

**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:** | |

|  |
| --- |
|  |
| **Here we define a function order\_numbers, which takes in a variable and will carry out a sorting method to rearrange the elements of the array to be ascending. (This was created in the Test class)** |
|  |
| **Here we test the function order\_numbers on the testarray [2,5,3,4]. The function should rearrange testarray’s order to be [2,3,4,5]** |
|  |
| **From the terminal output we can see the assertion has passed and the function operates as expected.** |

|  |  |  |  |
| --- | --- | --- | --- |
| **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:** | |

|  |
| --- |
|  |
| **This is the basic array of customers. It contains 2 entries of hashes** |
|  |
| **This is the function “remove\_customer\_cash” it takes 2 arguments, person and cost. The 2nd line of code sets a variable “sum\_to\_remove” equal to the negative of the variable cost. The 3rd line of code returns the entry in @customers which matches the specified “person”. It then updates the :cash key by removing the cost value(sum\_to\_remove)** |
|  |
| **This shows the test function for remove\_customer\_cash. When we input a cost of 100 the customer’s cash should reduce from 1000 to 900. As you can see from the terminal image this assertion comes back true.** |

**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:** | |

|  |
| --- |
|  |
| **This is a find function defined within the Hero class of a website. It takes all the information for the database on a particular hero where the id matches the search id.** |
|  |
| **This is where the viewer calls on the find function to display a page for a specific hero(id).** |
|  |
| **This is the output of the view page, it displays all the information on the hero with id 2.** |

|  |  |  |  |
| --- | --- | --- | --- |
| **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:** | |

|  |
| --- |
|  |
| **defining the function all\_true for the Hero Class. this selects from the database all the information from the heroes table on the condition where hireable = true. This is used to restrict/sort the list of all heroes down to just ones that are hireable.** |
|  |
| **This function is called in the join table (hero\_teams) controller, for the new view. it is carried out on the @heroes variable which goes through all the heroes in the heroes table and gives the user only the heroes which are hireable.** |
|  |
| **This shows where the reduced hero list is used in the new view form. It is a dropdown list that shows each heroes name.** |
|  |
| **Here we can see the resulting interface on the website. The 4 heroes listed are the from the heroes table where their hireable status is equal to true.** |

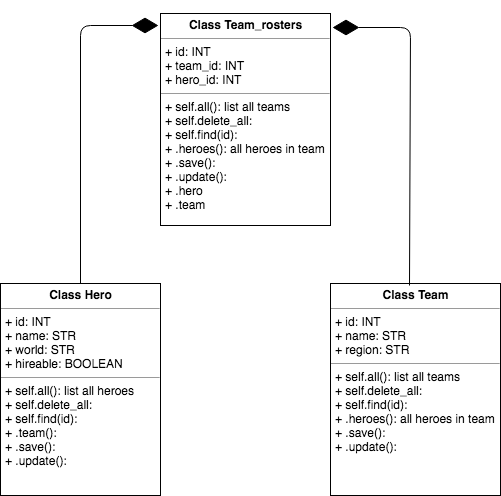
**Week 5 and 6**

|  |  |  |  |
| --- | --- | --- | --- |
| **Unit** | **Ref** | **Evidence** |  |
| **A&D** | A.D.1 | A Use Case Diagram | |
|  |  | **Description:** | |

[****](https://www.draw.io/#G12zTVdYTJELatyx538YwznHbEyE77LflO)

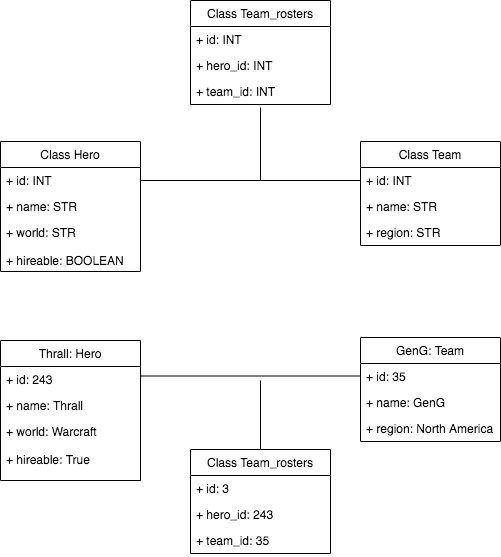
**This Case diagram shows the user which is the league manager and the functions that the subsystem Hero academy app needs to be able to perform.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Unit** | **Ref** | **Evidence** |  |
| **A&D** | A.D.2 | A Class Diagram | |
|  |  | **Description:** | |

