

Presented by: **Maryam Siddiqui**

**www.linkedin.com/in/maryam-siddiqui-ghani**

Task: **Random Password Generator**

**Project Report: Enhanced Random Password Generator**

**Introduction**

The Enhanced Random Password Generator is a Java application designed to create customized random passwords for users. In today's digital age, strong and unique passwords are essential for maintaining online security. This tool aims to provide users with a convenient way to generate passwords that meet their specific criteria, such as length, character types, and user feedback.

**Project Overview**

The Enhanced Random Password Generator is implemented in Java and utilizes the `SecureRandom` class for secure random number generation. It interacts with users through the command-line interface, prompting for password criteria and feedback. The key features of the application include:

* Customizable password length
* Inclusion/exclusion of uppercase letters, lowercase letters, symbols, and digits
* Gathering and displaying user feedback on generated passwords
* Welcoming and farewell messages for a user-friendly experience

Certainly! Here is a detailed explanation of the features implemented in the Enhanced Random Password Generator:

**Features of the Enhanced Random Password Generator**

* **Customizable Password Length**

The Enhanced Random Password Generator allows users to specify the desired length of the generated password. This feature ensures that users can create passwords of varying lengths based on their security needs and the requirements of different online services.

* **Inclusion/Exclusion of Character Types**

Users have the flexibility to choose which character types they want to include in the generated password. The available character types are:

* Uppercase Letters (A-Z)
* Lowercase Letters (a-z)
* Symbols (!@#$%^&\*()-\_=+[]{}|;:,.<>?)
* Digits (0-9)

By selecting or deselecting these character types, users can tailor the password to meet specific security policies or preferences.

* **User-Friendly Interaction**

The application provides a clear and user-friendly command-line interface for interacting with users. Users are guided through the process of generating a password, making choices for character types, and providing feedback.

* **Welcoming and Farewell Messages**

The tool starts with a welcoming message that introduces users to the Enhanced Random Password Generator's purpose and features. This message creates a positive tone and sets the context for the user's interaction. Additionally, when users decide to exit the password generation loop, a farewell message is displayed, thanking them for using the tool and encouraging responsible online security practices.

* **User Feedback Collection**

After generating a password, users have the option to provide feedback about the generated password. This feedback collection mechanism allows users to express their thoughts, suggestions, or concerns regarding the password. User feedback can be valuable for improving the tool and addressing user needs.

* **Secure Random Password Generation**

The application utilizes the `SecureRandom` class for secure random number generation. This ensures that the generated passwords are truly random and suitable for use in security-sensitive contexts.

* **Looping for Multiple Passwords**

The program is designed to allow users to generate multiple passwords consecutively without restarting the application. After generating a password and receiving feedback, users can choose to generate another password or exit the loop.

* **Exception Handling**

The application incorporates exception handling to handle various scenarios, such as invalid input or conflicting preferences. For example, if a user selects no character types for inclusion, an `IllegalArgumentException` is thrown to prompt the user to select at least one character type.

* **Consume Newline**

The `getUserFeedback()` function includes a mechanism to consume newline characters to ensure accurate user input capture. This ensures that user feedback is correctly recorded and processed.

* **Encouraging Secure Practices**

Throughout the interaction, the tool emphasizes the importance of online security and the need for strong, unique passwords. The messages and prompts encourage users to make informed choices to enhance their overall digital security.

The Enhanced Random Password Generator successfully combines various features to provide users with a powerful and customizable tool for generating secure passwords. By allowing users to define password length, character types, and feedback, the application empowers users to create strong passwords tailored to their preferences and security needs. The user-friendly interface and inclusion of welcoming and farewell messages contribute to an overall positive user experience, reinforcing the importance of responsible online security practices.

**Functionality**

* **Main Method**

The `main` method is the entry point of the application. It sets up the user interaction loop, where users can generate multiple passwords with different criteria and provide feedback.

* **printStartingMessage()**

The `printStartingMessage()` function displays a welcoming message to users, explaining the purpose of the tool and encouraging them to get started. This function sets a positive tone and provides context for the user.

* **printFarewellMessage()**

The `printFarewellMessage()` function is called when the user decides to exit the password generation loop. It thanks the user for using the tool, reinforces the importance of online security, and wishes the user a great day. This function ensures a pleasant conclusion to the user's interaction with the program.

* **getUserInput()**

The `getUserInput()` function prompts the user for input and returns an integer representing the desired password length. This function ensures that the user's input is correctly captured and can be used for password generation.

