Dynya Mckinney

CPSC - 39

11/30/24

Report: Guess the Number Game

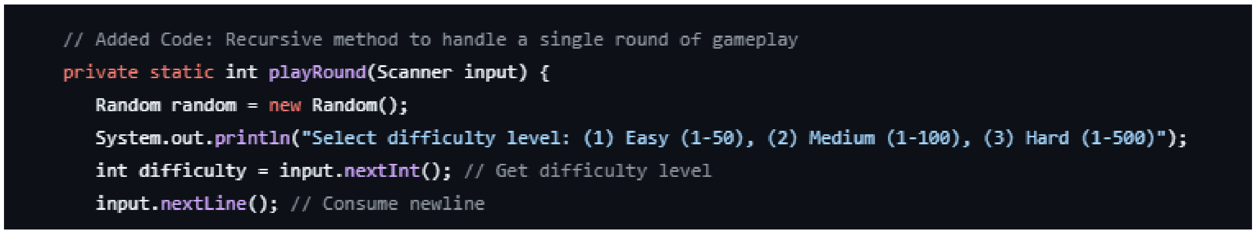
What the Game Does and Why It's Useful or Entertaining

The "Guess the Number" game is a simple and fun terminal-based game where players try to guess a random number within a set number of tries. There are three difficulty levels: Easy, Medium, and Hard, which change the range of numbers and the number of guesses allowed. The game keeps track of each guess, gives feedback, and calculates scores. It also shows guess history for each round and saves the player's total score over time. The game is enjoyable because it mixes logic, decision-making, and luck. The difficulty levels and score tracking make it exciting to play again and again.

Algorithms in Steps

1. Random Number Generator Algorithm

- **Steps**:
    1. Use Java's Random class to generate a number.
    2. Set the range of the number based on the selected difficulty level.
    3. Return the random number for the current round.

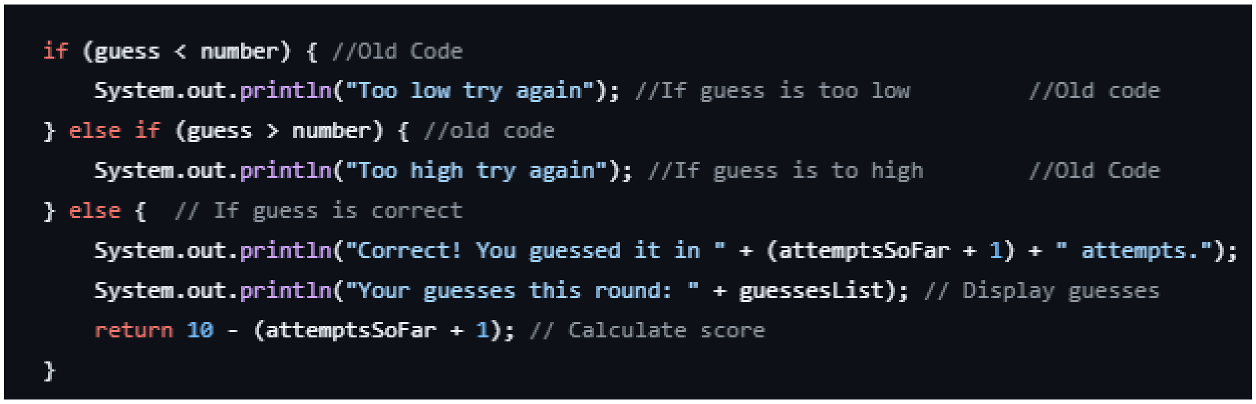


```
// Added Code: Recursive method to handle a single round of gameplay
private static int playRound(Scanner input) {
   Random random = new Random();
   System.out.println("Select difficulty level: (1) Easy (1-50), (2) Medium (1-100), (3) Hard (1-500)");
   int difficulty = input.nextInt(); // Get difficulty level
   input.nextLine(); // Consume newline
```

Usage: This algorithm generates the number that players need to guess in each round.

2. Guess Evaluation Algorithm

- **Steps**:
    1. Compare the player’s guess to the target number.
    2. Provide feedback: "Too low" or "Too high."
    3. End the round if the guess is correct or if the maximum attempts are reached.

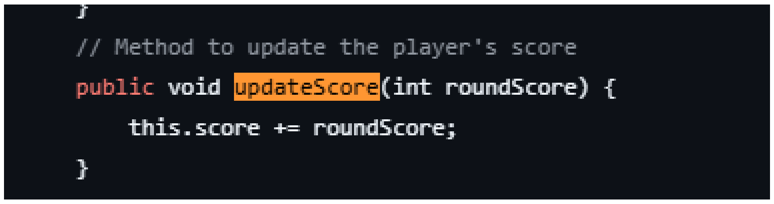


```
if (guess < number) { //Old Code
    System.out.println("Too low try again"); //If guess is too low         //Old code
} else if (guess > number) { //old code
    System.out.println("Too high try again"); //If guess is to high        //Old Code
} else {  // If guess is correct
    System.out.println("Correct! You guessed it in " + (attemptsSoFar + 1) + " attempts.");
    System.out.println("Your guesses this round: " + guessesList); // Display guesses
    return 10 - (attemptsSoFar + 1); // Calculate score
}
```

Usage: This algorithm checks each guess and guides the player toward the correct answer while keeping track of attempts.

3. Score Calculation Algorithm

- **Steps**:
    1. Retrieve the current score from a HashMap using the player’s name as the key.
    2. Subtract the number of guesses from a fixed maximum score (e.g., 10 points).
    3. Update the score in the HashMap.



```
}
// Method to update the player's score
public void updateScore(int roundScore) {
    this.score += roundScore;
}
```

Usage: This algorithm rewards players for guessing and updates their scores after each round.

Explanation of Algorithms

The random number generator makes the game more fun and unpredictable by creating a new number for each round based on the chosen difficulty. I used Java's Random class to build it and added difficulty settings to improve the experience. The guess-checking system tells players if their guess is too high or too low. I improved it by adding checks to handle invalid inputs. I also created the scoring system from scratch to update scores based on how well players do. ChatGPT helped explain ideas and suggested improvements, but I wrote and finalized the code myself.

Big O Analysis

- **Random Number Generator**: O(1) because generating a random number is a constant-time operation.
- **Guess Evaluation**: O(1) for each comparison, but the loop operates in O(n), where n is the number of guesses.
- **Score Calculation**: O(1) since it involves simple arithmetic and updating a HashMap.

Data Structures Used

1. **Array**: Keeps track of the player's guesses during each round. It's simple and makes it easy to show the guesS history.
2. **Stack**: Keeps count of the guesses in each round. It works well for storing and retrieving data in order, like how a player did in each game.
3. **LinkedList:** Stores guesses during each round. It dynamically grows to accommodate any number of guesses.

Opportunity Encountered

While working on the game, I decided to add difficulty levels to make it more flexible. This lets players choose how challenging they want the game to be. By using different number ranges and attempts, I made the game more fun for players with different skill levels.

Error Encountered

At first, I had trouble keeping track of player turns when multiple players wanted to play more rounds. Sometimes the system skipped players or didn't put them back in the right order. After checking the code, I realized I wasn't always adding players back to the queue after their turn. I fixed it by making sure players were added back to the queue only if they chose to play another round. This solved the problem and kept the turns in the correct order..

Future Improvements

In the next version of the game, I want to add a leaderboard to track high scores for multiple players. This would make the game more competitive and fun to play again and again. I also plan to add a timer to make it more challenging by limiting how long players have to guess. Lastly, I want to turn the terminal-based game into a full app with a graphical interface to make it more interactive, visually appealing, and exciting to use.