Project 1 Phase 2 – Discussion

CMPT 276

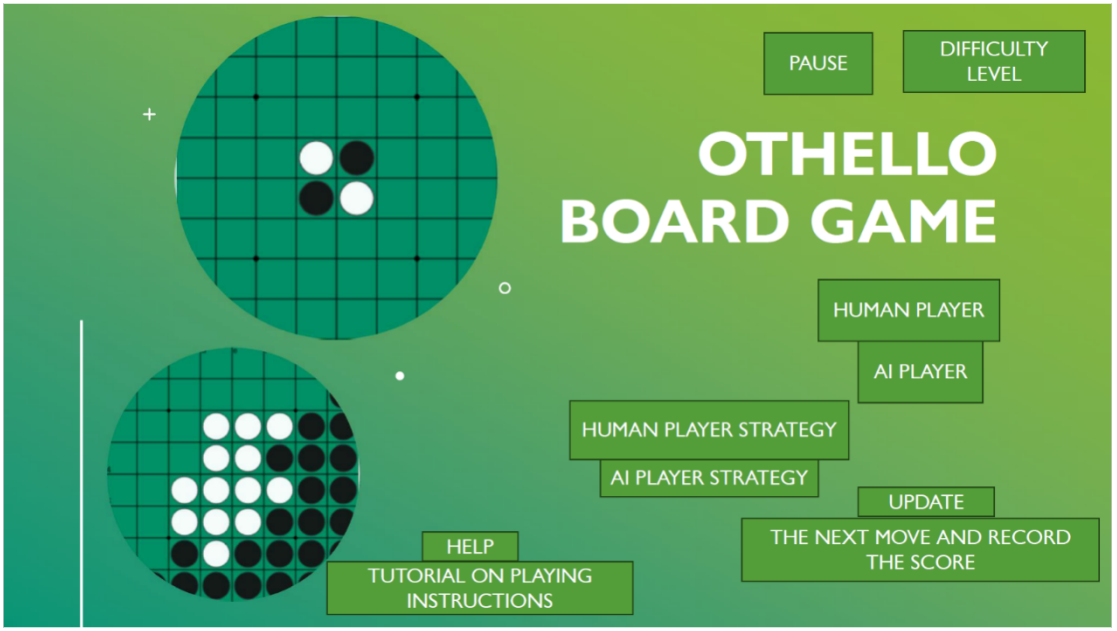
Prof: Dr. Tao Wang

Group: Ringtail

July 4, 2023

UI prototypes

Introduction to prototypes of gameplay page



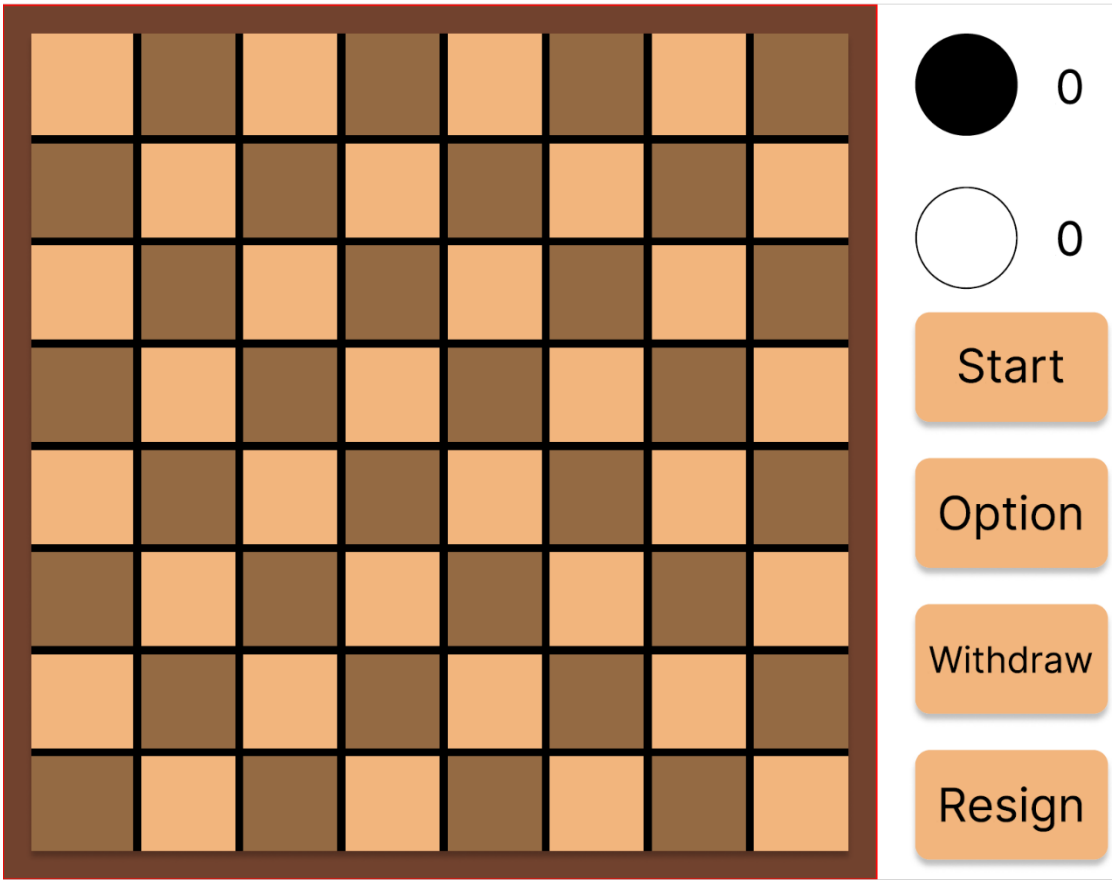
Romeesa's prototype

Introduction: This user interface (UI) prototype integrates all the game's features and presents a high degree of consistency. The logic of the information presented within the pages could be clearer, but each element has been designed with the novice user in mind. The distinctive grass-green colour of the board contrasts sharply with the black and white tiles, designed to catch the user's eye and provide clear visual feedback, reflecting one of the Nielsen Norman Group's ten takeaways - the principle of visibility.

The player can select a game mode and access a tutorial by clicking on the 'Help' button, applying the principle of 'user control and freedom'. In addition, the 'Strategy' button provides hints that human players need to help them develop strategies in the game, in line with the principle of 'Helping users identify, diagnose and correct errors'. The text and layout used are very consistent and do not include too many visual elements, ensuring the simplicity of the game interface, in line with the design principle of 'simple beauty'.

However, the prototype is not without its shortcomings. Firstly, the information on the page is poorly organized. While beginners can easily find all the information, more advanced players may find the presentation a little confusing as they progress through the game. This issue reflects the need to design for consistency and standards as much as possible to ensure a good user experience for all levels of players. In addition, designers need to consider the principle of 'flexible and efficient use' to ensure that more experienced users are able to perform more efficient actions.