[****](https://www.draw.io/#G1SD--vxqSGNbf9vfih_SDQEc0GtGtu6ze)

**This class diagram shows the three classes that are required for the hero\_academy app. The class Hero and class Team contain several properties outside their id’s, and many functions. Both class’ feed into the class Team\_rosters which only contains properties of the id of the other classes. It contains several functions**

|  |  |  |  |
| --- | --- | --- | --- |
| **Unit** | **Ref** | **Evidence** |  |
| **A&D** | A.D.3 | An Object Diagram | |
|  |  | **Description:** | |

[****](https://www.draw.io/#G13aqjgMZtgtuA8MhMvI1qi0rodcAZRGKq)

**This object diagram shows an example of what a Hero would be, a Team and how the Team\_rosters information relates between the two.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Unit** | **Ref** | **Evidence** |  |
| **A&D** | A.D.4 | An Activity Diagram | |
|  |  | **Description:** | |

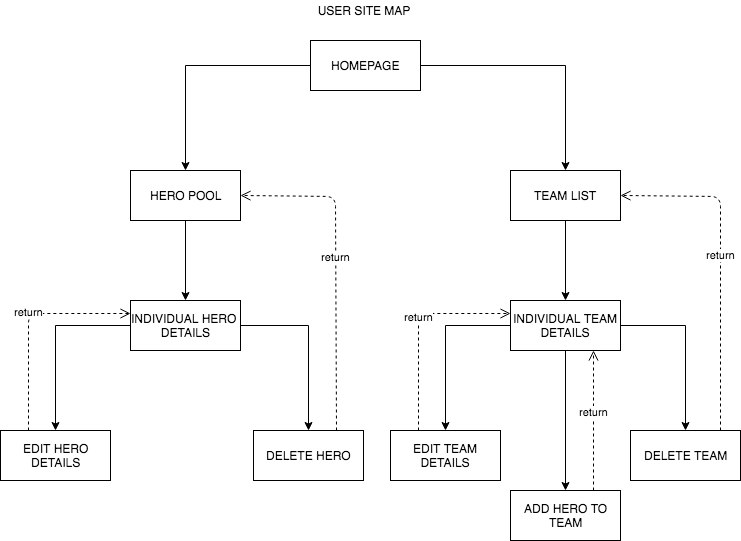
[****](https://www.draw.io/#G1YNbeMjMw7Y4PJb8J9B9Dq10Ztw4ONxE8)

**This activity diagram shows the activity of adding a new hero to a team’s roster. The flow shows the steps that the manager might take and the sort of responses and process the Hero Academy app would need to carry out to fulfill those needs.**

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

|  |  |  |
| --- | --- | --- |
| **Constraint Category** | **Implementation Constraint** | **Solution** |
| Hardware and software platforms | users using different web browsers, compatibility issues with the designed app. If the app crashes then the user might not come back to the site.  If this happens we will lose traffic and eventually income.  **MORE detail. Why? impact? etc** | carrying out testing of the app on each platform, altering back end to compensate for any errors. |
| Performance requirements | bandwidth of user. If the user doesn’t have enough bandwidth our page might load slowly. This could put the user off using our site, and stop using the site.  If this happens we will lose traffic and eventually income. | ensure the app is built efficiently, no large downloads |
| Persistent storage and transactions | not enough storage space on server for all user information.  This could cause error/crashes on our site, and discourage users from using our site.  If this happens we will lose traffic and eventually income. | buy additional server space/servers if required |
| Usability | app is very visual, not being usable by the visually impaired. If the user can’t interact with the site then they will not continue to use the site. This is a loss in potential income | ensure usability is coded in from the beginning so all users can use the app |
| Budgets | enough money to pay programmers . if we don’t control out budget and stop paying the programmers we will run into contractual issues with them. While sorting this out the app will stop getting developed. If we can’t develop then we will fall behind our competitors, and potential lose market share and therefore income | project manage the app development to ensure the budget lasts the duration |
| Time | lose market share by releasing too late, another developer release a competitive app.  If we lose market share we’ll lose customers and therefore potential income | project manage the app development to ensure the app releases in a set time frame. Know when to cut off features relative to release |

|  |  |  |  |
| --- | --- | --- | --- |
| **Unit** | **Ref** | **Evidence** |  |
| **P** | P.5 | User Site Map | |
|  |  | **Description:** | |

[****](https://www.draw.io/#G1pQ7iZrweWRiM2x9ubzql7BMu_gd0cIRj)

This user map shows the layout of the website hero\_academy. It demonstrates how the user can navigate across the individual pages and how the pages will return the user to previous pages once their function is completed.

