1. Project Overview

Project Name: Blazn' Barrels

Description:

Blazn' Barrels is a Java-based shooting game where the player must shoot fish hidden in a grid. The game includes three difficulty levels, keeps track of the player's high score, and features a three-round challenge system. The game uses JavaFX for the graphical interface and user interactions.

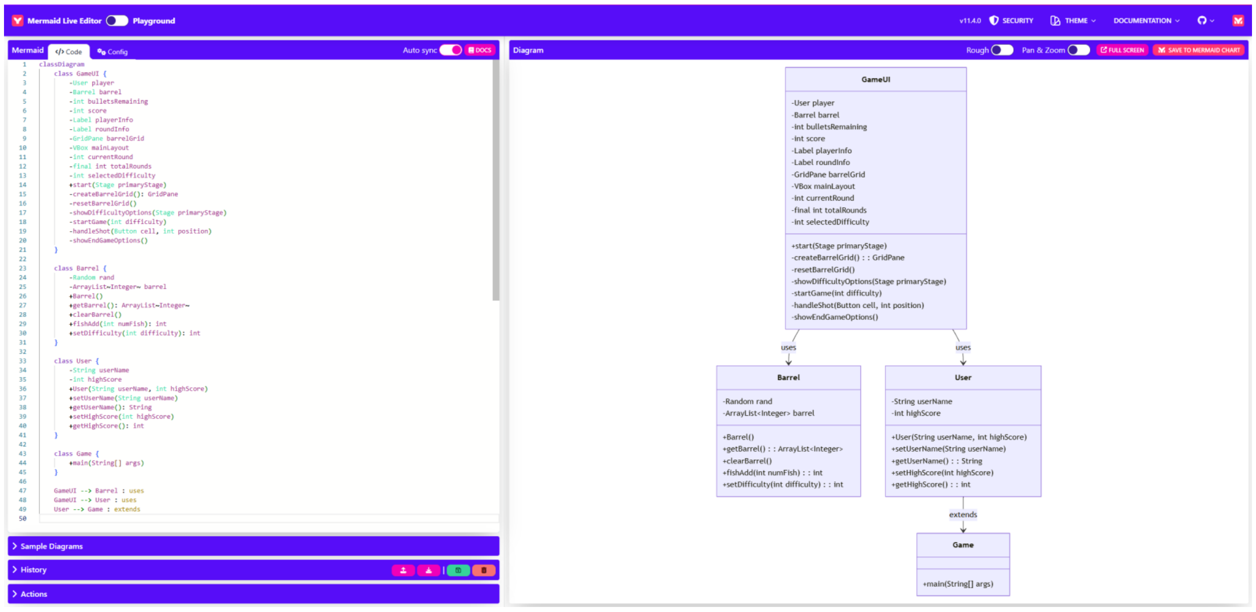
2. Class Descriptions

The following classes form the core of the game:

Class Name	Description
`GameUI`	Manages the graphical user interface, game flow, and user interactions using JavaFX.
`Barrel`	Handles fish placement, game logic, and difficulty settings.
`User`	Tracks the player's name and high score.
`Game`	Provides the terminal-based version of the game (alternate main entry point)