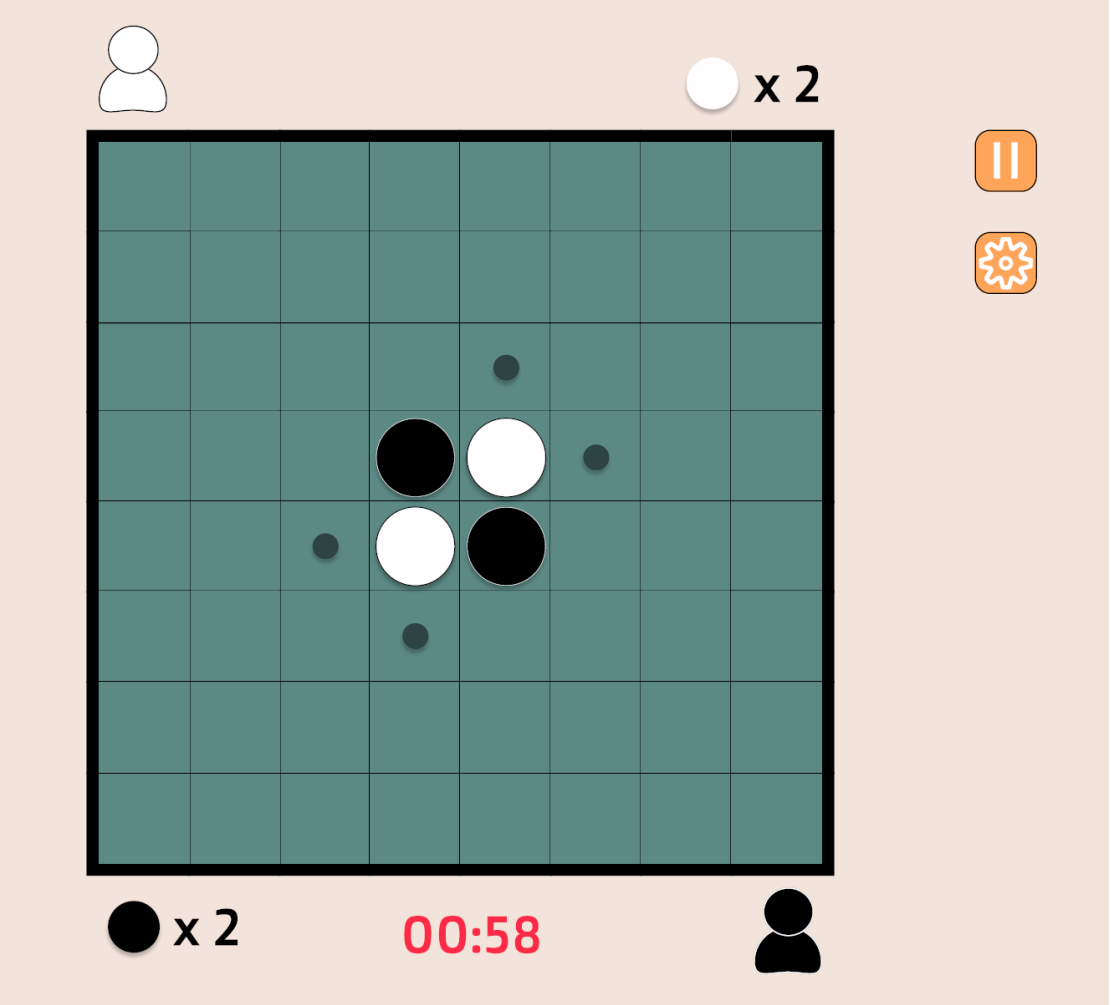
Overall, the UI prototype does a good job of implementing the basic functionality and some of the Nielsen Norman Group's inspiring principles, but there is room for improvement and refinement to provide a more advanced and richer user experience.



Leon’s Prototype:
Strengths: This prototype offers a highly intuitive understanding of the gameplay and the status of each player's pieces. The button design upholds a high degree of consistency from one to the next, furnishing players with a cohesive experience, an application of the "Consistency and Standards" principle outlined by the Nielsen Norman Group. The integration of information is streamlined, enabling players to easily locate the information they desire without distraction from superfluous elements, echoing the "Aesthetic and Minimalist Design" heuristic.

Weaknesses: Despite the prototype's simplicity, it provides inadequate guidance for new players, making it challenging for them to start experiencing the game. There's a lack of appropriate prompts to instruct newcomers, contravening the "Help users recognize, diagnose, and recover from errors" principle. Moreover, the features are rather scarce, potentially failing to provide players with the desired functionality, partly failing to meet player needs, opposing the "Match between system and the real world" heuristic. The design appears somewhat outdated, lacks freshness, and does not excel at capturing players' attention, which signifies room for improvement in adhering to the "Recognition rather than recall" and "Visibility of system status" principles.

