Algonquin College Logo

# SCHOOL OF ADVANCED TECHNOLOGY

### ICT - Applications & Programming

### Computer Engineering Technology – Computing Science



A21

Computer Science Challenge

Lab Professor / Lab Session:

[Daniel Cormier /300]

Team:

[Hetvi Patel] - Id: [041077850] / [Dhara Patel] - Id: [041081736]

CS Challenge 2: Backend Design

***This template is suggested (not mandatory) to answer A21 Specification.***

|  |  |
| --- | --- |
| **Part**  **1** | **Backend Design** |

**Note 1: Read Specification**

*This is only a suggested template. Please check instructions in the A21 specification.*

* 1. **Example UC Model**

Include a diagram or a screenshot of your operational UI for reference's sake.

A screenshot of a video game

Description automatically generated

**UC Diagram** (Replace this diagram to accommodate the actors and functionalities to be used):

A diagram of a cross

Description automatically generated

**Actors table**:

|  |  |
| --- | --- |
| **Actors** |  |
| Player | This actor represents the person playing the battleship game. |

**UC table**:

|  |  |
| --- | --- |
| **Use Cases** |  |
| Language. | This case describes how the user change the Language. We have English and French. |
| Dimensions. | This use case describes how the user can change the dimensions. We have 3 different kind of measurements followed by: 5X5, 10X10, 12X12 |
| Game. | This setup includes two functions:  The first function starts a new game while keeping the current player.  The second function loads a new game board and begins from the start. |
| Help | It will provide a demonstration on how to play the game with steps. |
| Network | It connects the user and another user like they can play as multiplayer and they can also play from different location. |
| File | This will help to save the current game like pause button. Save and New game. |

1.2. Translation Table

Translation table (example):

|  |  |  |
| --- | --- | --- |
| Location | English Title | French Translation |
| File Menu | File | Fichier |
| File Menu | Save | Enregistrer |
| Game Menu | Game | Jeux |
| Game Menu | Reset | Recommencer |
| File Menu | Network | Réseau |
| File Menu | Language | Langue |
| File Menu | Help | Aide |
| Game Menu | Lifeline | Lifeline |
| Game Menu | Redo move | Rétablir le déplacement |
| Game Menu | Send Button | Bouton d’envoi |
| Game Menu | Swap Button | Bouton d’échange |

**1.3. Example CD Solution**

🛠 To draw the diagram, you can use tools (ex: <https://app.diagrams.net/>) or desktop applications (ex: Visio / Powerpoint) or simply take photos from drawings.

Proposed Relationships

- \*\*BattleshipGame\*\*: The main class managing the game's flow.

- \*\*GameBoard\*\*: BattleshipGame has a GameBoard.

- \*\*LogPanel\*\*: BattleshipGame has a LogPanel.

- \*\*PowerButtons\*\*: BattleshipGame has PowerButtons.

- \*\*InfoPanel\*\*: BattleshipGame has an InfoPanel.

- \*\*MenuItems\*\*: BattleshipGame uses MenuItems for various actions.

- \*\*SaveGame\*\*: BattleshipGame can save/load the game state.

- \*\*OfflinePlayersName\*\*: Allows for setting offline player names.

- \*\*OnlinePlayers\*\*: Manages online player information.

- \*\*Multiplayer\*\*: Manages multiplayer capabilities and connections.

