Assignment cover sheet

Please complete this cover sheet for each assignment submission.

|  |  |
| --- | --- |
| Module: | SET11102 – Database Systems |
| Assessment Title: | Software Development 1 Assignment |
| Module Leader: | |  | | --- | | John Owens | |
| Student registration number: | 40732418 |
| Student name | Oleksandr Sharha |
| Group id | TR1 001 |

**Declaration**

I declare, in accordance with [Edinburgh Napier University’s Academic Integrity Regulations](https://staff.napier.ac.uk/services/dlte/Regulations/Documents/2023-24/Academic_Integrity_2023_24_FINAL.pdf)that: except where explicit reference is made to the contribution of others\*, this assignment is the result of my own work, and has not been submitted for any module, programme or degree at Edinburgh Napier University or any other institution.

\*IMPORTANT: Contribution of includes use of generative Artificial Intelligence (AI) tools. Ensure you have read the University [Guidelines for Students on AI & Writing Assistant Tools](https://my.napier.ac.uk/your-studies/improve-your-academic-and-study-skills/referencing-and-academic-integrity/artificial-intelligence-tools)). Please declare here whether you have used such tools, and to what extent:

NO I have not used such tools

YES I have used such tools and I have provided details and included sample prompts and responses below <or: in an appendix>.

If you have used AI tools in completing this submission, please briefly describe in approximately 100 words in the box below how you have used these tools:

|  |
| --- |
|  |

**Introduction**

Report to describe the development of the Connect 3 game, a simplified adaptation of the traditional Connect Four game. The project focuses on demonstrating object-oriented programming practices taught in the SET11102 module. It was implemented by using Java programming language with Eclipse IDE.

#### **User instructions for Connect3 game**

#### **Starting the Game**

1. **Launching the Game**: Run the Connect3 Java program. Upon starting, the game will check for an existing saved game file.
2. **Continuing or Starting New Game**:
   * If a saved game is detected, you'll be prompted with "Found a saved game. Do you want to continue? (yes/no):". Respond with yes to continue from where you left off or no to start a new game.
   * If no saved game is found or you choose to start a new game, the game will proceed to initialize a new grid.

#### **Setting Up**

1. **Choosing Colors for the player**:
   * You will be prompted to choose a color for Player 1. Enter R for Red or G for Green. Player 2 will automatically be assigned the other color.
   * Example prompt: "Player 1, what color do you want to play for (R for 'RED', G for 'Green'):".
   * After typing ‘quit’ on this step the game will be finished with saving for next continue from the same position, statistic and player input turn;

#### **Playing the Game**

1. **Making a Move with stop and save game functionality**:
   * Players take turns inputting the column number where they wish to drop their disc. The columns are numbered from 1 to 6.
   * Example prompt: "Player 1 (R), your turn. Enter column (1-6):".
   * Enter the column number and press Enter. If the column is full, you will be prompted to choose another column.
2. **Game Progress**:
   * After each move, the grid will be displayed showing the current state with your disc placements.
   * Continue taking turns until a player wins by connecting three discs vertically, horizontally, or diagonally, or until the grid is full.

**Ending the Game**

1. **Winning the Game**:
   * If a player successfully connects three discs in a row, the game will display a victory message stating which player has won.
   * Example victory message: "Congratulations, Player 1 (R) wins!".
2. **Draw**:
   * If the grid is filled without any player connecting three discs, the game will announce a draw.
   * Example draw message: "The game is a draw!".
3. **Replaying or Exiting**:
   * After the game concludes by win or draw, you'll be asked if you want to play again.
   * Example prompt: "Do you want to play again? (y or yes):".
   * Respond with y or yes to start a new game or any other input to exit.
4. **Exiting During Play**:
   * At any input prompt, you can type quit to save and exit the game. The game will automatically save the current state to be resumed later.
   * Example quit execution: "Game saved. Exiting...".

#### **Additional Notes**

* The game interface is text-based, and all interactions occur through the console.
* Ensure inputs are correct as per the prompts to avoid errors during play.

This comprehensive guide should help you navigate the game effectively, ensuring an engaging and enjoyable gameplay experience. Enjoy your game of Connect 3!

