**Evidence Cover Sheet – Object Orientated Programming**

|  |  |  |  |
| --- | --- | --- | --- |
| **General Information** | | | |
| Analyst/Engineer | Ross Fletcher | Location | BAE Systems Applied Intelligence |
| Date | 26/07/16 | Time Taken | 10 hours |

|  |  |
| --- | --- |
| **Evidence to provide** | **Use this column to either add suitable images/ and or comments.**  **You can also attach documents separately if you wish.** |
| Designs (flowcharts, pseudocode, UML etc) which should include information about:   * Objects, data and files structures you intend to create/use * Variables you intend to use * Object relationships * Object behaviours/control structures that will be required * Input/Output that are required * Operators that are required | My design document has been attached separately |
| Use of the Development IDE, for example code window, using templates and creating projects | Below is a screenshot of my development IDE, I used Netbeans as it has better integration for GUI forms and has native support for form generation and allows GUI programs to be edited on the fly. This allowed me to develop my program quickly and also use the built in tools like the debugger.    Net beans also has a built in GUI form builder, this is shown below and it allows me to easily create java forms and use them within my finished program.    You can download NetBeans from the official oracle website  https://www.jetbrains.com/idea/ |
| A full code listing, which should include:   * Standard naming conventions * Layout * Comments * Encapsulation * Polymorphism * Inheritance * Data validation * Error handling & reporting | I have attached the source code for my program.  Within these I have used standard naming conventions such as camelCase, this is a variable and class naming convention where multiple words are concatenated into one string and the first letter of each word is upper case – except the first letter of the first word.  My program then has large amount of comments on it that describe what is happening at complex parts of the program.  I have also used JavaDocs on complex functions to explain how they use parameters and what they return  Within the program itself I have then used inheritance; my Car, Employee and Customer classes all have content that is used my by Search class to search through a large amount of records. Because of this I have had them all extending a SearchableObject class, this allows me to create a series of separate GUI search objects that will be shown to the user all built on top of the same class.  I have also implemented forEach methods and Lambda’s, these are a feature included in Java 8 and onwards. They allow quick looping though ArrayLists and doing an action for each item in the list.  Data validation is another thing that I have implemented, on my login page I ask for a username and a password and the user can only get to the rest of the app if they enter credentials that are initialised as an employee object  I have also implemented an ActionListener interface in my main – this allows me to overwrite actions within methods that have been called by any method inside the main. For example all of the GUI objects are constructed in the main, this means that whenever any actions are taken by the user in any of the classes then they will call the methods that are implemented as part of the main. All overwrites have also been noted with the @Overwrite notation  I have also used a consistent layout within the whole of my project and consistently used spaces for indentation.  I have also used constructor overloading within my Car and Employee classes where there is multiple constructors that allow different amounts of data to be entered so objects can be created easily. |
| Use of debugging tools within the Development IDE | I have used the built in debugging features of my IDE, these allow me to create break points and step into methods to find out how my code is being executed and how variables are being used. Within my testing document I have demonstrated the use of the debugger and how I used it to fix a bug that I found in my code. |
| Test plan which should include:   * Test data * Expected results * Actual results * Identified discrepancies * Discrepancies rectified | My test plan has been attached as a separate document. |
| On-screen help to support the user | There is onscreen support throughout that app that demonstrates actions to the user on the screen whenever they click on the question mark icons – the screenshot below shows how a popup is shown when the user presses the icon on the search page. |
| Document to support the maintenance of the program | I have attached the maintenance guide as a separate document |