**CARDIFF SCHOOL OF MANAGEMENT: ASSIGNMENT FEEDBACK PROFORMA**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| STUDENT NAME: | | | | | PROGRAMME: Computing, Software Engineering, BIS | | | |
| STUDENT NUMBER: | | | | | YEAR: 2017-18 | | GROUP: Comp & IS | |
| Module Number: BCO6008 | Term: 1 | | | | Module Title: Object Oriented Systems 2 | | | |
| Tutor Responsible For Marking This Assignment: Paul Angel | | | | | | | | |
| Module Leader: Paul Angel | | | | | | | | |
| Assignment Due Date: 15/12/17 | | | Hand In Date: 15/12/17 | | | | | |
| ASSIGNMENT TITLE: Development of a Historical Timeline Application using Object Oriented Techniques | | | | | | | | |
| **SECTION A: SELF ASSESSMENT (TO BE COMPLETED BY THE STUDENT)** | | | | | | | | |
| **In relation to each of the set assessment criteria, please identify the areas in which you feel you have strengths and those in which you need to improve. Provide evidence to support your self-assessment with reference to the content of your assignment.** | | | | | | | | |
| STRENGTHS | | | | AREAS FOR IMPROVEMENT | | | | |
| I certify that this assignment is a result of my own work and that all sources have been acknowledged:  Signed:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | | | | | | | |
| SECTION B: TUTOR FEEDBACK **(based on assignment criteria, key skills and where appropriate, reference to professional standards)** | | | | | | | | |
| STRENGTHS | | | | AREAS FOR IMPROVEMENT AND TARGETS FOR FUTURE ASSIGNMENTS | | | | |
| MARK/GRADE AWARDED | | DATE: | | | | SIGNED | | |
| ASSIGNMENT MODERATED BY: | | | | | | | | DATE |
| MODERATOR’S COMMENTS: | | | | | | | | |

# Assignment

This assignment is worth 100% of the total assessment for this module.

1. **Learning Outcomes and Key Skills**

This assignment addresses the following Learning Outcomes of the module…

* Synthesise the methods of object oriented (OO) analysis and design using an appropriate notation.
* Critically evaluate and choose from a range of approaches to developing OO software.
* Critically evaluate alternative OO designs as potential solutions to particular software requirements.

This assignment also addresses the following Key Skills…

* Written Communication
* Use of IT

1. **Description of Task and Guidance Notes**

You have been asked to develop a Java-based application that allows the user to setup and view historical timelines. The application should support the following features…

* Show the timeline graphically, and allow the user to scroll forwards and backwards through the timeline.
* Allow the user to add new events to the timeline. Each event should contain at least a date, title and description. You can add additional information such as images or links to images or webpages with more information should you wish to. More marks are awarded for this!
* Allow the user to maintain a list of historical figures that can be assigned to the events you create. Aside from the link between the historical figure and the event, additional information about their involvement can also be stored.
* Allow the user to save and load the timeline that has been created. This can be done using a suitable data format (XML, JSON etc.) but you must justify the chosen format in the design document.

**Hint #1**: Do an image search in Bing or Google to get some ideas of what timelines can look like, but don’t be overwhelmed by some of the more elaborate ones!

**Hint #2**: The GUI layout, visual style and on-screen controls are left to your imagination. The GUI library to be used is also left to your discretion, but you must justify your choice in the design document.

**Hint #3**: **Don’t be afraid to get feedback from your tutor on your design, or make changes as you progress with the assignment. If you do make changes, keep track of each version of your design and include them as an appendix in your design document.** *We’re interested in seeing your progress through the development of the assignment. You’ll not be penalised for making mistakes in an early design that are later identified and corrected!*

1. **Assessment Criteria**

Design Document (50%)

* Use-Case diagram (5%)
* Class diagram – this should encapsulate ALL functionality of the application (15%)
* Behavioural diagram for your chosen scenario (8%)
* GUI Design / Layout (5%)
* Identification of relevant design patterns and where they apply in the design (5%)
* Justification of design decisions (i.e. chosen behavioural diagram and scenario to be modelled; chosen GUI library; data format for storing timeline) (6)
* Test plan and results (6%)

Implementation (50%)

* Implementation of classes identified in the class diagram. Marks are awarded for completeness (properties and methods) and correctness (methods working correctly, correct use of data types to model associations for example). (25%)
* GUI Implementation using chosen GUI library (15%)
* Derived Javadoc documentation (documentation of code) (10%)

1. **Submission Details**

Submit your assignment as a zip file on Moodle no later than the hand-in date shown on the front sheet of the assignment. **Please name your zip file with your student number (e.g. st12345678.zip)**. Your zip file should contain your implementation and design documentation.

1. **Referencing Requirements**

All referencing should use the Harvard Referencing Style. **Any images used in the implementation should also be cited and included in an appendix in the design document.**

1. **Marking Scheme**

|  |  |
| --- | --- |
| **Mark Range** | **Criteria** |
| 1. **– 100** | An excellent design is given that covers all the requirements of the assignment and correctly uses UML notation throughout. A very good justification of the chosen diagrams is also given. An excellent implementation is given that is both efficient and implements all aspects of the design. Few to no errors are evident in the code and additional elements have been included to good effect. A very detailed test plan is given that covers all use-cases. A very detailed analysis / discussion of the results obtained is given that discusses why the results were obtained, demonstrating a very good understanding of the code. |
| 1. **– 69** | A very good design is given that covers all the requirements of the assignment, though some notational errors are evident. A good justification of the chosen diagrams is also given. A good implementation is given that closely reflects the given design. Few to no errors are evident in the code but some aspects could be implemented more efficiently. A very good test plan is given that covers a good range of use-cases. A good analysis / discussion of the results obtained is given that discusses to some extent why the given results were obtained, demonstrating a good understanding of the code. |
| 1. **– 59** | A good design is given that covers the key requirements of the assignment but this could be expanded. A good implementation is given that closely reflects the design, but a number of omissions and errors are evident in the code. The implemented code could also be more efficient. A good test plan is given that covers a range of use-cases. Some analysis of the results is given, but this is largely descriptive and needs to be expanded. |
| 1. **– 49** | A basic design is given but does not address all of the requirements of the assignment. A basic implementation is given but this does not implement all aspects of the design and contains a number of errors and inefficiencies throughout. A test plan is given that covers the basic use-cases of the application. This needs to be significantly expanded. Some analysis of the results obtained is given but this is descriptive in nature. More discussion of *why* the given results were obtained is needed. |
| 1. **– 39** | A very basic design is given that does not meet all of the requirements of the assignment and does not convey enough information to implement the solution. A basic implementation is given, but this does not implement all of the required features and contains numerous errors and inefficiencies throughout. Only a very basic test plan and analysis of the results is given. |
| **Under 35** | No meaningful design has been given and little to no code has been implemented. There is no evidence of any meaningful testing or analysis of the results obtained. Little to no understanding of the problem and the techniques required to create a solution are evident. |