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

|  |
| --- |
|  |
| **This is where the project began,it shows the general layout of the basic pages. The view at the top is for the front page, it hold a title and two buttons. The two buttons link to the pages below and to the left. These pages have data and buttons to link further on.** |
|  |
| **A later step in the planning it has similar beginning pages, but contains more details for each add hero/team/hero to team page.**  **It can be seen for the add hero to team roster that the content of the drop down menu for the team select and hero select has now been detailed.** |
|  |
| **This shows the webpage wireframe for reusable views inside a webpage which lists a continent and all of its countries in list elements** |

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

|  |
| --- |
|  |
| This is the Pseudocode for the below guest\_pay method. the method is to check whether a guest can afford to get into a lub and updates their wallet value. |
|  |

|  |
| --- |
|  |
| This is the Pseudocode for the below customers method. It grabs information from sql tables. |
|  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Unit** | **Ref** | **Evidence** |  |
| **P** | P.13 | Show user input being processed according to design requirements. Take a screenshot of:  \* The user inputting something into your program  \* The user input being saved or used in some way | |
|  |  | **Description:** | |

|  |
| --- |
|  |
| **This is the form for adding a new hero to the academy. It has two text fields for the user to fill in and one radial button to choose from.** |
|  |
| **This is the form filled in by the user** |
|  |
| **This is the resulting information produced by submitting the add hero form, displayed on the list of heroes page.** |
|  |
| **This is the individual hero’s detail page, you can see all the information that was entered about the new hero ‘Ragnaros’ entered by the user.** |

|  |  |  |  |
| --- | --- | --- | --- |
| **Unit** | **Ref** | **Evidence** |  |
| **P** | P.14 | Show an interaction with data persistence. Take a screenshot of:  \* Data being inputted into your program  \* Confirmation of the data being saved | |
|  |  | **Description:** | |

|  |
| --- |
|  |
| **This is the starting point for the team list of the website and the corresponding postico/server information for the teams table.** |
|  |
| **This is the form for adding a new team, it has been filled in with information** |
|  |
| **This is the resulting display for the team list on the website, ‘Red Lobsters’ have now been added to the list. The data has also been saved to postico/server under id:3.** |

|  |  |  |  |
| --- | --- | --- | --- |
| **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 homepage of the website. The user selects the “Teams” button** |
|  |
| **Which takes them to a list of all Teams. Selecting the 1st team “GenG” link** |
|  |
| **We see all of the team’s data, including the table containing the hero’s associated with that specific team.** |
|  |
| **Here we can see all of the heroes in the hero pool, those hired and those available for hire. Looking at the id’s Jaina is 5, Kael’thas is 6 and Kerrigan is 7.** |
|  |
| **When we look at the join table “hero\_teams” we can see that for team GenG (team\_id =1) the correct heroes should 5, 6 and 7 which is what the table on the Team details page displays.** |

|  |  |  |  |
| --- | --- | --- | --- |
| **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:** | |

|  |
| --- |
|  |
| **Github link; https://github.com/Castlecelts/Ruby\_Project** |

|  |  |  |  |
| --- | --- | --- | --- |
| **Unit** | **Ref** | **Evidence** |  |
| **P** | P.12 | Take screenshots or photos of your planning and the different stages of development to show changes. | |
|  |  | **Description:** | |

|  |
| --- |
|  |
| **This is where the project began, creating some wireframes for each web page with basic functionality for each.** |
|  |
| **This was the 2nd step in planning, thinking about what each database table would contain. As you can see the name of the join table altered during the project from “Team\_roster” to “Hero\_teams” based off a formatting suggestion.** |
|  |
| **A later step in the planning was to map how each web page would interact with others. the controllers that needed to be in place to run those pages and interactions and what the url for each page would be. This was particularly relevant for the add hero to team roster page as it was accessed from a “Team” page but would be updating the join table information.**  **This step in the planning was near how the finished project ended.** |

