****

# Bachelor of Information Communications Technology

# IT8118: Advanced Programming

# Project Design Semester 2, 2018/19

|  |  |  |  |
| --- | --- | --- | --- |
|  | |  | |
| **Group** |  | **Student Name** | **Student ID** |
| **Group Members** | | 201400603 | Ali Almoallem |
| 201400303 | Mohamed Hasan |
| 201501387 | Abdulhameed Zainal |
|  |  |

**Note**: This document is yours to keep. Please review the marking guide below to understand where you gained and lost marks. Please pay close attention to any comments inserted by your marker

|  |
| --- |
| Overall Comment |
| Your score: 10.9/15 = 72.67%. A very good attempt which is largely correct. Only one method was off target. Be careful not to try to work with more parameters than required. Review the target solutions below and compare with what you wrote. |

**Deliverable 1: Code Design Document for Class Library: Marking Guide and Student Feedback Document**

**Note:** This target solution as described is probably the most efficient to meet the required functionality using the fewest number of new methods. Equivalent functionality if described correctly will get full marks, however marks may be deducted later regarding efficiency of the solution, e.g. if multiple methods are described when one method would be sufficient, or the methods as described are unnecessarily complex.

**Note**: The shaded areas showing no marks are completed components which are provided to students in the template.

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Target Solution** | **Max** | **Grade** |
| **Name** | GetMaxID (or similar) |  |  |
| **Description** | A method in the DataList class which finds the current maximum id value in the table and returns that value, or returns the next value |  |  |
| **Rationale** | Because the primary key field in the Course table is not autoincrement type, the application needs to provide a new unique ID value when new rows are added. | 3 | 3 |
| **Parameters** | None |  |  |
| **Return type** | int | 1 | 1 |
| **Total** | | 4 | 4 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Target Solution** | **Max** | **Grade** |
| **Name** | TotalValue (or similar) |  |  |
| **Description** | A method in the DataList class which calculates the total of a column | 2 | 2 |
| **Rationale** | Needed to calculate the total of all scheduled hours |  |  |
| **Parameters** | *string* column |  |  |
| **Return type** | int | 1 | 1 |
| **Total** | | 3 | 3 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Target Solution** | **Max** | **Grade** |
| **Name** | TotalValue (or similar) |  |  |
| **Description** | A method in the DataList class which calculates the total of a column where another column is equal to a value. Overloaded version of previous TotalValue method |  |  |
| **Rationale** | Needed to calculate the total scheduled hours for a Section or a Location | 4 | 2 |
| **Parameters** | *String* sumColumn*, string* column, *string* value |  |  |
| **Return type** | int | 1 | 1 |
| **Total** | | 5 | 3 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Target Solution** | **Max** | **Grade** |
| **Name** | TotalValue (or similar) |  |  |
| **Description** | A method in the DataList class which calculates the total of a column where a column in a related table is equal to a value . Overloaded version of previous TotalValue method | 5 | 4 |
| **Rationale** | Needed to calculate the total scheduled hours for an Instructor |  |  |
| **Parameters** | *String* sumColumn*, string* table1*, string ke*y*1, string* key2*, string* column*,*  string value | 3 | 2.5 |
| **Return type** | int | 1 | 1 |
| **Total** | | 9 | 7.5 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Target Solution** | **Max** | **Grade** |
| **Name** | TotalValue (or similar) |  |  |
| **Description** | A method in the DataList class which calculates the total of a column where a column in a related table is equal to a value, and the related table needs to be joined via another related table. Overloaded version of previous TotalValue method | 8 | 7 |
| **Rationale** | Needed to calculate the total scheduled hours for a Course or a Student. Can also be used to apply the rule that a Student should not be scheduled for more than 20 hours | 4 | 2 |
| **Parameters** | *String* sumColumn*, string* table1*, string ke*y*1, string* key2*, string* *table2*, *string* *key3*, *string* *key4, string* column*,*  string value | 4.5 | 3.5 |
| **Return type** | int | 1 | 1 |
| **Total** | | 17.5 | 13.5 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Target Solution** | **Max** | **Grade** |
| **Name** | AverageValue (or similar) |  |  |
| **Description** | A method in the DataList class which calculates the average of a column |  |  |
| **Rationale** | Needed to calculate the average of all grades | 2 | 2 |
| **Parameters** | *string* column | 0.5 | 0.5 |
| **Return type** | double | 1 | 1 |
| **Total** | | 3.5 | 3.5 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Target Solution** | **Max** | **Grade** |
| **Name** | AverageValue (or similar) |  |  |
| **Description** | A method in the DataList class which calculates the average of a column where another column is equal to a value . Overloaded version of previous AverageValue method | 6 | 6 |
| **Rationale** | Needed to calculate the average grade for a Section or Student |  |  |
| **Parameters** | *string* sumColumn, *string* column, *string* value | 1.5 | 1.5 |
| **Return type** | double | 1 | 1 |