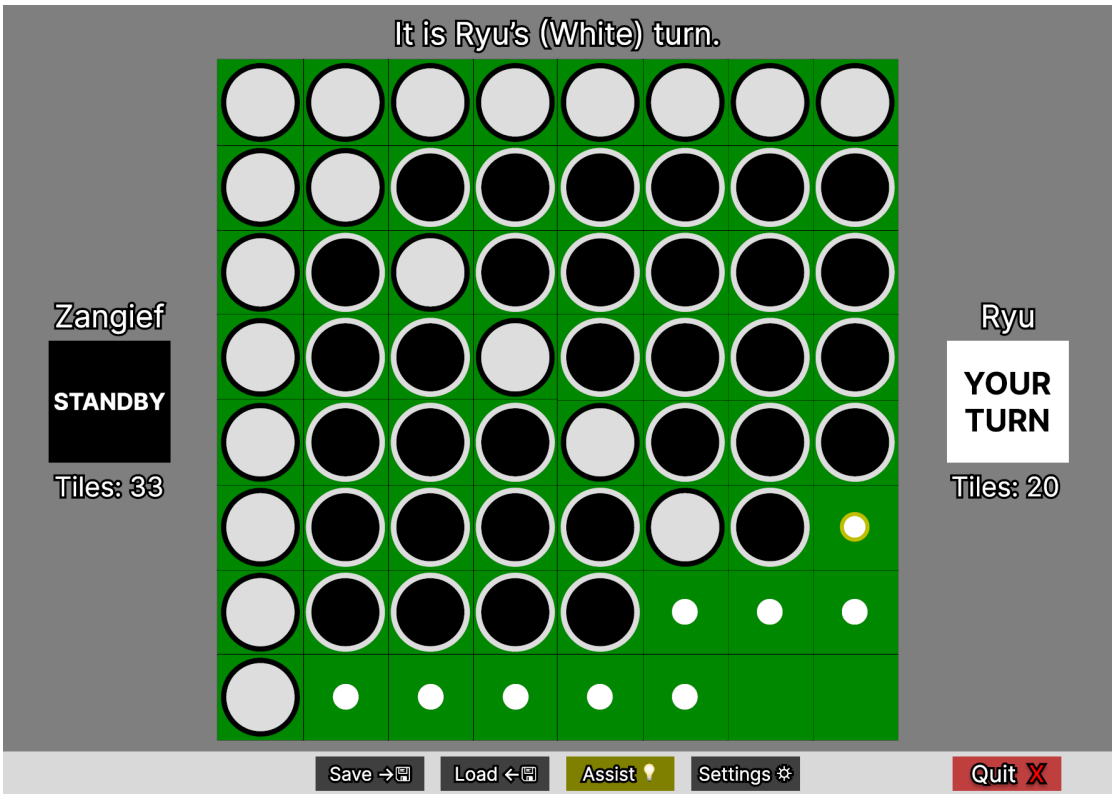
In conclusion, while the prototype performs well in certain design principles, such as "Consistency and Standards" and "Aesthetic and Minimalist Design", it leaves room for enhancement in others like "Helping users recognize, diagnose, and recover from errors", "Match between system and the real world", "Recognition rather than recall", and "Visibility of system status".



Josie’s Prototype
Strengths: The design of this prototype embraces a minimalist style, with a clear game board that enables players to concentrate on gameplay, embodying the "Aesthetic and Minimalist Design" principle from the Nielsen Norman Group. Additionally, the hints about pieces movement during the game provide a more user-friendly experience for newcomers, enhancing their engagement with the game, which speaks to the principle of "Help Users Recognize, Diagnose, and Recover from Errors". Furthermore, icons rather than text are utilized to represent the game status, contributing to a more succinct overall design, aligning with the "Recognition Rather than Recall" heuristic. Each player's turn time shifts positions depending on the player, making the interface more intuitive and adhering to the principle of "Flexibility and Efficiency of Use".

Weaknesses: A major drawback of the prototype is that it does not offer ample features, many of which are not intuitively found in the game interface, adversely affecting the players' gaming experience. This drawback is contrary to the principle of "Match between the System and the Real World". The design layout seems somewhat outdated and not sufficiently trendy, indicating room for improvement in implementing the "Aesthetic and Minimalist Design" principle. There's also a lack of necessary text to assist players during the game, falling short on the "Help and Documentation" heuristic.

In summary, while this prototype excels in applying several of the Nielsen Norman Group's principles such as "Aesthetic and Minimalist Design", "Help Users Recognize, Diagnose, and Recover from Errors", "Recognition Rather than Recall", and "Flexibility and Efficiency of Use", it could be improved by further incorporating "Match between System and the Real World" and "Help and Documentation".