**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:** | |

|  |
| --- |
|  |
| This is the code for the defining the RequestHelper. This is used to make requests to an API |
|  |
| This is the model “BeerData” it calls in the previous RequestHelper to access the brewdog api (“https:api.punkapi.com/v2/beers”). using the .get().then() we extract the data from the api and store it to the beerData. this allows us to use it in the rest of the app. We now publish that data over the channel “BeerData:all-data-ready”. We also publish a refined set of data (“abvs”) over the “BeerData:abv-data-ready” channel. We will follow this second channel. |
|  |
| The SelectAbvView handles the drop down select menu on the app. We begin by subscribing to the channel “BeerData:abv-data-ready” and store those details in “abvs”. |
|  |
| We then call the function populateAbvs to manipulate the data and assign each abv % to its own option in the dropdown menu. |
|  |
| In the browser, this is the main view display. |
|  |
| clicking on the select dropdown we see the list of available ABV %. |
|  |
| After selecting 4.5 we can see the resulting beers that have the ABV content of 4.5%. |

|  |  |  |  |
| --- | --- | --- | --- |
| **Unit** | **Ref** | **Evidence** |  |
| **P** | P.18 | Demonstrate testing in your program. Take screenshots of:  \* Example of test code  \* The test code failing to pass  \* Example of the test code once errors have been corrected  \* The test code passing | |
|  |  | **Description:** | |

|  |  |
| --- | --- |
|  |  |
| **Initial Code** | **Finished Code** |

|  |
| --- |
|  |
| **Running spec file, no assertions** |
|  |
| **checkforace true failing** |
|  |
| **checkforace true passing** |
|  |
| **checkforace false passing** |
|  |
| **checkhighest where card1 is higher failing** |
|  |
| **checkhighest where card1 is higher passing** |
|  |
| **checkhighest where card2 is higher passing** |
|  |
| **checkhighest where card1 equals card2 passing** |
|  |
| **cardstotal on an array of 2 cards failing** |
|  |
| **cardstotal on an array of 2 cards passing** |
|  |
| **cardstotal on an empty array passing** |

|  |  |
| --- | --- |
|  |  |
| **Final testing spec code (page1)** | **Final testing spec code (page2)** |

**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:** | |

|  |
| --- |
|  |
| Here is our contribution for the Javascript group project. My username is Castlecelts. The work was split between individual, paired and mob programming. |

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

|  |
| --- |
|  |
| As a group we needed to write our own MVP due to our chosen brief. The MVP and Extensions were agreed by all of the group members and tutors. |

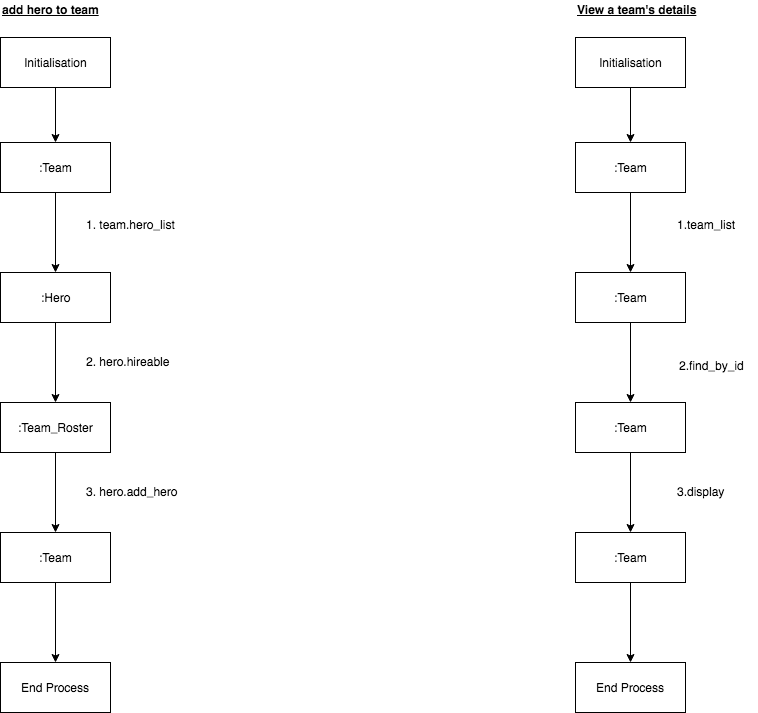
|  |  |  |  |
| --- | --- | --- | --- |
| **Unit** | **Ref** | **Evidence** |  |
| **P** | P.3 | Provide a screenshot of the planning you completed during your group project, e.g. Trello MOSCOW board. | |
|  |  | **Description:** | |

|  |
| --- |
|  |
| This is our Moscow Board, laying out what we must achieve, should achieve, could achieve and won’t achieve over the course of the project. |
|  |
| This is our “sprint 1” where we planned out our diagrams for the project. Including our user, wireframe, and class relationships. |

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

|  |
| --- |
| **in slides**  **3 column table**  **looking at how user can operate your app** |
| These are the acceptance criteria for our expected user for the top trumps app. The user is a teenager interested in science and the stars. |

|  |  |  |  |
| --- | --- | --- | --- |
| **Unit** | **Ref** | **Evidence** |  |
| **P** | P.7 | Produce two system interaction diagrams (sequence and/or collaboration diagrams). | |
|  |  | **Description:** | |

[****](https://www.draw.io/#G17vwQXVGR2mPYja_f-vvUZZbR0JHrbVKD)

There are 2 wireframes here of the collaboration diagram type.

