Password Strength Analyzer in Java

Abstract:

The Password Strength Analyzer project evaluates the strength of user-entered passwords based on predefined security rules. It classifies passwords as **Weak**, **Medium**, or **Strong** by checking parameters such as length, uppercase and lowercase letters, numbers, and special characters.
This project demonstrates the use of **regular expressions, conditional logic, and string handling** in Java to analyze text-based input efficiently.

Introduction:

Passwords are one of the most common methods of user authentication. Weak passwords can make systems vulnerable to hacking attempts.
This project analyzes the strength of passwords entered by a user and provides instant feedback to encourage stronger password creation.

Features:

- Checks password **length**.
- Verifies presence of **uppercase and lowercase** characters.
- Ensures inclusion of **digits** and **special characters**.

- Classifies strength as **Weak**, **Medium**, or **Strong**.

This project uses **Java Regular Expressions (Regex)** and **String methods** to evaluate password patterns and demonstrates practical use of logic-based programming in cybersecurity.

Program Code:

```
import java.util.Scanner;
import java.util.regex.*;

public class PasswordStrengthAnalyzer {

    // Function to check password strength
    public static String checkStrength(String password) {
        int strengthPoints = 0;

        // Criteria 1: Length
        if (password.length() >= 8)
            strengthPoints++;

        // Criteria 2: Contains lowercase letters
        if (password.matches(".*[a-z].*"))
            strengthPoints++;

        // Criteria 3: Contains uppercase letters
        if (password.matches(".*[A-Z].*"))
            strengthPoints++;
        // Criteria 4: Contains digits
        if (password.matches(".*[0-9].*"))
            strengthPoints++;

        // Criteria 5: Contains special characters
        if (password.matches(".*[!@#$%^&*(),.?\":{}|<>].*"))
            strengthPoints++;

        // Strength evaluation
        if (strengthPoints <= 2)
            return "Weak";
        else if (strengthPoints == 3 || strengthPoints == 4)
            return "Medium";
        else
            return "Strong";
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String password;
        char choice;

        System.out.println("=== Password Strength Analyzer
===");

        do {
            System.out.print("\nEnter a password to analyze: ");
            password = sc.nextLine();

            String strength = checkStrength(password);
            System.out.println("Password Strength: " + strength);

            if (strength.equals("Weak")) {
                System.out.println("Tips to improve your password:");
                System.out.println("- Use at least 8 characters");
                System.out.println("- Add uppercase, lowercase,
digits, and symbols");
            }

            System.out.print("\nDo you want to check another
password? (y/n): ");
            choice = sc.next().charAt(0);
            sc.nextLine(); // clear buffer

        } while (choice == 'y' || choice == 'Y');

        System.out.println("\nExiting Password Analyzer... Stay
Secure!");
        sc.close();
    }
}
```

Sample Output:

```
=== Password Strength Analyzer ===

Enter a password to analyze: hello
Password Strength: Weak
Tips to improve your password:
- Use at least 8 characters
- Add uppercase, lowercase, digits, and symbols

Do you want to check another password? (y/n): y

Enter a password to analyze: Hello123
```

```
Password Strength: Medium

Do you want to check another password? (y/n): y

Enter a password to analyze: Hello@123
Password Strength: Strong

Do you want to check another password? (y/n): n

Exiting Password Analyzer... Stay Secure!
```

Conclusion:

The **Password Strength Analyzer in Java** project effectively demonstrates the use of **regular expressions** and **string manipulation** to evaluate password quality. By categorizing passwords as Weak, Medium, or Strong, it promotes better security practices among users. This project can be extended to include file storage for tracking user passwords or integration with GUI interfaces (like JavaFX or Swing) for a more interactive experience.
