**Design Overview**

The project is designed to simulate a Tombola game, which is a variant of Bingo, playable by multiple players. Each player's card is represented by a multi-linked list, where each node contains a number, a boolean flag to indicate whether the number has been marked, and pointers to the next node in the same row and column. This data structure is essential for efficiently marking numbers and checking for winning conditions without relying on arrays or ArrayLists, as stipulated in the project requirements.

**Classes and Their Responsibilities**

`Node.java`: Defines the structure of each node in the multi-linked list, including the number, whether it's been marked, and pointers to adjacent nodes (right and down).

`MultiLinkedList.java`: Manages the multi-linked list, including card generation, number marking, win condition checks, and card printing. It employs a unique method to shuffle numbers and arrange them according to Tombola rules.

`RandomPermutationGenerator.java`: Generates a random permutation of numbers from 1 to 90 to simulate the drawing of Tombola numbers without replacements.

`TombolaGame.java`: Orchestrates the game's flow, including initializing player cards, drawing numbers, marking numbers on cards, checking for win conditions, and updating the GUI with game progress.

`GameRunner.java`: A utility class to run the game in a non-GUI mode, mainly used for testing.

`CardPanel.java`, `GameFrame.java`: GUI components for displaying the Tombola cards and the game frame, allowing a visual representation of the game state.

**Implementation Details**

**Card Generation and Representation**

Each player's card is represented using a `MultiLinkedList` data structure. The `MultiLinkedList` class contains methods for generating cards according to Tombola rules, marking numbers as they are drawn, and checking for win conditions (e.g., Birinci Çinko, İkinci Çinko, Tombala).

- The cards are generated with numbers distributed according to Tombola's rules, ensuring a random yet valid distribution.

- The `markNumber` method updates nodes as numbers are drawn, facilitating the game's progress.

**Random Number Drawing**

The `RandomPermutationGenerator` class provides a static method to generate a shuffled list of numbers from 1 to 90, mimicking the drawing of numbers in a Tombola game without replacement.

**Game Progression and Win Condition Checks**

- The `TombolaGame` class uses the `drawNumbers` list from `RandomPermutationGenerator` to simulate the drawing of numbers.

- As numbers are drawn, they are marked on each player's card, and win conditions are checked after each number is marked.

- The game checks for three win conditions: Birinci Çinko (first line), İkinci Çinko (two lines), and Tombala (full house).

**GUI Implementation**

- The GUI, developed using Swing, enhances the game's usability and visual appeal.

- `GameFrame` serves as the main window, displaying each player's `CardPanel` and the current drawn number.

- `CardPanel` dynamically updates to show marked numbers and current game state, providing real-time feedback to players.

Conformance with Project Requirements

Multi-Linked List Usage: The implementation strictly adheres to the requirement of using a multi-linked list for card representation, ensuring efficient marking and checking operations.

Random Permutation Generation: The game simulates the drawing of numbers without replacements accurately.

Win Condition Logic: The project includes comprehensive logic to check for all specified win conditions, updating the game state accordingly.

GUI: The bonus requirement of a GUI has been fulfilled, providing an interactive and user-friendly interface for the game.

**Additional Observations and Recommendations**

Code Organization and Readability: The code is well-organized, with clear class responsibilities and method names that reflect their functionality. Comments and method descriptions further enhance readability.

Extensibility: The design allows for easy extension, such as adding more players or varying the win conditions.

Performance: The use of linked lists over arrays for the game's core logic demonstrates a thoughtful approach to managing memory and performance, particularly in marking numbers and checking win conditions.

**Conclusion**

The Tombola game project demonstrates a solid understanding of data structures, specifically the use of multi-linked lists to efficiently manage game state. The implementation covers all project requirements, including the advanced GUI component, providing a complete and functional game. Further enhancements could include networked multiplayer capabilities or more sophisticated AI for single-player modes. Overall, the project is a testament to effective problem-solving and software development skills in a game development context.