The 1 on the left is for **adding a hero to a team.**

* step one calls the method hero\_list to grab all of the heroes.
* step two calls the method hireable to filter the list of heroes to only ones that are hireable.
* step three call the method add\_hero which adds the hero to the team in the join table Team\_Roster.

The 1 on the right is for **viewing a Team’s details.**

* step one call the method team\_list to grab all of the teams.
* step two calls the method find\_by\_id to find all of the information on the specific team.
* step three calls the method display which displays the information on the specific hero.

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

|  |
| --- |
|  |
| This shows the object relationships within the javascript project. The information that is planned to be transferred between the different objects. |
|  |
| this is the extended object diagram for the javascript project at the point where there was information issues. This helped us to track where duplicate information was being transferred and where objects were talking to objects when they shouldn’t be. |

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

|  |
| --- |
|  |
| This is the Bug Report for an error on our group project. It shows an issue with the game logic as a game is completed. |

**Week 12**

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

|  |
| --- |
| The scenario is we are creating a vehicle which has various properties as well as having a list of all components it is constructed of. The components themselves have many properties but not many overlaping ones i.e. tyres, engines, fuel tank all rely on different attributes. To be able to create an (Array)List of all of the components in a vehicle we must use polymorphism as JAVA does not let you create a list of different types of objects. |
|  |
| Here we create an Interface of IComponent which is used to give a common type to all of the classes which should be components. The function hasA is used to allow us to query what each object in the arrayList is. |
|  |
| In the Engine class we implement IComponent which requires it to have a hasA function. |
|  |
| In the FuelTank class we implement IComponent which requires it to have a hasA function. |
|  |
| Now when we create our Vehicle class we can have an ArrayList<IComponent> in the constructor. When we now create an instance of Vehicle the vehicle can contain components of class engine and/or class fuelTank resolving our issue. |

|  |  |  |  |
| --- | --- | --- | --- |
| **Unit** | **Ref** | **Evidence** |  |
| **A&D** | A.D.5 | An Inheritance Diagram | |
|  |  | **Description:** | |

|  |
| --- |
|  |
| This shows the use of inheritence for types of players. PlayerClass defines certain properties which all types of players should have. The Barbarian and SpellCaster class inherit these properties. The SpellCasterClass defines a new variable mana, and a new method useMana. When Warlock & Cleric Class inherit from SpellCaster Class they take the new variable and method and everything from the Player 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. here brand and type are defined as “private” which means we cannot set or read them. This is why we need to define the methods getBrand and getType. These methods allow you to access the information of the brand and type. |

|  |  |  |  |
| --- | --- | --- | --- |
| **Unit** | **Ref** | **Evidence** |  |
| **I&T** | I.T.2 | Take a screenshot of the use of Inheritance in a program. Take screenshots of:  \*A Class  \*A Class that inherits from the previous class  \*An Object in the inherited class  \*A Method that uses the information inherited from another class. | |
|  |  | **Description:** | |

|  |
| --- |
|  |
| This is our abstract class Vehicle. useFuel and refillFuel is taken from the class FuelTank via IComponents. |
|  |
| This is our class Electric which extends/inherits from Vehicle. This uses reFuel from Vehicle and adds an addition to the string output. |
|  |
| This is the test for Electric (Vehicle), it has an object of Electric car |
|  |
| The second last test “drive” uses the function drive which is defined in the Vehicle class and is inherited by the Electric Class. This shows that drive can be enacted on car(Electric Class) |

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

|  |
| --- |
|  |
| I chose this algorithm “getFuelTanks” as it helped to solidify my understanding of Interfaces. This algorithm is created in the abstract class vehicle which contains a variable ArrayList of IComponent(s). If ArrayList contains any bodies of the class FuelTank (which implements IComponent) then the body will be added to the ArrayList results.  line43 creates a new variable results of type ArrayList  line 44 starts a for loop. for every component of “this.components” if the component is an instance of FuelTank (of same class) then add the component to results. |

|  |
| --- |
|  |
| I chose this algorithm “drive” as it builds on the previous algorithm which is interesting. This algorithm is created in the same abstract class. line 53 sets up a fueltanks variable which is populated using the previous algorithm getFuelTanks. line 54 then empties the fuelTank by an integer distance (how much fuel was used to drive a distance). This algorithm can then be used in any children of VehicleClass. |