NumberGameGUI Code Documentation

1. Introduction

The **NumberGameGUI** is a Java-based graphical application that implements a number guessing game using Swing components for an interactive user interface. The game challenges the player to guess a randomly generated number between 1 and 100, with varying difficulty levels affecting the number of attempts and hint provision.

2. Overview

- **Language:** Java
- **GUI Framework:** Swing
- **Primary Libraries:** `javax.swing`, `java.awt`, `java.awt.event`, and `java.util`
- **Functionality:** This application provides an interactive game where users can input their guesses, receive dynamic feedback, and view their score in a formatted table at the end of the game. The game includes difficulty settings (Easy, Medium, Hard) that adjust the number of attempts and hint frequency.

3. Detailed Code Explanation

3.1. Initialization and Setup

- **Class Declaration and Inheritance:**
  - The class extends `JFrame` and implements `ActionListener` to create the main window and handle user events.
- **GUI Components:**
  - **JTextArea (`displayArea`):** Displays game instructions, feedback, and guess history. It is set to be non-editable and styled with a specific font.
  - **JTextField (`inputField`):** Allows the user to enter their guesses.
  - **JButton (`submitButton`):** Triggers the action to process the user's guess.
- **Other Variables:**
  - **Random Object:** Used to generate the target number for each round.
  - **Counters and Score Variables:** Variables such as `attempts`, `round`, `totalScore` keep track of the game state.
  - **Difficulty Settings:** Variables like `difficultyMultiplier` and `difficulty` adjust game parameters based on the selected difficulty.
  - **Guess History:** An `ArrayList<String>` stores the history of guesses and corresponding feedback.
  - **Instructions Flag:** A boolean flag (`instructionsShown`) to manage whether the instructions have been displayed.

3.2. GUI Components and Layout

- **Frame Settings:**
  - The frame is titled "NumberGame" and is set to a size of 600x500 pixels, with the default close operation defined.
  - The window is centered on the screen using `setLocationRelativeTo(null)`.
- **Panel Layout:**
  - A main panel using `BorderLayout` organizes the components.
  - **Display Area Panel:** A `JTextArea` is wrapped inside a `JScrollPane` with a styled border to allow scrolling of text.
  - **Input Panel:** Contains a label, the input field, and the submit button, all styled with a consistent font and placed at the bottom of the frame.

3.3. Event Handling and Game Logic

- **Instruction Display:**
  - The `showInstructions()` method outputs a set of game rules and details (including difficulty level, number range, etc.) in the `displayArea`.
- **Starting a New Round:**
  - The `startNewRound()` method increments the round counter, resets the number of attempts, generates a new target number, clears the guess history, and updates the display to indicate the start of a new round.
- **Handling User Input:**
  - The `actionPerformed()` method validates the input from the `inputField`, ensuring it is a valid number.
  - The number of attempts is incremented, and the guess is compared against the target number.
  - Depending on the guess, feedback is generated:
    - If the guess is too low or too high, an appropriate message is shown.
    - Additional hints (indicating whether the target number is even or odd) are provided based on the selected difficulty level (always for Easy; after 3 attempts for Medium; none for Hard).
    - If the guess is correct, a congratulatory message is appended.
  - Each feedback message is added to the guess history and then displayed by the `updateDisplay()` method.

3.4. Scoring and Round Completion

- **Scoring Mechanism:**

Upon a correct guess, the score for the round is calculated with the formula:
(maxAttempts - attempts + 1) * difficultyMultiplier * 10

    -
    - The round score is added to the total score.
- **Round Completion:**
    - When the user either correctly guesses the number or runs out of attempts, a dialog (using `JOptionPane`) is displayed to show the round summary (including round score and total score) and prompt whether to continue to another round.
    - If the user chooses to continue, a new round is started. Otherwise, the game ends.

3.5. End-of-Game Process

- **Displaying Final Score:**
    - The `endGame()` method creates a new JFrame that displays a final score summary using a `JTable`. The table shows the total rounds played and the accumulated score.
    - The main game window is disposed to conclude the application.

3.6. Application Launch

- **Main Method:**
    - The main method first presents a dialog with initial game instructions.
    - It then prompts the user to select a difficulty level (Easy, Medium, or Hard), which determines the maximum number of attempts and the scoring multiplier.
    - Finally, the GUI is launched using `SwingUtilities.invokeLater()`, ensuring that the creation and update of the GUI components occur on the Event Dispatch Thread.