Eric's Prototype: figma link
Strengths: This prototype boasts an exceptionally simple and comprehensible design. It uses gray as the base color for the interface outside the board, highlighting the game board and enhancing players' focus, adhering to the "Visibility of System Status" heuristic by the Nielsen Norman Group. The prototype also offers piece predictions, showing players where the pieces can be played during gameplay, effectively aiding players in recognizing and diagnosing potential moves, aligning with the "Help Users Recognize, Diagnose, and Recover from Errors" heuristic. At the interface's bottom, the buttons providing hints are colored in yellow, allowing players to easily locate help or hints when needed, thereby improving action efficiency, and exemplifying the "Recognition Rather than Recall" heuristic.

Weaknesses: The overall prototype design is somewhat bland, lacking unique features to attract and engage players, suggesting an opportunity for improvement in the "Aesthetic and Minimalist Design" heuristic. Though the provided functions are quite comprehensive, the amount of text is substantial, potentially imposing a cognitive load on players, especially those with dyslexia. This aspect contradicts the "Aesthetic and Minimalist Design" principle as it does not minimize irrelevant information, and indicates a potential lack of consideration for accessibility, which is an important part of user-centered design.

In conclusion, while the prototype effectively implements several Nielsen Norman Group's heuristics such as "Visibility of System Status", "Help Users Recognize, Diagnose, and Recover from Errors", and "Recognition Rather than Recall", there is room for improvement in the "Aesthetic and Minimalist Design" principle, as well as the inclusion of accessibility considerations in the design.

Comparison:

All these prototypes exhibit a commendable adherence to the principle of high degrees of consistency, a tenet that aligns with Nielsen's "Consistency and Standards"

heuristic. They effectively integrate complex elements into the design, thereby preventing cluttered elements from disrupting user focus and mitigating any potential confusion among players. Such an approach allows for a seamless user experience, where players can engage with the game effortlessly and without the need to adjust to inconsistent design elements.

However, certain prototypes possess a surplus of informational text, which can lead to reading fatigue quickly and potentially impede users from intuitively and swiftly performing their desired operations. This issue underscores the importance of the "Aesthetic and Minimalist Design" heuristic as outlined by Nielsen Norman Group, which advocates for avoiding a glut of information on a single page, thereby maintaining a clean, user-friendly interface.

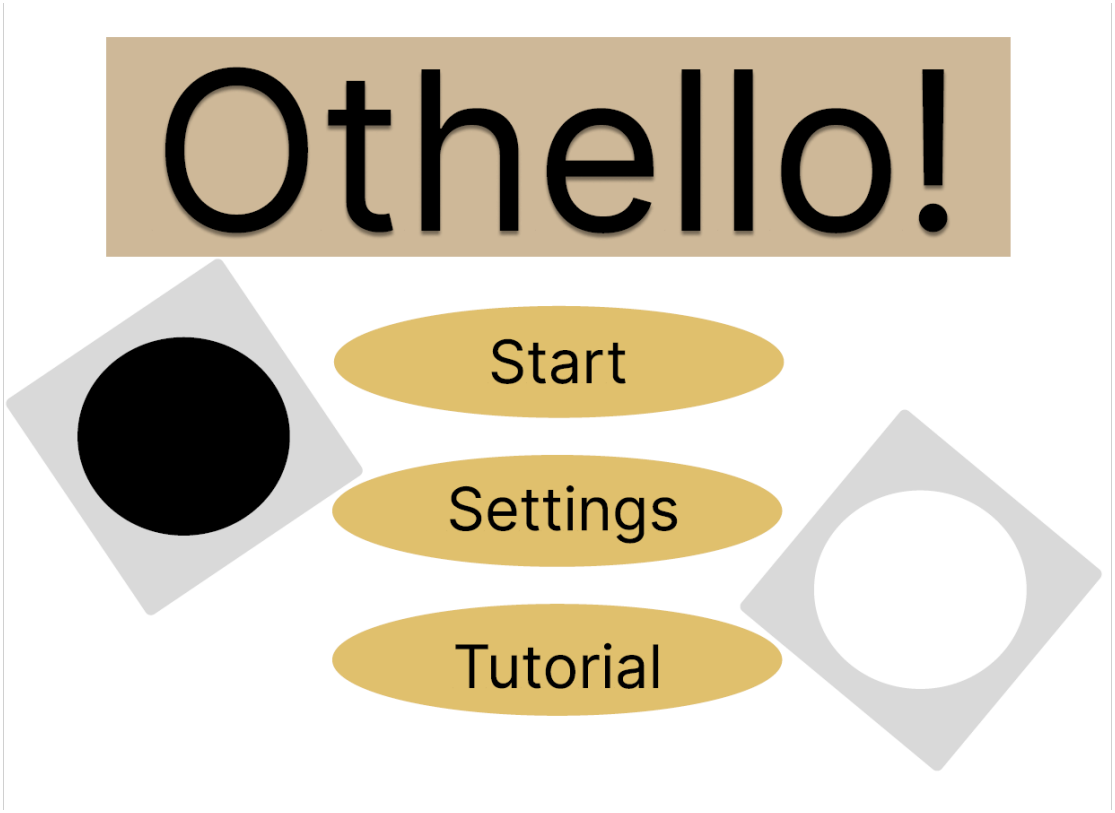
On the other hand, some prototypes are excessively simplistic and lack sufficient functional gateways for players, increasing their learning curve and diminishing user engagement. This is contrary to Nielsen's "Flexibility and Efficiency of Use" heuristic, which encourages designs that cater to both novice and expert users, providing shortcuts that speed up the interaction for the experienced user without overwhelming the novice.

