ASSESSMENT AND INTERNAL VERIFICATION FRONT SHEET (Individual Criteria)

(Note: This version is to be used for an assignment brief issued to students via Classter)

Course Title	**B.Sc. (Hons.) Software Development**		**Lecturer Name & Surname**	**Ryan Attard David Debono Laurent Azzopardi**	
Unit Number & Title	ITSFT-506-2006 | Object Oriented Programming				
Assignment Number, Title / Type	2, Implementing an Object Oriented based application / Home				
Date Set	16/11/2023	**Deadline Date**	8/1/2024		
Student Name	Lucas Said	**ID Number**	26005H	**Class / Group**	6.1A

Assessment Criteria	Maximum Mark
AA1.4: Implement both static and dynamic polymorphism in an Object-Oriented application.	7
KU3.1: Show ability to efficiently create and use the storage layer of a layered application.	5
KU3.2: Describe the use of LINQ queries in an Object-Oriented application.	5
AA3.3: Use LINQ to implement persistent data storage in an object-oriented application.	7
SE3.4: Assess an application and break it down into different layers to identify where object-database mapped classes should be inserted.	10
AA4.2: Implement exception handling processes to respond to any exceptions occurring in an application.	7
SE4.3: Analyse an implementation of an Object-Oriented application.	10
Total Mark	51

Notes to Students:

- This assignment brief has been approved and released by the Internal Verifier through Classter
- Assessment marks and feedback by the lecturer will be available online via Classter (Http://mcast.classter.com) following release by the Internal Verifier
- Students submitting their assignment on Moodle/Turnitin will be requested to confirm online the following statements:

Student's declaration prior to handing-in of assignment

- ❖ I certify that the work submitted for this assignment is my own and that I have read and understood the respective Plagiarism Policy

Student's declaration on assessment special arrangements

- ❖ I certify that adequate support was given to me during the assignment through the Institute and/or the Inclusive Education Unit.
- ❖ I declare that I refused the special support offered by the Institute.

Assignment Guidelines

Read the following instructions carefully:

- The assignment coversheet should be the first sheet in your assignment. Moreover, the coversheet should be fully completed with all the necessary details.
- You are required to use the .NET Framework/.NET CORE and the C# programming language for this assignment.
- **Copying or working in pairs/groups is strictly prohibited and will be penalized** in line with the College's disciplinary procedures.
- You need to submit your application via a link on VLE. Ensure that you submit via the correct group link. You are also required to submit your work on Github. Ensure that you assign your lecturer as a collaborator to your assignment repository. Also, please make sure that the assignment repository has been set as **private**.
- You need to submit the .Net project, the script representing your database and a word document with all the code used in your application.
- The lecturer will hold a post-submission interview. Attendance to such interview is mandatory. Moreover, marks assigned to all criteria can be affected by the interview performance.
- AI generated material can be used BUT you need to make it yours by integrating it properly and make sure that the feature being implemented runs properly according to the requirements. Your lecturer will not be expected to refine/reconfigure/re-assess any code so if the code presented does not work OR you fail to explain what's happening OR errors are raised during the interview you will lose the marks associated with the respective feature.

Scenario: Battleship

Th aim of this assignment is to develop the well-known Battleship game.

You need to ensure that the application uses the most efficient **Object-Oriented Techniques**, is structured in a **three-tier architecture,** and has proper **Object-Relational mapping (ORM)** to allow for efficient persistence.

Battleship is a 2 player game and each of the players has a fleet of 5 ships which have to be placed on a grid of 8 by 7 squares/holes. Each player will take his/her turn to launch an attack by guessing the opponent ships' position on the grid. The first who manages to hit all the opponents' ships will win. A summary of the game rules can be viewed using the following link https://www.hasbro.com/common/instruct/battleship.pdf

The application should have the following persistent types. You are to use a Database First ORM approach by first implementing the database and then mapping your database to an object-oriented application. The following database structure must be implemented.

Players

Field	**Type**
Username	Nvarchar(50)
Password	Nvarchar(50)
ID	Int [optional: if used as foreign key or else you may use Username as foreign key]

Games

Field	**Type**
ID	Int [auto]
Title	Nvarchar(50)
CreatorFK	Nvarchar(50)
OpponentFK	Nvarchar(50)
Complete	Boolean
Title	Nvarchar(50)

Attacks

Field	**Type**
ID	Int [auto]
Coordinate (See footnote 1)	Nvarchar(50) or int
Hit	Boolean
GameFK	Int

Ships

Field	**Type**
ID	Int [auto]
Title	Nvarchar(50)
Size	Int

GameShipConfigurations

Field	**Type**
ID	Int [auto]
PlayerFK	Nvarchar(50)
GameFK	Int
Coordinate[1]	Nvarchar(50)or Int

The following menu should be presented to the end user:

1. Add Player details
2. Configure Ships

3. Launch Attack
4. Quit

SE3.4: Assess an application and break it down into different layers to identify where object/database mapped classes should be inserted.

