|  |  |
| --- | --- |
| **Free Full Frame Shot of Abstract Pattern Stock Photo**  **Java assessment 2**  **Creator GUI** | **diploma of information technology (cloud engineering)**  **Oisin Aeonn S3952320** |

|  |  |
| --- | --- |
| **Table of Contents:** | |
| **Cover Page** | **Page 0** |
| **Table of Contents** | **Page 1** |
| **Task 1** | **Page 2-4** |
| **Task 2** | **Page 4** |
| **Task 3** | **Page 5** |
| **Task 4** | **Page 6** |
| **Task 5** | **Page 7** |
| **User Guide** | **Page 8** |
| **References** | **Page 9** |
| **Appendices** | **Page 9** |

**Task 1: Analysis – Scenario**

**Outline a brief document outlining the project scenario that you have chosen to implement including any assumptions that you have made (see sample scenario). (Approximately 400-500 words)**

As previously discussed in the first phase of implementation of this project. GameCo is a large Gaming Retailer. It has many stores around Australia, New Zealand and North America.

We have developed a Creation Utility to help aid GameCo to break into the Video Game Development Market and publish its first title. They have identified a Market Gap of which they want to expand the business and increase their yearly profits.

We have developed a basic command-line basic character and location generation tool video game. Its purpose is to allow for users to create new characters and locations for any game for which to base their persona. GameCo has conducted research into the Market and there does seem to be a market for such a service. Other tools similar do exist, however after trying them they are quite unfinished buggy and unfinished which will allow this particular tool to shine in comparison.

It is to be made using Java to ensure easy cohesion with their website. It is also the language that our development team is most comfortable with and will allow for them to make future changes and additions such as writing the New Characters and Locations straight to GameCo’s Database on their website. GameCo wants a fully functioning game, manual, testing / debugging and accompanying documentation such as UML diagrams so that other developers can easily understand and continue to make updates after the initial implementation.

The assumptions I have made are:

* A GUI Interface will now be developed to show off your new snazzy characters and to improve User Experience.
* It will write to two Database tables (Website Database) in which all characters and places are stored. This Database will be hosted in a XAMPP SQL / Apache Server. The Database will be called creator, with two tables called characters and places.
* It will have 4 classes to ensure it is simple, but easy-to-read and not cluttered. These classes are to be named Gui, Database, Character, and Place.
* Must be accompanied by Documentation such as a Report, Use Guide, known Bugs & Errors, UML Diagrams, etcetera.
* It must be fully tested, working and well documented.
* A fully functioning GUI Interface with an error if you input an invalid number. (through else { command.
* Will have 7 instance variables for Character Class. (2 string, 4 integer and 1 boolean). – these are Name (str), Type (str), HP (int), Mana (int), Attack (int), Speed (int) and Evil (bool).
* Will have 3 instance variables for Place Class. (3 string). – these are Location (str), Size (str) and Biome (str).
* Whole project must be under 50kb in total size to reduce server bandwidth. So, we must keep our code concise. No one file can be over 10kb either.
* Must involve inheritance and the usage of Methods from other classes to function.
* Must have Database connectivity and the ability to query the SQL Database Tables.
* Have two separate GUI tabs, one for characters and one for places.

Here are some extra notes I made, just to improve documentation, prove the thinking and level of thought I have put into this project.

|  |
| --- |
|  |

**Task2: Design – Class Diagram**

**Your class diagram should have associated classes and the tester class. This class diagram must be clearly drawn showing all the class's attributes, constructors, and methods (see class diagram in sample scenario). Complete UML diagrams for your project so that you can start working on implementation of your project.**

