|  |  |  |
| --- | --- | --- |
| **TABLE OF CONTENTS** | | |
| **CHAPTER NO** | **TITLE** | **PAGE NO** |
|  | **ACKNOWLEDGEMENT** | I |
|  | **ABSTRACT** | II |
|  | **TABLE OF CONTENT**  Introduction | 1  2 |
| 1 | UML Diagram | 3 |
| 2 | Flowchart | 4 |
| 3 | Code | 5 |
| 4 | OUTPUT | 8 |
| 5 | Explanation Of Code | 9 |
|  | **CONCLUSIONS** | 13 |

Introduction

Welcome to the Number Guessing Game! In this game, your goal is to guess a secret number that the computer has randomly selected. You will have a limited number of attempts to figure out the number based on whether your guess is too high, too low, or correct.The computer will choose a random number between 1 and 100.You will have a maximum of 10 attempts to guess the correct number. After each guess, the game will tell you if your guess was too high or too low, helping you narrow down the possible numbers .You can track your previous guesses and receive feedback on how many attempts remain.The game ends when you either guess the number correctly or exhaust all of your attempts Enter a number between 1 and 100.Based on your guess, the computer will provide feedback (too high, too low, or correct).Keep guessing until you find the correct number or run out of attempts! Good luck, and have fun! Let’s see if you can guess the right number!

This introduction provides players with a brief overview of how the game works, including the rules and objectives. It sets the stage for an engaging experience while keeping things simple and clear for new players.

|  |  |  |  |
| --- | --- | --- | --- |
| **1 .UML Diagram**   |  | | --- | | **NumberGuessingGame**  **+ main(args:String[]):void**  **+ random:Random**  **+ scanner:Scanner**  **+ numberToGuess:int**  **+ maxAttempts:int**  **+ guessCount:int**  **+ hasGuessedCorrectly:boolean** | | | |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  | 2. Flowchart |  |
|  | 3.Numberguessing game |  |
|  |  |  |
|  |  |  |
|  |  |  |

import java.util.Scanner;

import java.util.Random;

public class NumberGuessingGame {

public static void main(String[] args) {

// Game Setup

Random random = new Random();

Scanner scanner = new Scanner(System.in);

int numberToGuess = random.nextInt(100) + 1; // Random number between 1 and 100

int maxAttempts = 10;

int[] guesses = new int[maxAttempts]; // Array to store guesses

int guessCount = 0;

boolean hasGuessedCorrectly = false;

System.out.println("Welcome to the Number Guessing Game!");

System.out.println("I have selected a number between 1 and 100.");

System.out.println("You have " + maxAttempts + " attempts to guess it.");

// Game Loop

while (guessCount < maxAttempts && !hasGuessedCorrectly) {

System.out.print("Enter your guess (Attempt " + (guessCount + 1) + " of " + maxAttempts + "): ");

int playerGuess = scanner.nextInt();

// Store the guess in the array

guesses[guessCount] = playerGuess;

guessCount++;

// Check if the guess is correct

if (playerGuess == numberToGuess) {

hasGuessedCorrectly = true;

System.out.println("Congratulations! You guessed the correct number.");

} else if (playerGuess < numberToGuess) {

System.out.println("Too low! Try again.");

} else {

System.out.println("Too high! Try again.");

}

// Provide feedback on previous guesses

System.out.print("Previous guesses: ");

for (int i = 0; i < guessCount; i++) {

System.out.print(guesses[i] + " ");

}

System.out.println();

}

// End of Game

if (!hasGuessedCorrectly) {

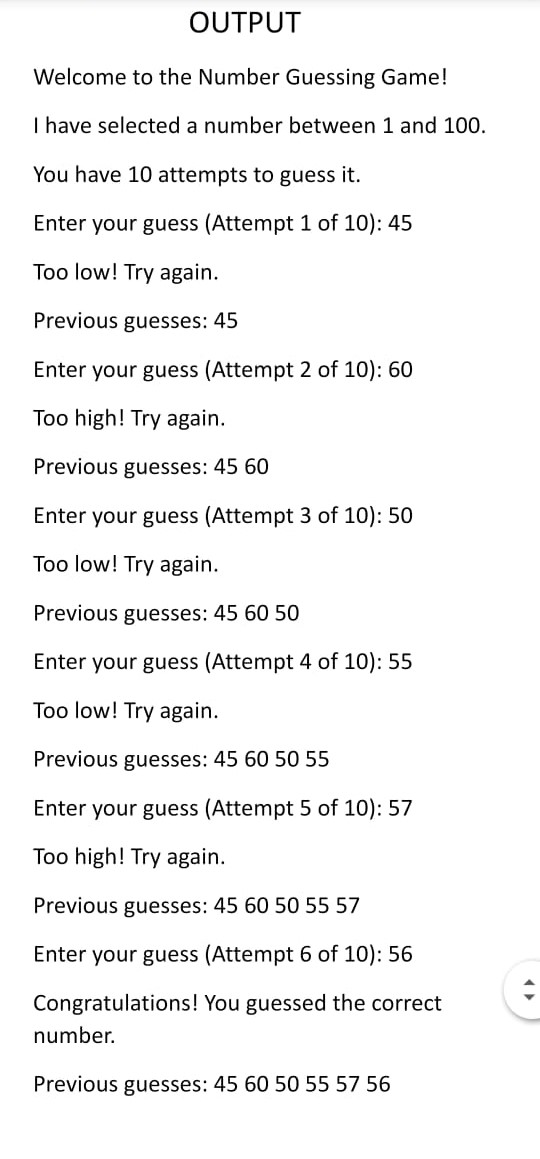
System.out.println("Sorry! You've used all " + maxAttempts + " attempts. The correct number was " + numberToGuess + ".");

}

scanner.close();

}

}



5.Explanation of Number Guessing Game

1. Imports

import java.util.Scanner;

import java.util.Random;

Scanner: This class is used to read input from the user via the console. It will help us get the user's guess.