Divide the above program into multiple tiers

a) Data Layer composed of
   - the Database – this should show correct and properly structured tables [1 mark]
   - the Models – the above classes abstracted into an ERM
     [1 mark]

b) Data Access Layer/Business Logic – Classes which contain only LINQ code to be able to read or write data from and to the database
   [1 mark]

c) Presentation Layer – Console application or any other application you feel comfortable working with
   [1 mark]

d) Connection string in config/settings file
   [1 mark]

Note: Points are given if the project exists and has correct project references to fit this architecture.

[1] This may be changed to make the process easy as possible. Instead of letters you may label each of the cells in the table with a unique number so you avoid Letters and Numbers and an eventual parsing of the input.

The following tasks describe the methods to be implemented. Points will be reduced if the 3-tier architecture is not respected. Violation of the rules will result in a deduction of 1 point for each method in the incorrect class or tier. LINQ code must reside in the DataAccess layer, Console.WriteLine or ReadLine statements should be placed in the Presentation Layer and models should be placed in a separate project/folder.

KU3.1: Show ability to efficiently create and use the storage layer of a layered application.

Implement menu option1

a) Add Player details: The application should ask the two players to input their details and save them in the Players table. [1].
b) If the username inputted exists, the application should ask the player to input the password (to confirm identity) [1]
c) Both users should then be set as the players for the game by saving the details in the Games table including a title for the Game [2]
d) Note: Password input or verification should be masked with asterisks [1]

AA3.3: Use LINQ to implement persistent data storage in an object-oriented application.

Each player has to configure on the game grid, 5 ships.

Determine a way how to ask both players that information and implement menu 2.
Note: It should be clearly visible on screen who you are asking the information from i.e Player 1 or Player 2?

a) Select a ship (by id), input coordinate (you may choose a way you deem the easiest) and save in the table GameShipConfiguration. This has to be done for all the 5 ships. When ready Player 2 should configure his/her own [3]

b) Ships cannot overlap each other for the same player. Validation for clashes should be implemented and in case of an overlap the user should be asked to repeat the input [2]

c) Ships may be placed vertically or horizontally [2]

d) Note: Continue implementing this menu option in SE4.3 below.

SE4.3: Analyse an implementation of an Object-Oriented application.

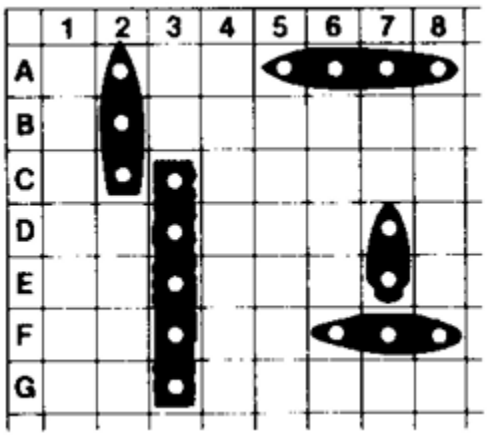

Figure 1

Figure 1 depicts how the game play should look.

Analyse how a grid can be displayed on screen and create a method within the Program.cs which allows you to display the grid everytime you need it. It does not need to be polished. Displaying just the contents of the cells (no borders) next to each other will do the job.

Two views need to be implemented:

a) [Menu option 2] A view with the player's ship configuration (loaded from the

GameShipConfiguration table) showing the ships in place on screen similar to Figure 1 [3]

b) [Menu option 3; you may complete this after Ku3.2] After the launch of an attack happens, a screen showing the (past) attacks and whether they were a hit or a miss (loaded from the Attacks table) and shown on screen with an X or a O as a grid, in case hit and missed respectively [3]
c) **Note**: If you manage to complete (a) and (b) in one method (PrintGrid) as described in AA1.4 you will get an extra 3 marks.
d) **Menu options should have constraints [1]**:
    a. **Menu option no 1 or 4 should be allowed to be chosen in the beginning**
    b. **Once players are added, menu option 2 and option 4 should only be allowed to be chosen**
    c. **Once game configuration is finalized, then only option 3 or 4 should be allowed;**

KU3.2: Describe the use of LINQ queries in an Object-Oriented application.

Implement menu option 3. It should be clearly visible on screen whose turn it is i.e Player 1 or Player 2?