3. UML Class Diagram:

```
classDiagram

  class GameUI {

    -User player

    -Barrel barrel

    -int bulletsRemaining
    -int score

    -Label playerInfo

    -Label roundInfo

    -GridPane barrelGrid

    -VBox mainLayout

    -int currentRound

    -final int totalRounds

    -int selectedDifficulty

    +start(Stage primaryStage)

    -createBarrelGrid(): GridPane

    -resetBarrelGrid()

    -showDifficultyOptions(Stage primaryStage)

    -startGame(int difficulty)

    -handleShot(Button cell, int position)

    -showEndGameOptions()

}


class Barrel {

    -Random rand

    -ArrayList~Integer~ barrel

    +Barrel()

    +getBarrel(): ArrayList~Integer~
    +clearBarrel()

    +fishAdd(int numFish): int

    +setDifficulty(int difficulty): int

}


class User {

    -String userName

    -int highScore

    +User(String userName, int highScore)

    +setUserName(String userName)

    +getUserName(): String

    +setHighScore(int highScore)

    +getHighScore(): int

}


class Game {

    +main(String[] args)

}


GameUI --> Barrel : uses

GameUI --> User : uses

User --> Game : extends
```



```
classDiagram
    class GameUI {
        -User player
        -Barrel barrel
        -int bulletsRemaining
        -int score
        -Label playerInfo
        -Label roundInfo
        -GridPane barrelGrid
        -VBox mainLayout
        -int currentRound
        -final int totalRounds
        -int selectedDifficulty
        +start(Stage primaryStage)
        -createBarrelGrid(): GridPane
        -resetBarrelGrid()
        -showDifficultyOptions(Stage primaryStage)
        -startGame(int difficulty)
        -handleShot(Button cell, int position)
        -showEndGameOptions()
    }

    class Barrel {
        -Random rand
        -ArrayList~Integer~ barrel
        +Barrel()
        +getBarrel(): ArrayList~Integer~
        +clearBarrel()
        +fishAdd(int numFish): int
        +setDifficulty(int difficulty): int
    }

    class User {
        -String userName
        -int highScore
        +User(String userName, int highScore)
        +setUserName(String userName)
        +getUserName(): String
        +setHighScore(int highScore)
        +getHighScore(): int
    }

    class Game {
        +main(String[] args)
    }

    GameUI --> Barrel : uses
    GameUI --> User : uses
    User --> Game : extends
```

4. Class Details and Key Methods

4.1. GameUI (Main Class)

Description:

Manages the game's user interface and user interactions through JavaFX. It controls the game flow, including selecting difficulty, managing the game rounds, and tracking the score.

Important Methods

- `start(Stage primaryStage)`: Initializes the game UI.
- `createBarrelGrid()`: Creates the game grid.
- `resetBarrelGrid()`: Clears the grid between rounds.
- `showDifficultyOptions(Stage primaryStage)`: Displays the difficulty selection menu.
- `startGame(int difficulty)`: Starts the game based on the selected difficulty.
- `handleShot(Button cell, int position)`: Handles what happens when a player clicks a grid cell.
- `showEndGameOptions()`: Displays the game-over screen with retry and quit options.

4.2. Barrel Class

Description:

Handles fish placement, randomization, and difficulty management.

Important Methods

- `clearBarrel()`: Clears the fish positions.
- `fishAdd(int numFish)`: Adds fish to random positions.
- `setDifficulty(int difficulty)`: Sets the number of fish based on the difficulty.
- `getBarrel()`: Returns the list of fish positions.

4.3. User Class

Description:

Tracks player-specific data such as the player's name and high score.

Important Methods

- `getUserName()`: Returns the player's name.
- `setUserName(String userName)`: Sets the player's name.
- `getHighScore()`: Retrieves the high score.
- `setHighScore(int highScore)`: Updates the high score.

5. Game Flow Overview

1. **Game Start**:
    a. The player is prompted to enter their name.
    b. They select a difficulty (Easy, Medium, or Hard).
    c. The game starts with the appropriate number of bullets and fish.

2. **Gameplay**:
    a. Players click grid buttons to shoot fish.
    b. If a fish is hit, the player earns points.
    c. The game continues until the player runs out of bullets.
3. **Game End**:
    a. After all rounds are complete, the player is shown the final score.
    b. The player can either retry or quit.
4. **High Score Management**:
    a. The game tracks and displays the player's highest score.

6. Design Decisions and Challenges

6.1. Design Decisions

- **Separation of Concerns**: The game logic (`Barrel`) and UI management (`GameUI`) are kept separate.
- **Data Management**: The player's data is managed using the `User` class.
- **Replayability**: The player can retry the game after each session.

6.2. Challenges Faced

1. **JavaFX Integration**:
    a. Transitioning from a terminal-based game to a graphical JavaFX interface required event-driven logic.
2. **Grid Reset Logic**:
    a. Correctly resetting the grid after retries took extra debugging.
3. **Button Visibility Management**:
    a. Managing dynamic button visibility across game states required several layout updates.

7. Conclusion

The game "Blazn' Barrels" was successfully developed using JavaFX and core Java. The game supports multiple rounds, tracks scores, and dynamically adjusts difficulty. This project provided hands-on experience with JavaFX, object-oriented programming concepts, and GUI-based game development.