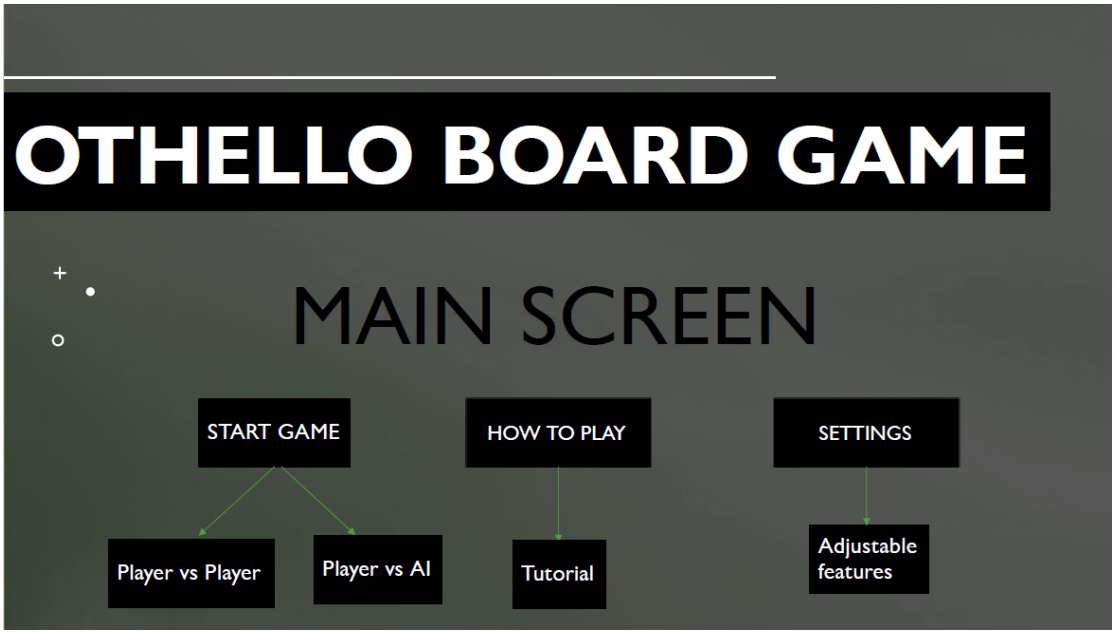
Furthermore, the employment of icons in lieu of text is an astute move, aligning with Nielsen's "Recognition Rather Than Recall" heuristic. This heuristic aims to reduce the cognitive burden on users by ensuring elements, actions, and options are visually represented. It encourages the idea that users should not be required to retain details from one part of the interface to apply it in another. Thus, incorporating more visual aids like pictures or icons to signify the functions of our games can simplify interactions and improve user comprehension.