Launch the attack by asking for the coordinate, check/save data via a LINQ queries

a. whether there was a hit or a miss against the GameShipConfigurations [2],
b. save the attack in the Attacks table [1]
c. check via a LINQ query that if all positive attacks amount to the total size of all the ships, then we have a winner and it should be shown. Game should be concluded by marking it complete [2].
d. See Se4.3 (b)

AA1.4: Implement both static and dynamic polymorphism in an Object-Oriented application.

In the following screenshot, sample code has been initiated for you to be able to answer this task successfully. The idea is to implement polymorphism to make the code that prints on screen more efficient.

```
1 reference
public class GameScreen
{
    0 references
    public GameScreen(List<Cell> cells)
    { }

    //this method has to go through the list of cells (e.g. cells)
    //populated/fetched from the database and it will have to call the PrintCell method of each of them
    0 references
    public void PrintGrid() { //... }
}

3 references
public abstract class Cell
{
    //common properties for both a cell showing a ship or a successful/missed attack
    //e.g. public int GridCellNo;

    //this has to be implemented differently in the next two classes shown here
    //for a cell representing a Ship or cell representing a successful attack or representing a missed attack.
    //e.g. if it is a Ship, on screen print S; if it is a missed attack on screen print O;
    //     if it is a successful attack on screen print X
    0 references
    public abstract void PrintCell();
}

0 references
public class ShipCell:Cell
{ }

0 references
public class AttackCell:Cell
{ }
```

Dynamic Polymorphism: Your task is to

a. Implement the classes ShipCell and AttackCell with the overridden method PrintCell. Follow the guidance in the comments. [1]
b. Implement the method PrintGrid in such a way that it displays a neat grid on screen after calling the PrintCell of each and every cell to display. [3]

Static Polymorphism: Your task is to

c. Create an additional constructor in the GameScreen class where you can pass a *List<Attack>* OR *List<GameShipConfiguration>* and everything will still work seamlessly. Hint: Attack and GameShipConfiguration are the models used for the database, therefore you fetch all the entries in the database for that game, you forward them to the constructor, and the GameScreen constructor forms a List<Cell> mapped from the data in the database waiting to be displayed on screen [3]

AA4.2: Implement exception handling processes to respond to any exceptions occurring in an application.

The Application should be running in an infinite loop, showing clearly the respective menu everytime the end user chooses a menu option until “Quit” is selected. If you implement a GUI type of application you have to make sure that there is some kind of home screen where the user can choose to close and quit everything.
[2 marks]

Try-Catch should be applied:

- A general exception in case some unhandled exception is raised, showing the user a message and forward him/her back to the main screen where he/she sees the menu; [1 mark]
- Validations:
    - If a user inputs an out-of-range cell from the grid to attack or set up one of his/her own ships, validation should be implemented and an error shown on screen WITHOUT halting the application. User should continue from the last step. [2]

- Use LINQ to assist the player in avoiding duplicate attacks (i.e. attacks on a cell that has already been executed in same gameplay). You should warn the user before submitting the attack. [2]

Rubric

		Max	Achieved
AA1.4	• Override PrintCell • Implement PrintGrid and see that it works!! • Static polymorphism which wraps database data to cell-data	[1] [3] [3]	
KU3.1	• Save Player data • Confirm Player already exists • Save Game Data • Mask Password Input	[1] [1] [2] [1]	
KU3.2	• Hit Or Miss using LINQ • Save the attack in Attacks table • Declare winner using LINQ	[2] [1] [2]	
AA3.3	• Set up ships (x5 for 2 players) • Do not overlap – warning/validation • Ships can be horizontally or vertically set up	[3] [2] [2]	
SE3.4	• Data Layer - the Database (correct tables and relationships) • Data Layer – Models (correct models in ERM model) • Data Access – Only LINQ code in Repository classes • Presentation Layer – No LINQ code there • Connection string in settings file • Deduction of 1 point for any violation of the above "guidelines" or lack of relationships between projects. A maximum of 5 points can be deducted	 [1] [1] [1] [1] [1] [5]	
AA4.2	• Program looping endlessly until a specific menu option is chosen to quit • A general try-catch • Validation of out-of-range cells input • Validation of already played attacks	 [2] [1] [2] [2]	
SE4.3	• Printing method on screen showing ship configuration data • Printing method on screen showing attacks data • Menus Activation or Deactivation depending on entries • Static polymorphism correctly applied? AA1.4	[3] [3] [1] [3]	

Note: Functions must properly work and show the right data on screen; hence testing should be done prior to the submission not during the interview. Coding a functionality but is left detached from any constructive outcome that contributes to the game play/application /the interface or never being called will not be considered.