Name: Krunal Savaj

Reflection Note:

During the implementation of the given design, several challenges arose due to the absence of a clearly defined design pattern. I had to assume the overall structure, including adopting the MVC Responsibility driven architecture, which required careful management of class responsibilities. Additionally, some methods, like markSquaresAsEate() and isSquareUneaten(), overlapped in functionality, creating redundancy that needed to be streamlined. The dual use of graphical components (Graphics g) and Jbuttons also caused ambiguity, requiring the removal of one for a more efficient design.

Portions of the design that were clear and easy to understand included the use of the GameState enum to manage the game's status and the separation of game logic into distinct classes such as ChocolateBar, Player, and ChompGame. These made it intuitive to track the game state and the players' actions. The ButtonClickListener class also stood out for encapsulating event-handling logic in a modular way, ensuring reusability and maintaining a clean separation of concerns.

Based on the design I implemented, I would prioritize eliminating redundant methods, particularly in the ChocolateBar class, and ensure clearer input/output responsibilities. Additionally, aligning the design explicitly with the MVC pattern from the start would simplify implementation. By implementing someone else's design, I learned the value of clear structure, consistency in naming conventions, and the importance of modularity, particularly in how responsibility-driven programming and separation of concerns improve code flexibility and scalability.

Some Deviations in files:

Note: Please note that a more detailed explanation and reasoning for the changes have been provided in the code files through JavaDoc comments. The following are just concise notes summarizing the key deviations.

1. ChocolateBar.java
Added methods getRows() and getColumns() to access grid dimensions. These were not present in the original design.
Introduced isOnlyPoisonSquareLeft() to check if only the poison square remains uneaten.
Refined markSquaresAsEaten() to accurately mark squares to the right and above the selected square.

2. GameState.java
The GAME_OVER state was removed as it was redundant. The game logic was simplified to only track ONGOING, PLAYER_1_WIN, and PLAYER_2_WIN.

3. ChocolateBarPanel.java
The method for rendering graphics and handling button actions was simplified. Removed direct use of the Graphics gobject from this class to make the controller (ChompGame) manage the game state and updates, ensuring better separation of concerns.