Finally, the enduring theme throughout the prototypes is their unwavering commitment to "Consistency and Standards". This steadfast consistency in style ensures players experience no disruption or confusion while interacting with the game, cementing a cohesive and intuitive gaming journey. This design approach validates the prototypes' alignment with Nielsen's heuristics, showcasing a deep understanding of UI design principles, which foster an optimal user experience.

Extra.

Pause page.


Othello!
Start
Settings
Tutorial


OTHELLO BOARD GAME
MAIN SCREEN
START GAME
HOW TO PLAY
SETTINGS
Player vs Player
Player vs AI
Tutorial
Adjustable features

Start Menu

Design Patterns (Eric)

1. Behavioural D.P. – Command

This program implements a singleton Driver class to facilitate change of Panel instances (A custom class inheriting JPanel, handles composition of the game screen) for the game. It is responsible for keeping track of all Panel instances and implements methods that allows switching active panels by calling it from the interactable classes, utilising the Command design pattern to forego the need to implement additional methods or variables in the client classes, preventing class bloat and achieves the separation of concerns principle.

This is achieved by having the class handle and register the instantiation of each panel, paired with method changePanel() as follows:

Code snippet: Driver.changePanel(Panel panel)

```
public static synchronized void changePanel(Panel panel){
    instCheck();
    if(currentPanel != null)
        frame.remove(currentPanel);
    previousPanel = currentPanel;
    currentPanel = panel;
    frame.add(panel);
    frame.setVisible(b:true);
    frame.revalidate();
    frame.repaint();
}
```

…which is then paired with getter methods for the client code to pass the registered panels to the method, therefore minimising implementation required other than importing the class and calling the method, for example:

```
Driver.changePanel(Driver.getSettingsPanel());
```

…instead of needing to pass the references of panels into the classes during instantiation to keep track of in order to change panels, which is required for removal of the panel from the active JFrame, and to add the new panel back for display.

Furthermore, as this class is also a singleton, this comes with the side benefit of easier thread access control by having all client code use this static synchronous method, removing the need of using thread locks in these client

classes.

2. **Structural D.P. – Decorator**

This program also implements decorator patterns in the Gameplay and MenuBarButton classes to override the paint() and paintComponent() method from the inherited classes JPanel and JLabel respectively. This enables custom painting of elements and interactables on the display to coordinate aesthetics.

The Gameplay class overrides the default paint method to control rendering of TilePieces to dynamically change their appearance reflect the current board state of the game.

Code snippet: Gameplay.paint(Graphics g)

(Constructor parameters renamed for simplicity)

```
@Override
public void paint(Graphics g) {
    super.setBackground(new Color(rgb:0x3F3F3F));
    super.paint(g);

    for(int x = 0; x < 8; x++){
        for(int y = 0; y < 8; y++){
            boardTile[x][y] = new TilePiece(renderX, renderY, x, y, boardState[x][y]);
            this.add(boardTile[x][y]);
            boardTile[x][y].render(g);
        }
    }

    menuBar.render(g);
    revalidate();
}
```

The MenuBarButton class overrides the default paintComponent method to allow rendering of attributed strings, which enables changing font size or colour of the button text on the fly.

Code snippet: MenuBarButton.paintComponent(Graphics g)

```
@Override
protected void paintComponent(Graphics g){
    g.setColor(fillColour);
    g.fillRect(renderX, renderY, LENGTH, HEIGHT);
    g.setColor(new Color(rgb:0x0));
    g.drawRect(renderX, renderY, LENGTH, HEIGHT);
    if(buttonString != null){
        g.drawString(buttonString.getIterator(), renderX + 5 , renderY + HEIGHT / 4 * 3);
    }
    revalidate();
```

This will also allow easier implementation of accessibility features in the future by removing the need to reinstantiate a duplicate object just for