| **Total** | | 8.5 | 8.5 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Target Solution** | **Max** | **Grade** |
| **Name** | AverageValue (or similar) |  |  |
| **Description** | A method in the DataList class which calculates the average of a column where a column in a related table is equal to a value, and the related table needs to be joined via another related table. Overloaded version of previous AverageValue method | 8 | 8 |
| **Rationale** | Needed to calculate the average grade for a Course | 3 | 3 |
| **Parameters** | *String* sumColumn*, string* table1*, string ke*y*1, string* key2*, string* table2, *string* key3, *string* key4,*string* column*,*  string value | 4.5 | 3.5 |
| **Return type** | double | 1 | 1 |
| **Total** | | 16.5 | 15.5 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Target Solution** | **Max** | **Grade** |
| **Name** | Exists (or similar) |  |  |
| **Description** | A method in the DataList class which selects two columns which are equal to two values where another column is equal to a value, and returns true if any rows are found |  |  |
| **Rationale** | Needed to determine if a Location is already scheduled for a particular day and time |  |  |
| **Parameters** | *string* column1*, string* value1*, string column2, string* value2*, string* column*,*  string value |  |  |
| **Return type** | bool | 1 | 1 |
| **Total** | | 1 | 1 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Target Solution** | **Max** | **Grade** |
| **Name** | Exists (or similar) |  |  |
| **Description** | A method in the DataList class which selects two columns which are equal to two values where a column in a related table is equal to a value, and returns true if any rows are found. Overloaded version of previous Exists method | 7 | 6 |
| **Rationale** | Needed to determine if an Instructor is already scheduled for a particular day and time |  |  |
| **Parameters** | *string* table1*, string* key1*, string* key2*,* *String* column1*, string* value1*, string* column2*, string* value2*, string* column*,*  string value | 4.5 | 3.5 |
| **Return type** | bool | 2 | 2 |
| **Total** | | 13.5 | 11.5 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Target Solution** | **Max** | **Grade** |
| **Name** | Exists (or similar) |  |  |
| **Description** | A method in the DataList class which selects two columns which are equal to two values where a column in a related table is equal to a value, and the related table needs to be joined via another related table, and returns true if any rows are found. Overloaded version of previous Exists method | 8.5 | 0 |
| **Rationale** | Needed to determine if a Student is already scheduled for a particular day and time | 4 | 0 |
| **Parameters** | *string* table1*, string* key1*, string* key2*,* *string* table2*, string* key3*, string* key4*,* *string* column1*, string* value1*, string* column2*, string* value2*, string* column*,*  string value | 6 | 1.5 |
| **Return type** | bool | 2 | 2 |
| **Total** | | 20.5 | 3.5 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Target Solution** | **Max** | **Grade** |
| **Name** | Delete (or similar) |  |  |
| **Description** | A method in the DataList class which deletes all rows from a table where a column is equal to a value  Overloaded version of existing Delete method. | 5 | 3 |
| **Rationale** | Needed for deleting a Student. As cascading deletes are not implemented in the database, an attempt to delete a Student with related records in the SectionStudent table will throw an exception. First the related records in the SectionStudent table will need to be deleted using this method. Then the Student can be deleted |  |  |
| **Parameters** | *string* column, *string* value | 1 | 1 |
| **Return type** | None | 1 | 1 |
| **Total** | | 7 | 5 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Target Solution** | **Max** | **Grade** |
| **Name** | Delete (or similar) |  |  |
| **Description** | A method in the DataList class which deletes the child records from another table based on a foreign key. The code uses Delete SQL syntax with an Inner Join and the primary and foreign key.  Overloaded version of existing Delete method. | 8.5 | 4 |
| **Rationale** | Needed for the process of deleting an Instructor or TaughtCourse. As cascading deletes are not implemented in the database, an attempt to delete an Instructor with related records in the Section, Schedule or SectionStudent tables will throw an exception. First the Schedule rows will need to be deleted for an Instructor using this method. Then the SectionStudent rows will need to be deleted for the Instructor using this method. Then the Sections for the Instructor need to be deleted from the Section table using this method or the previous method. Then the Instructor can be deleted. A similar process needs to be followed when deleting a TaughtCourse | 9 | 9 |
| **Parameters** | *string* table2, *string* key1, *string* key2, *string* column, *string* value | 2.5 | 1.5 |
| **Return type** | None | 1 | 1 |
| **Total** | | 21 | 15.5 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Code Quality** | **Rubric** | **Max** | **Grade** |
| **Overall efficiency of described solution** | Deduct marks for reduced efficiency of the solution;  if multiple methods are described when one method would be sufficient (-10) , or the methods as described are unnecessarily complex. (-10) | 20 | 14 |
| **Total** | | 20 | 14 |
| **Grand Total** | | 150 | 109 |
| 15 | 10.9 |