**System Overview**

The game is structured around a modular object-oriented architecture, which includes several core classes:

* **Connect3 Class**: Acts as the main game controller.
* **Grid Class**: Manages the game's 6x5 grid.
* **Row Class**: Represents rows within the grid.
* **Square Class**: Represents individual cells within rows.
* **Disc Class**: Represents the discs with color attributes.

These classes interact to manage game state, enforce rules, and handle user interactions, providing a robust foundation for gameplay logic.

**Grid Representation**

The grid is implemented as an ArrayList of Row objects, where each Row contains Square objects. This design allows for dynamic access to any cell in the grid, crucial for checking win conditions and updating the game state.public Grid() {

for(int i = 0; i < GRID\_HEIGHT; i++) {

Row newRow = new Row(i);

theGrid.add(newRow);

}

}

Each Row is instantiated with its respective index, ensuring ordered access. This method is called during the initialization phase of the Grid class to set up the board.

**Visual Representation of the Grid**

The visual representation of the grid is achieved through a method that iterates over each Row and Square, outputting a text-based representation of the board state. The method handles the formatting, ensuring the grid is displayed in a user-friendly manner.

public void displayGrid() {

System.out.println("+---+---+---+---+---+---+");

for (Row row : theGrid) {

System.out.print("|");

for (Square square : row.getTheSquares()) {

System.out.print(" " + (square.isEmpty() ? " " : square.getDisc().getColourLetter()) + " |");

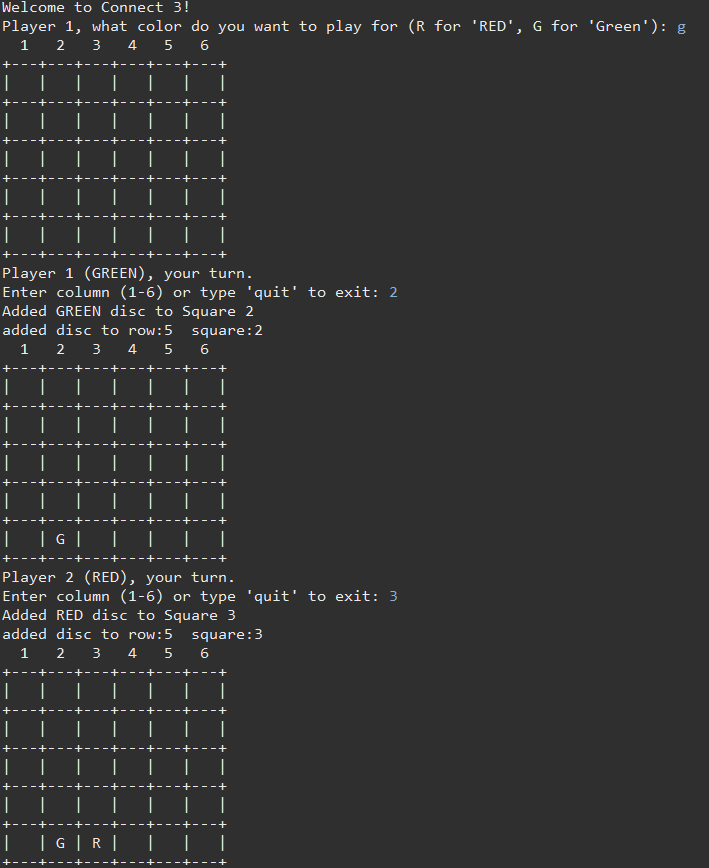
}

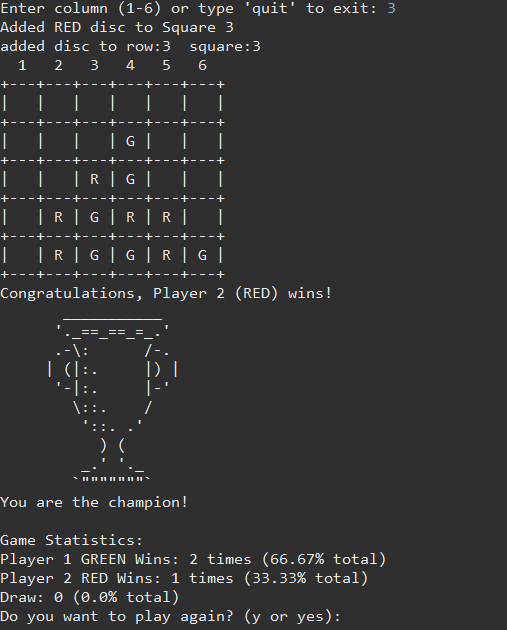
System.out.println("\n+---+---+---+---+---+---+");

