|  |  |
| --- | --- |
| Submission Deadline | Marks and Feedback |
| Before 10am on:  15/11/2019 | **20 working days after deadline (L4, 5 and 7) 15 working days after deadline (L6) 10 working days after deadline (block delivery)**  13/12/2019 |





|  |  |
| --- | --- |
| Unit title & code | Object-Oriented Programming and Software Engineering |
| Assignment number and title | Assignment 1: Control an Elevator - A C# Project |
| Assessment type | WR-I (Individual Coursework) |
| Weighting of assessment | 30% |
| Unit learning outcomes | 1. Demonstrate the following knowledge and understanding  • Systematically categorize the concepts, principles and techniques for object-oriented programming and software engineering process.  2. Demonstrate the following skills and abilities  • Analyse a complex problem and apply a programming language to design, develop and critically evaluate a software solution to the given problem; and analyse and apply a range of concepts, principles and techniques to the software development process. |





|  |
| --- |
| What am I required to do in this assignment? |
| Aim This assignment aims to develop students’ capabilities and skills of solving real-world problems with C# programming language.    A simple version of the elevator is illustrated below:   Task description: **Task 1:** To create a GUI which contains **(30 marks)**   * Two request buttons corresponding to the two floors, respectively * One control panel with two buttons and a display window * Two display areas that display the status of the elevator, i.e. which floor the elevator currently stays, one is of each floor * A log button, which trigger the historical information of the elevator operation to be displayed   **Task 2:** To create a control program that processes the events published by the GUI. **(10 marks)** That is,   * When any request button is pressed,   + the elevator appears in the corresponding floor   + then the display areas and the display window on the control panel show the corresponding floor number at the same time * When the floor number buttons on the control panel are pressed,   + the elevator appears to the corresponding floor   + then the display areas and the display window on the control panel show the corresponding floor number at the same time   **Task 3:** To create a log which record all operations. **(20 marks)** That is,   * The corresponding status of the elevator and the time information is stored in a database (it can be MS Access or other equivalent databases) * To display the information which has been stored in the database on the GUI   **Task 4:** To animate the events described in Task 2 using delegation and timer **(10 marks)**  **Task 5:** To optimise the programs developed in Tasks 1 to 4. **(30 marks)** That is,   * 5.1 Optimise the portability by using relative path instead of absolute path * 5.2 Optimise the maintainability by avoiding any duplication among the event handlers over the database related functions * 5.3 Optimise the robustness by eliminating logical errors and handling exception * 5.4 Optimise the efficiency of GUI by implementing multiple tasks concurrently via BackgroundWorker - animation * 5.5 The elevator might have more than two floors, apply state design pattern in the program to accommodate future changes of the requirement.  Submission: What to submit – ***Testing*** ***Report*** and ***The Entire C# Project Code***    **Testing Report**:  Name your submission – Your submission should be named with the following format: studentIDAssignmentOneReport.doc  Where to submit –Under the assignment 1 folder within the Assessment menu of this Unit’s BREO site, called “Assignment One – Report”  The testing report should include the “Marking Matrix with Self-Assessment” table  ***The Entire C# Project Code:***  The entire C# project including all folders and files, and the DB which should be saved in bin->debug folder and saved as .zip file  Name your submission – Your submission should be named with the following format: studentIDAssignmentOne.zip  Where to submit – Under the assignment 1 folder in the Assessment menu of this Unit’s BREO site, called “Assignment One C# Project Code Here” |
| Is there a size limit? |
| There is no limit on the size of the source code.  The testing report should not exceed 3000 words (source code in the report is not counted as the part of the work count). |
| What do I need to do to pass? (Threshold Expectations from UIF) |
| Design and produce a demonstrable artefact using C# within the .NET framework using the object-oriented paradigm. Check the marking matrix at the end of the brief for more details. |
| How do I produce high quality work that merits a good grade? |
| Detailed in the marking matrix at the end of the brief. |
| How does assignment relate to what we are doing in scheduled sessions? |
| Implementation of the knowledge and skills delivered from Week 1 to Week 6. |



|  |
| --- |
| How will my assignment be marked? |
| Your assignment be marked according to the threshold expectations and the criteria on the following page.  You can use them to evaluate your own work and estimate your grade before you submit. |

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Lower 2nd – 50-59%** | **Upper 2nd – 60-69%** | **1st Class – 70%+** |

Marking Matrix with Self-Assessment

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Task Number** | **Sub-tasks** | **Possible Marks** | **Self-assessment (completed Yes/No)** | **Reference to your testing report** | **Mark Awarded** |
| **Task 1** | Complete GUI for Task 1 | 20 |  |  |  |
| Skeleton of event handlers in place for all buttons | 10 |  |  |  |
| **Task 2** | All event handlers are functional | 10 |  |  |  |
| **Task 3** | Database (DB) is designed and can be connected | 5 |  |  |  |
| Log Information can be retrieved from DB and displayed in the GUI | 5 |  |  |  |
| When the log button is pressed, log information is sent to and stored in the DB | 5 |  |  |  |
| Data source is updated via DataAdapters Update() method instead of ExecuteNonQuery() method | 5 |  |  |  |
| **Task 4** | Events described in Task 2 animated using delegation and timer | 10 |  |  |  |
| **Task 5** | Using relative path instead of absolute path | 5 |  |  |  |
| Avoiding any duplication among the event handlers over the database related functions | 5 |  |  |  |
| Eliminating logical errors and handling exceptions with try and catch | 5 |  |  |  |
| Optimise the efficiency of GUI by implementing multiple tasks concurrently via BackgroundWorker | 5 |  |  |  |
| Use state patterns instead of if-else statements to accommodate future changes of the requirement | 10 |  |  |  |
| **Total** |  | 100 |  |  |  |