Random: This class is used to generate a random number. In this case, it's used to select the number the player will try to guess.

2. The main Method

public static void main(String[] args) {

This is the entry point of the Java program. When you run the program, this method gets executed. All of the game logic resides inside this method.

3. Initializing Variables

Random random = new Random();

Scanner scanner = new Scanner(System.in);

int numberToGuess = random.nextInt(100) + 1; // Random number between 1 and 100

int maxAttempts = 10;

int[] guesses = new int[maxAttempts]; // Array to store guesses

int guessCount = 0;

boolean hasGuessedCorrectly = false;

random: This object is used to generate a random number. random.nextInt(100) generates a number between 0 and 99. By adding 1, it ensures the number is between 1 and 100.

scanner: This object is used to read user input from the console.

numberToGuess: A random number between 1 and 100 that the player has to guess.

maxAttempts: This is the maximum number of guesses the player is allowed (10 guesses in this case).

guesses: An array to store the guesses the player has made. This is used to give feedback on all past guesses.

guessCount: This variable keeps track of how many guesses the player has made so far. It starts at 0 and increments each time the player makes a guess.

hasGuessedCorrectly: A boolean flag that tracks if the player has guessed the correct number. Initially, it is set to false and will be set to true once the player guesses correctly.

4. Displaying Instructions

System.out.println("Welcome to the Number Guessing Game!");

System.out.println("I have selected a number between 1 and 100.");

System.out.println("You have " + maxAttempts + " attempts to guess it.");

These lines provide the player with some basic instructions about the game, telling them the range of the number (1 to 100) and how many attempts they have (10 attempts).

5. Game Loop

while (guessCount < maxAttempts && !hasGuessedCorrectly) {

This is the core of the game. The while loop will keep running as long as the player hasn't guessed correctly (!hasGuessedCorrectly) and the player hasn't used up all of their attempts (guessCount < maxAttempts).

guessCount < maxAttempts: Ensures the player can only make up to 10 guesses.

!hasGuessedCorrectly: Ensures the loop continues until the player guesses the correct number.

6. Player Input (Guess)

System.out.print("Enter your guess (Attempt " + (guessCount + 1) + " of " + maxAttempts + "): ");

int playerGuess = scanner.nextInt();

The program prompts the user to enter their guess. The prompt includes the current attempt number (e.g., "Attempt 1 of 10").

scanner.nextInt() reads the user’s input as an integer and stores it in playerGuess.

7. Storing the Guess

guesses[guessCount] = playerGuess;

guessCount++;

The guess entered by the player is saved in the guesses array at the index guessCount.

After storing the guess, the guessCount is incremented by 1 to prepare for the next guess.

8. Check the Guess

if (playerGuess == numberToGuess) {

hasGuessedCorrectly = true;

System.out.println("Congratulations! You guessed the correct number.");

} else if (playerGuess < numberToGuess) {

System.out.println("Too low! Try again.");

} else {

System.out.println("Too high! Try again.");

}

Correct Guess: If the player's guess is equal to numberToGuess, the game ends, and the player is congratulated.

Too Low: If the player's guess is less than the target number, the program informs the player that the guess is "Too low!" and prompts them to try again.

Too High: If the player's guess is greater than the target number, the program informs the player that the guess is "Too high!" and prompts them to try again.

9. Displaying Previous Guesses

System.out.print("Previous guesses: ");

for (int i = 0; i < guessCount; i++) {

System.out.print(guesses[i] + " ");

}System.out.println();

After each guess, the program displays all of the previous guesses the player has made so far. It loops through the guesses array and prints each guess, providing feedback to the player about what they’ve already tried.

10. End of the Game

Once the player either guesses the number or runs out of attempts, the loop ends.

if (!hasGuessedCorrectly) {

System.out.println("Sorry! You've used all " + maxAttempts + " attempts. The correct number was " + numberToGuess + ".");

}

If the player has not guessed correctly after 10 attempts, the program reveals the correct number and tells the player they’ve used up all their guesses.

11. Closing the Scanner

scanner.close();

Finally, the scanner is closed to release the system resources it was using. This is a good practice when you are done using a Scanner.

**CONCLUSION**

The Number Guessing Game is a simple yet engaging console-based game that allows players to guess a randomly chosen number between 1 and 100. The game provides feedback after each guess, indicating whether the guess is too low, too high, or correct. The player has 10 attempts to guess the correct number, and the program tracks all previous guesses to help the player make informed choices.

1. Random Number Generation: The use of Random to select a number between 1 and 100 ensures that each game is different, providing variety and replayability.2. Input Validation: The getPlayerGuess() method ensures that the player only inputs valid numbers within the acceptable range (1–100). This prevents runtime errors and ensures a smooth user experience, even when the player makes mistakes.3. Feedback Mechanism: After each guess, the program tells the player whether their guess is too high, too low, or correct. It also prints the previous guesses, which helps the player track their progress and avoid repeating guesses.4. Game Flow Control: The game uses a loop that runs until the player either guesses correctly or uses up all 10 attempts. This ensures that the game has a clear starting point and ending, with appropriate messages displayed when the game is over.5. Code Organization: By breaking down the game into smaller, reusable methods (getPlayerGuess() and printPreviousGuesses()), the code becomes more modular, easier to maintain, and easier to read. Each method has a specific responsibility, which is a good practice in programming.