handlePlayPause(audio){ audio.paused ? audio.play() : audio.pause(); audio.classList.toggle('playing'); } render(){ return (<div> <Header/> <SongList songs={this.state.songs} handlePlayPause={this.handlePlayPause} /> </div>); } </pre>	



Unit	Ref.	Evidence	Done
P	P 1	Take a screenshot of the contributor's page on Github from your group project to show the team you worked with.	



P

P 2

Take a screenshot of the project brief from your group project.

Shares App

More and more people are playing the stock market. 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 minimal viable product (MVP) that uses the data to display her portfolio in useful ways so that she can make better decisions.

MVP

- View total current value of the portfolio
- View information for individual shares held
- Ability to 'buy'/add more stocks to the portfolio and 'sell'/remove them and update total value.
- View performance trends for individual stocks and the portfolio as a whole

Stock api's can be found here - <https://github.com/toddmotto/public-apis#finance>

If you want to make your own api an object may look something like this:

```
{
  "name": "Fusionex",
  "epic": "FXI",
  "price": 120.00,
  "quantity": 2000,
  "buyPrice": 80.00,
  "pastCloseOfDayPrices": [92.00, 89.00, 103.00, 125.00, 108.00, 98.00, 110.00],
  "buyDate": "2014-11-15"
}
```

P

P 3

Provide a screenshot of the planning you completed during your group project, e.g. Trello MOSCOW board.

Week
13

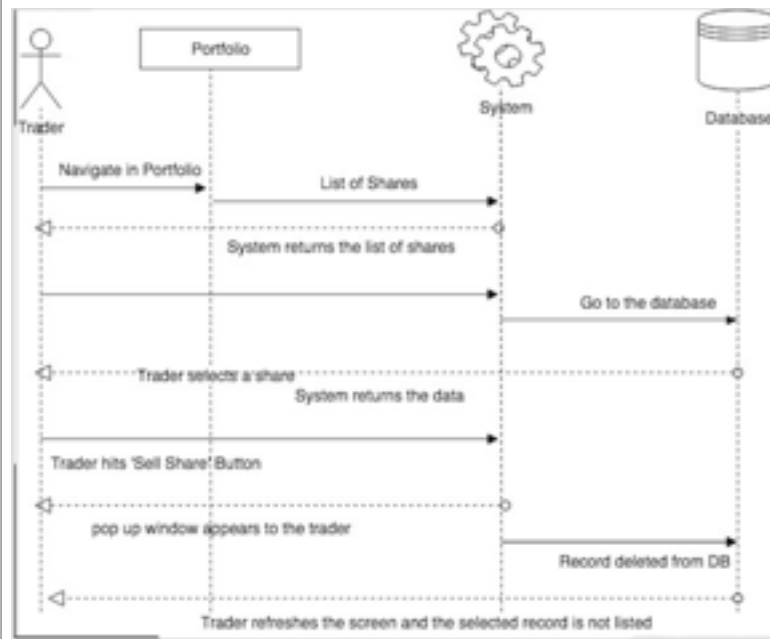


P

P 4

Write an acceptance criteria and test plan.

		<table><tr><th colspan="3">Acceptance Criteria</th></tr><tr><th>Acceptance Criteria</th><th>Expected Result/Output</th><th>Pass/Fail</th></tr><tr><td>User is able to view the total value their portfolio</td><td>When link to portfolio is clicked, user is navigated to new page which displays total value of the portfolio on the page</td><td>Pass</td></tr><tr><td>User is able to view an individual share that they own</td><td>On the Portfolio page, the user will select from a dropdown the desired share they want to view details of that share on screen</td><td>Pass</td></tr><tr><td>User can view the performance of their portfolio against the current market</td><td>When an individual share is selected, the page will display a comparison between the bought share and the current market price, showing a profit/loss value</td><td>Pass</td></tr><tr><td>User can buy an amount of shares from the stock market</td><td>On the stock market page, the user can select the chosen company, Select a quantity they'd like to buy and hit submit which adds to user's portfolio</td><td>Pass</td></tr></table>	Acceptance Criteria			Acceptance Criteria	Expected Result/Output	Pass/Fail	User is able to view the total value their portfolio	When link to portfolio is clicked, user is navigated to new page which displays total value of the portfolio on the page	Pass	User is able to view an individual share that they own	On the Portfolio page, the user will select from a dropdown the desired share they want to view details of that share on screen	Pass	User can view the performance of their portfolio against the current market	When an individual share is selected, the page will display a comparison between the bought share and the current market price, showing a profit/loss value	Pass	User can buy an amount of shares from the stock market	On the stock market page, the user can select the chosen company, Select a quantity they'd like to buy and hit submit which adds to user's portfolio	Pass	
Acceptance Criteria																					
Acceptance Criteria	Expected Result/Output	Pass/Fail																			
User is able to view the total value their portfolio	When link to portfolio is clicked, user is navigated to new page which displays total value of the portfolio on the page	Pass																			
User is able to view an individual share that they own	On the Portfolio page, the user will select from a dropdown the desired share they want to view details of that share on screen	Pass																			
User can view the performance of their portfolio against the current market	When an individual share is selected, the page will display a comparison between the bought share and the current market price, showing a profit/loss value	Pass																			
User can buy an amount of shares from the stock market	On the stock market page, the user can select the chosen company, Select a quantity they'd like to buy and hit submit which adds to user's portfolio	Pass																			
P	P 7	Produce two system interaction diagrams (sequence and/or collaboration diagrams).																			



P P 8 Produce two object diagrams.

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 P 17 Produce a bug tracking report

Bug Tracking

Deleting all portfolio shares should return an empty drop down list but instead the screen would go blank		Added an if statement to check if array is empty	Passed
Values returned from buying/selling should be return upto 2 decimal places		Adds code to round results to 2 decimals	Passed
Bootstrap table data should be accessed from props.		Data passed through render in route	Passed
Balance should persist throughout application after buy or sell	Failed		
Must re-render page after share has been sold	Failed		