* **getUserPreference()**

The `getUserPreference()` function prompts the user with a yes/no question and returns a boolean value (`true` for "yes" and `false` for "no"). It is used to gather the user's preferences for including uppercase letters, lowercase letters, symbols, and digits in the password.

* **generateRandomPassword()**

The `generateRandomPassword()` function takes user preferences and generates a random password accordingly. It concatenates character sets based on user preferences and uses the `SecureRandom` class to create a secure password of the specified length.

* **getUserFeedback()**

The `getUserFeedback()` function prompts the user to provide feedback about the generated password. It captures the user's input and returns it as a string. This feature allows users to express their thoughts on the generated password, which can be valuable for further improvements.

**Conclusion**

The Enhanced Random Password Generator offers a user-friendly and customizable solution for generating strong and secure passwords. By allowing users to define password criteria, providing feedback, and offering clear starting and farewell messages, the tool empowers users to enhance their online security practices. This project demonstrates effective use of Java programming concepts, user input handling, and secure random number generation. It serves as a valuable resource for individuals seeking a convenient and reliable method to create strong passwords tailored to their needs.

Feel free to expand on this report further and customize it according to your project's specific requirements and details.

**Acknowledgments:**

This project was conceived and executed by Maryam Siddiqui during her internship at Info Aidtech. Gratitude is extended to mentors and guides for their invaluable support and guidance during the project's development.

**Contact Information:**

For inquiries or feedback regarding this project, kindly reach out to:

Maryam Siddiqui

Note: This project report provides a concise overview and can be expanded as per the specific requirements and details of the actual project.

**Source code :**

import java.security.SecureRandom;

import java.util.Scanner;

public class EnhancedRandomPasswordGenerator {

private static final String LOWERCASE = "abcdefghijkmnpqrstuvwxyz";

private static final String UPPERCASE = "ABCDEFGHJKLMNPQRSTUVWXYZ";

private static final String DIGITS = "23456789";

private static final String SYMBOLS = "!@#$%^&\*()-\_=+[]{}|;:,.<>?";

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

printStartingMessage();

do {

int length = getUserInput(scanner, "Enter desired password length: ");

boolean useUppercase = getUserPreference(scanner, "Include uppercase letters? (y/n): ");

boolean useLowercase = getUserPreference(scanner, "Include lowercase letters? (y/n): ");

boolean useSymbols = getUserPreference(scanner, "Include symbols? (y/n): ");

boolean useDigits = getUserPreference(scanner, "Include digits? (y/n): ");

String password = generateRandomPassword(length, useUppercase, useLowercase, useSymbols, useDigits);

System.out.println("Generated Password: " + password);

String feedback = getUserFeedback(scanner);

System.out.println("Feedback: " + feedback);

System.out.print("Do you want to generate another password? (y/n): ");

String response = scanner.next();

if (!response.equalsIgnoreCase("y")) {

printFarewellMessage();

break;

}

} while (true);

scanner.close();

}

public static void printStartingMessage() {

System.out.println("Welcome to the Enhanced Random Password Generator!");

System.out.println("This tool allows you to create customized random passwords.");

System.out.println("Let's get started...");

System.out.println();

}

public static void printFarewellMessage() {

System.out.println();

System.out.println("Thank you for using the Enhanced Random Password Generator!");

System.out.println("Stay secure and have a great day!");

}

public static int getUserInput(Scanner scanner, String prompt) {

System.out.print(prompt);

return scanner.nextInt();

}

public static boolean getUserPreference(Scanner scanner, String prompt) {

System.out.print(prompt);

String response = scanner.next();

return response.equalsIgnoreCase("y");

}

public static String generateRandomPassword(int length, boolean useUppercase, boolean useLowercase,

boolean useSymbols, boolean useDigits) {

if (!useUppercase && !useLowercase && !useSymbols && !useDigits) {

throw new IllegalArgumentException("At least one character set must be selected.");

}

String characters = "";

if (useUppercase) characters += UPPERCASE;

if (useLowercase) characters += LOWERCASE;

if (useSymbols) characters += SYMBOLS;

if (useDigits) characters += DIGITS;

StringBuilder password = new StringBuilder();

SecureRandom random = new SecureRandom();

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

int index = random.nextInt(characters.length());

password.append(characters.charAt(index));

}

return password.toString();

}

public static String getUserFeedback(Scanner scanner) {

System.out.print("Please provide feedback about the generated password: ");

scanner.nextLine(); // Consume newline

return scanner.nextLine();

}

}

**Output:**



