DOCUMENTATION OF HOW THE PASSWORD GENERATOR WORKS.

Class : Main.java

This program is the entry point of an application that takes user input from the console. It initializes an instance of a class named ***Generator*** and executes its main logic loop.

Modules and Imports Used

java.util.Scanner: Used to read input from the keyboard.

How It Works

A ***Scanner*** named ***keyboard*** is created to read input.

In the main method, a ***Generator*** object is created using this scanner.

generator.mainLoop() is called to run the main logic.

The ***try*** block with ***keyboard*** ensures the scanner is closed automatically.

Class: Generator.java

This class handles the core logic for a password utility program, including generating passwords, checking their strength, and interacting with the user via a menu-driven interface.

Modules and Imports Used

java.util.Scanner: Used for capturing user input from the console.

Alphabet alphabet: Manages the allowed characters for password generation.

Scanner keyboard: Shared scanner instance for input.

Constructors

Generator(Scanner scanner): Sets up the shared scanner for user interaction from keyboard.

Generator(boolean IncludeUpper, boolean IncludeLower, boolean IncludeNum, boolean IncludeSym): Initializes the alphabet with user-selected character sets.

Main Methods

mainLoop(): Displays a menu and handles user choices in a loop.

requestPassword(): Asks user preferences, generates a password, and displays it.

checkPassword(): Accepts a password from the user and checks its strength.

printUsefulInfo(): Prints recommended password creation practices.

printMenu(): Displays the main options to the user.

printQuitMessage(): Outputs a closing message.

Helper Methods

GeneratePassword(int length): Builds a random password using the configured alphabet.

getUserPreference(String message): Gets a Yes/No response from the user and returns a boolean.

Class: Alphabet.java

The Alphabet class defines and builds a pool of characters to be used for password generation based on user preferences.

Constants

UPPERCASE_LETTERS: "ABCDEFGHIJKLMNOPQRSTUVWXYZ"

LOWERCASE_LETTERS: "abcdefghijklmnopqrstuvwxyz"

NUMBERS: "1234567890"

SYMBOLS: "!@#$%^&*_~?"

These represent the available character sets for password generation.

Modules and Imports Used

StringBuilder pool: Stores the combined character sets based on user selection.

Constructor

Alphabet(boolean uppercaseIncluded, boolean lowercaseIncluded, boolean numbersIncluded, boolean specialCharactersIncluded): Builds the character pool by conditionally appending selected character groups.

Method

getAlphabet(): Returns the final character pool as a String, which is used during password generation.

Class: GeneratorTest.java

This class contains unit tests for verifying the behavior of the Password, Alphabet, and Generator classes using **JUnit 5**.

Modules and Imports Used

org.junit.jupiter.api.Test: Enables test annotations.

org.junit.jupiter.api.Assertions.*: Provides assertion methods for writing test conditions.

Fields (Test Setup)

Password password = new Password("Secret"): A test password instance.

Alphabet firstAlphabet: Contains only uppercase letters.

Alphabet secondAlphabet: Contains lowercase letters, numbers, and symbols.

Generator generator: Configured to use only uppercase letters.

Test Methods

test1(): Verifies that the Password object returns the correct string.

test2(): Checks if firstAlphabet contains only uppercase letters.

test3(): Confirms secondAlphabet includes lowercase, numbers, and symbols.

test4(): Validates that the generator's alphabet matches the uppercase letters.

test5(): Ensures the alphabet used by the generator has a length of 26.

Class: Password.java

The Password class represents a password string and provides functionality to evaluate its strength based on specific criteria.

Field

String Value: The actual password string (immutable).

Constructor

Password(String value): Initializes a new Password object with the given string.

Methods

evaluateStrenght(): Analyzes the password and returns a message indicating whether it is **Strong**, **Fair**, or **Weak**, based on:

-Minimum of 2 uppercase letters

-Minimum of 2 lowercase letters

-Minimum of 2 digits

-Minimum of 2 special characters (from !@#$%^&*_~)

-Length greater than 8 characters

toString(): Returns the raw password string.