**UML:**

|  |
| --- |
|  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Task 3: Implementation: Write Java code (Observational Task)  *No Submission of the Task required as Students are going to demonstrate the Tasks within the class in Week 15 & Week 16. Teacher will mark your work through this checklist.*  *Encapsulation*   * All the classes stored under a package named <stage2\_StudentName> and saved under the project named <Project\_Student\_Name> * You need to create at least one graphical user interface (GUI) userInterface class with two tabs as explained in scenario. * There must be a database class(db.java) to display data from the database table. * Your classes must be properly encapsulated. This means that there is to be no input or output from your classes other than via your User Interface class. * All attributes must be declared private. | |  |  |  |
| *Checklist for the Project (Stage 2)* | | Satisfactory | Not Satisfactory | Comments |
| **Task 3: Implementation: Write Java code**  Encapsulation:  All the classes stored under a package named <stage2\_StudentName> and saved under the project named <Project\_Student\_Name> *and demonstrate all the Tasks as below during Week 15 and Week 16 to get Satisfactory.*  ***No Submission of the Task required as Students are going to demonstrate the Tasks within the class. Teacher will mark your work through this checklist.*** | |  |  |  |
| 1 | Graphical user interface (GUI) user Interface class with two tabs |  |  |  |
| 2 | User Interface containing the buttons for different options (e.g. add,delete,save,display etc.) for the user. |  |  |  |
| 3 | Database class(db.java) to display data from the database table. |  |  |  |
| 4 | Your classes must be properly encapsulated. This means that there is to be no input or output from your classes other than via your User Interface class. |  |  |  |
| 5 | All attributes must be declared private. |  |  |  |
| 6 | JavaDoc. |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | |  |  |  |
| *Checklist for the Project (Stage 2)* | | Satisfactory | Not Satisfactory | Comments |
| **Task 4: Testing and Functionality**  Encapsulation:  All the classes stored under a package named <stage2\_StudentName> and saved under the project named <Project\_Student\_Name> *and demonstrate all the Tasks as below during Week 15 and Week 16 to get Satisfactory.*  ***No Submission of the Task required as Students are going to demonstrate the Tasks within the class. Teacher will mark your work through this checklist.*** | |  |  |  |
| 1 | Java code must be created which aligns with the design aspect done in the UML Class diagram. |  |  |  |
| 2 | No Java syntax errors  (you should have used a debugging tool for debugging) |  |  |  |
| 3 | Comments Included |  |  |  |
| 4 | Tester/driver class  (contains the main () method) |  |  |  |
| 5 | Create buttons on the user interface for showing options for the user. One option would be to input data for your objects and add them to the ArrayList, and another would be to display the data of all the objects from the ArrayList onto JtextArea. |  |  |  |
| 6 | Both parent and child classes should have: two constructors getter (accessor) and setter (mutator) methods for each attribute; and, a toString() method |  |  |  |

**Task 5: Completion – finalise the app for presentation to client**

* You are to review your completed app by checking whether all user requirements have been met, and make any final changes, as necessary. Perform and document two-unit test cases. (Provide the Screenshots)
* Complete the project by handing the app (source code should include proper comments and javadoc documentation file) over to the client (class teacher). Ensure the app is approved before submitting for assessment.

|  |
| --- |
|  |
|  |

The problems I faced in the creation of this project were:

* Packaging errors
* Programming Environment errors / quirks
* Variable matching data types
* Inheritance
* Array accessibility
* Object / Method accessibility
* Static Method errors
* Saving objects to an external text document
* Migration of Database from Main Class to separate Database Class.
* Connecting rest of the program with Database.
* Confirming that Databases Queries are working.

**User Guide:**

* Add a new character object through inputting the following values:
  + -characterId (int)
  + -Name (str)
  + -Type (str)
  + -HP (int)
  + -Mana (int)
  + -Attack (int)
  + -Speed (int)

If you have navigated off of the character tab panel you will need to reselect it to go back.

* Click List Characters to list all known characters created in this session.
* Click Test Database to save preselected characters to the external database table of characters.
* Select the place tab panel to add a new place object through inputting the following values:
  + -Location (str)
  + -Size (str)
  + -Biome (str)
* Click List Places to list all known places created in this session.
* Click Test Database to save preselected places to the external database table of places.
* Exit the Program by clicking the x in the top right or use any of the other toolbar utilities. OR end the program on the Programming Tool you are using.

**Known Errors:**

* Error if entering invalid data type inputs into character or place instance variables while creating an object. (Integer instead of string or vice versa).

**References:**

Nisha. (2022). Apply Intermediate Object Orientated Programming [Canvas]. RMIT.Mosh Courses

Khan Academy. (2022).

Greater Internet (from previous research & learning)

**Appendices:**

1. javaReport2OISINS3952320.docx (this file)
2. characterCreator folder
3. Javadoc folder
4. UML Diagram
5. Screenshots

**End.**