}

}

This method enhances the game's usability by providing a clear and intuitive display of the current game state, allowing players to make informed decisions.





**Game Mechanics**

Gameplay is controlled by the play method in the Connect3 class, which manages the turn-taking system, processes user input, and validates moves. This method is the central loop where game state updates are triggered.

public void play() {

boolean continueGame = true;

while (continueGame) {

int column = getPlayerInput();

if (placeDisc(currentPlayer, column)) {

if (checkWinConditions()) {

endGame();

continueGame = false;

} else {

switchPlayer();

}

} else {

System.out.println("Invalid move, try again.");

}

}

}

The placeDisc method updates the grid based on the player's input, while checkWinConditions evaluates whether the current move results in a win.

**Advanced Features**

**Diagonal Win Check**: The game includes a method to check for diagonal connections, enhancing the complexity and strategic depth of the game.

**Game State Persistence**: The game supports saving and loading the state using serialization, allowing players to pause their game and resume later.

public void saveGame() {

try (ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream("save.dat"))) {

out.writeObject(this);

} catch (IOException ex) {

System.out.println("Error during game save: " + ex.getMessage());

}

}

This feature uses Java's built-in serialization mechanism to write the entire game state to a file, which can be reloaded to restore the session.

### **Advanced Features Implemented**

In the development of the Connect 3 game, several advanced programming techniques and features were employed to enhance the gameplay experience and functionality. These features extend beyond the core curriculum of the module SET11102, providing a more robust and user-friendly game environment.

#### **1. Use of Scanner for Input**

The Java Scanner class is utilized for handling user inputs throughout the game. This feature allows the program to read players' inputs from the console, enabling them to select colors, choose columns for placing discs, and decide on game options like continuing a saved game or starting anew.

* **Implementation**:
  + A Scanner object is instantiated at the beginning of the game and is used to capture user inputs in a loop until valid responses are provided.
  + This approach is critical for interactive games played in the console, as it provides an easy method for reading and processing user input in real-time.

#### **2. Save and Load Game Functionality**

The ability to save and load the game state was implemented to allow players to pause their game and resume it later without losing their progress. This feature uses object serialization to write the game state to a file and deserialize it back into the program.

* **Implementation**:
  + **Saving the Game**: The game state, including the grid configuration, player details, and current scores, is serialized into a file named save.dat using ObjectOutputStream. This process is triggered when a player inputs quit during their turn.
  + **Loading the Game**: When the game starts, it checks for the existence of save.dat. If the file exists, the game state is deserialized using ObjectInputStream, and the game resumes from the last saved state.
  + **Serialization Details**: Classes such as Grid, Row, Square, and Disc implement the Serializable interface to ensure they can be serialized and deserialized effectively.

#### **3. Exception Handling for Robustness**

Exception handling is integrated into the save and load functionality to manage IOExceptions and ClassNotFoundExceptions that might occur during file operations. This ensures the game remains stable and responsive in case of errors during serialization or deserialization.

* **Implementation**:
  + Try-catch blocks are employed to catch exceptions during file operations, providing error messages to the user and allowing the game to continue smoothly without crashing.

These advanced features significantly enhance the usability and sophistication of the Connect 3 game, providing a seamless and enjoyable user experience. They demonstrate a higher level of programming skill and understanding, which goes beyond the basic requirements of the module.

**Conclusion**

The Connect 3 game is a comprehensive application of the object-oriented concepts taught in the SET11102 module. Through detailed design and implementation, the game not only meets the academic requirements but also provides an engaging and functional gaming experience. The addition of complex game logic and persistence features significantly enhances the gameplay, offering players a rich interaction experience.