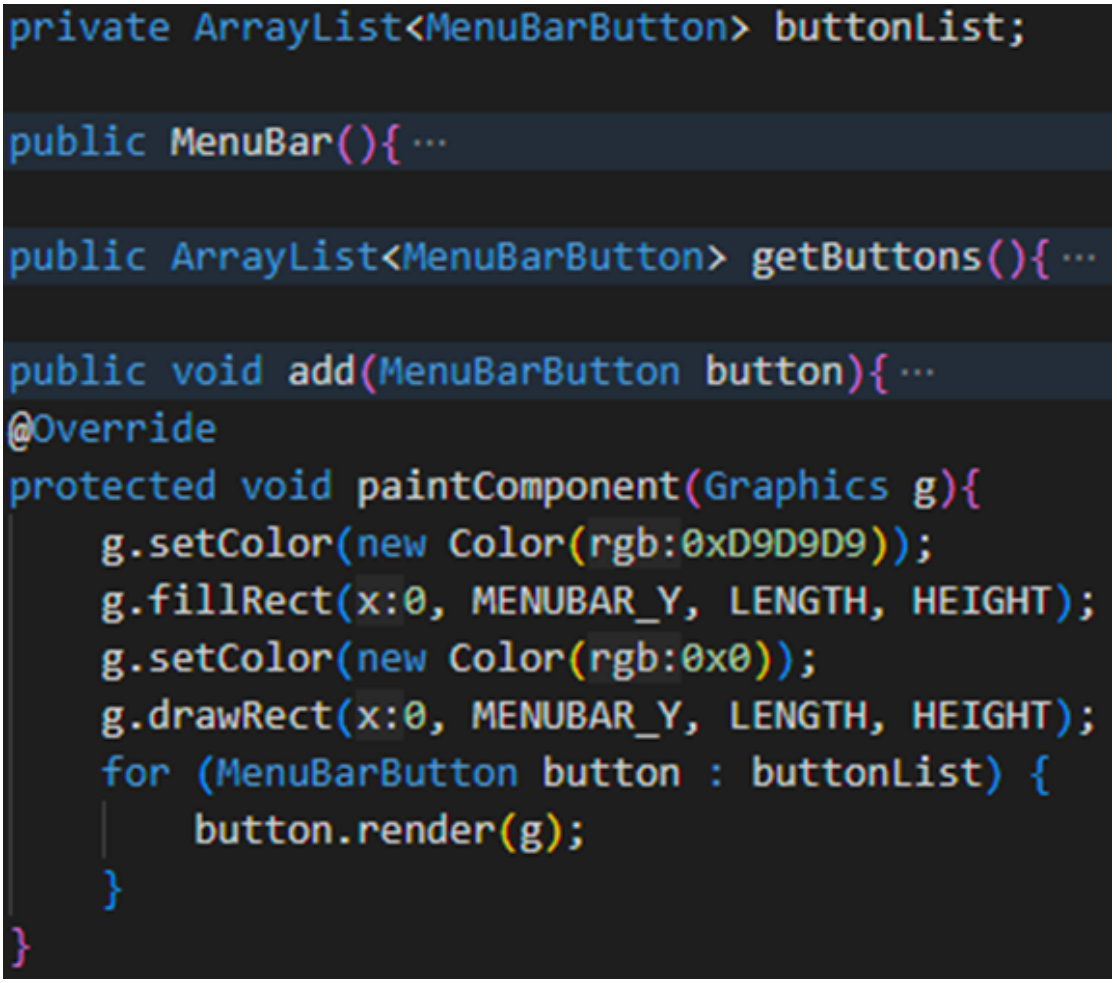
minor changes in colour or string attributes.

3. Structural D.P. – Composite

The MenuBar uses the Composite design pattern to act as a container for MenuBarButtons, centralising control of MenuBarButtons for Panel classes for easier tracking and manipulation.

The MenuBar overrides the paintComponent method to automatically iterate through all its buttons contained in buttonList and propagate render() calls, foregoing the need for the panel to keep track of and call render on each button individually.

Code snippet: class MenuBar and its methods



It will also enable panels to iterate through all buttons at once to change its position, size, background colour etc., which will further facilitate future implementation of accessibility features.