[Class] BattleshipGame

|--------------------------------------|

| + startGame(): void |

| + endGame(): void |

| + saveGame(): void |

| + loadGame(): void |

----------------------------------------

| has-a

v

[Class] GameBoard

|--------------------------------------|

| + initializeBoard(): void |

| + updateToken(int column, int playerID): void |

----------------------------------------

| has-a

v

[Class] LogPanel

|--------------------------------------|

| + logMessage(String message): void |

| + showLogs(): void |

----------------------------------------

| has-a

v

[Class] PowerButtons

|--------------------------------------|

| + useLifeline(): void |

| + redoMove(): void |

----------------------------------------

| has-a

v

[Class] InfoPanel

|--------------------------------------|

| + displayPlayerInfo(): void |

| + updateClock(): void |

| + updateTurn(int playerID): void |

----------------------------------------

| has-a

v

[Class] MenuItems

|--------------------------------------|

| + save(): void |

| + load(): void |

| + openNetworkSettings(): void |

| + changeLanguage(String lang): void |

----------------------------------------

| has-a

v

[Class] SaveGame

|--------------------------------------|

| + save(String filename): void |

| + load(String filename): void |

----------------------------------------

| used-by

v

[Class] AIBot

|--------------------------------------|

| + calculateNextMove(): void |

----------------------------------------

| used-by

v

[Class] OfflinePlayersName

|--------------------------------------|

| + setName(String playerName): void |

----------------------------------------

| used-by

v

[Class] OnlinePlayers

|--------------------------------------|

| + connectToServer(): void |

| + disconnect(): void |

----------------------------------------

| used-by

v

[Class] Multiplayer

|--------------------------------------|

| + startMultiplayerSession(): void |

| + endMultiplayerSession(): void |

----------------------------------------

```

Explanation of Relationships

- \*\*BattleshipGame\*\* is the primary class that controls the entire game.

- \*\*BattleshipGame\*\* has-a \*\*GameBoard\*\*, \*\*LogPanel\*\*, \*\*PowerButtons\*\*, \*\*InfoPanel\*\*, \*\*MenuItems\*\*, and \*\*SaveGame\*\*.

- \*\*BattleshipGame\*\* uses \*\*AIBot\*\*, \*\*OfflinePlayersName\*\*, \*\*OnlinePlayers\*\*, and \*\*Multiplayer\*\* for specific functionalities.

- \*\*GameBoard\*\*: Manages the status and state of the game board.

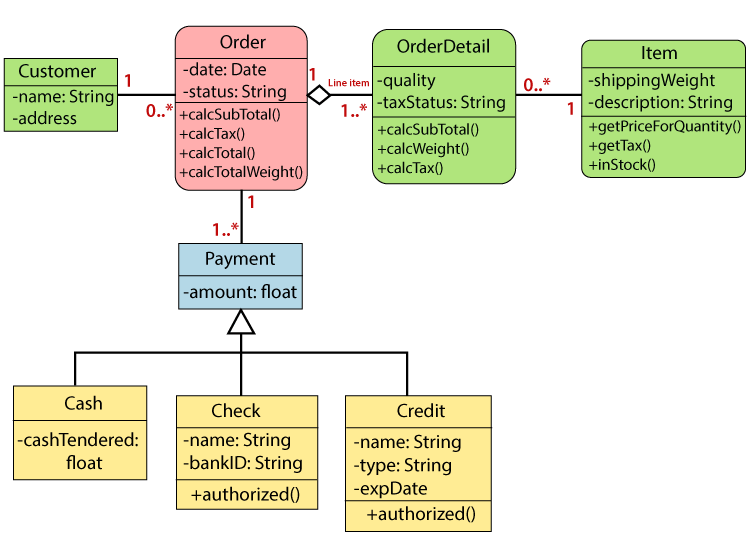
- \*\*LogPanel\*\*: Responsible for logging and displaying game-related messages.

- \*\*PowerButtons\*\*: Handles special actions a player can take during the game.

- \*\*InfoPanel\*\*: Displays player-specific information and game status.

- \*\*MenuItems\*\*: Manages menu related actions such as save, load, and network settings.

**Class Diagram** (change this diagram to accommodate the actors and functionalities to be used):



**Class table** (example):

|  |  |
| --- | --- |
| **Class name** | **Order** |
| Inner Fields | Date: Released date  Status: Order status |
| Relationships | OrderDetail: List of details id per item  Payment: List of payments id from order  Customer: Order customer id. |
| Methods | calcSubTotal: Calculates sub-total  calcTax: Total tax  calcTotal: Total order price  calcTotalWeight: Total weight from order. |

*Create tables for* ***all*** *classes.*

**Details**

*Draw the Class diagram (ex: in an image from Paint / Visio / Powerpoint slide, or any sketch tool such as umlet), describing:*

* *Class definition (properties / methods).*
* *Relationships between classes.*

**FINAL SUGGESTIONS**

*Don't just go through the motions. You have a lot of time to think about how this program will work, and the better you plan now, the easier it will be to write the code. If you just rush to complete this, you'll be making it harder on yourself when it's time to code.*

* *Try to be professional about this. A good design document is also a useful part of your portfolio, as it shows you think about the whole process.*
* *No code is required! Programming will happen in Assignment 22.*

**References**

*[Include eventual references used here]*

*Merelin AI for understanding the basics of UML diagram and Class Diagram*

|  |  |
| --- | --- |
|  | * ***NOTE****: Even if you use any AI tool (ex: ChatGPT), report here, including the references used.* |

Algonquin College

Summer, 2024