EX.NO.: 02
DATE: 23.06.2025

PASSWORD VALIDATION FOR GMAIL

AIM

To implement and validate a secure Gmail-style password validation system using JUnit 5 parameterized tests that dynamically read multiple test cases from an external CSV file.

ALGORITHM

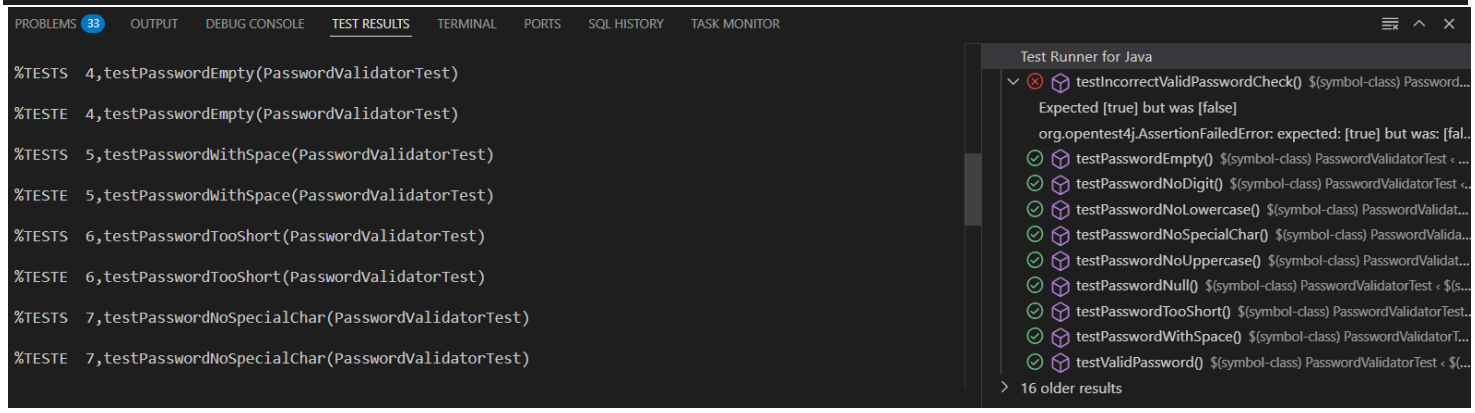
1. Define a **PasswordValidator** class containing a method `isValid(String password)` to verify password strength based on Gmail-like rules.
2. Specify the password validation criteria:
    a. Minimum 8 characters.
    b. At least one uppercase letter.
    c. At least one lowercase letter.
    d. At least one digit.
    e. At least one special character (`!@#$%^&*()-+`).
    f. No spaces allowed.
    g. Must not be null or empty.
3. Create a **CSV file (`password_test_cases.csv`)** with sample passwords and their expected validation results (`true/false`).
4. Implement a **JUnit 5 parameterized test class** (`PasswordValidatorTest`) using `@CsvFileSource` to load password values and expected results from the CSV.
5. For each test case:
6. Read the password and expected result.
7. Pass the password to the `isValid()` method.
8. Assert if the result matches the expected output.
9. Run the test suite to dynamically validate multiple passwords in a single execution.

CODE AND OUTPUT

```
public class PasswordValidator {

    public static boolean isValid(String password) {
        if (password == null || password.equals("null") || password.isEmpty()) {
            return false;
        }

        if (password.length() < 8) return false;
        if (password.contains(" ")) return false;

        boolean hasUpper = false, hasLower = false, hasDigit = false, hasSpecial =
false;
        String specials = "!@#$%^&*()-+";

        for (char c : password.toCharArray()) {
            if (Character.isUpperCase(c)) hasUpper = true;
            else if (Character.isLowerCase(c)) hasLower = true;
            else if (Character.isDigit(c)) hasDigit = true;
            else if (specials.contains(String.valueOf(c))) hasSpecial = true;
        }
```

```
        return hasUpper && hasLower && hasDigit && hasSpecial;
    }
}

import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;

class PasswordValidatorTest {

    @Test
    void testValidPassword() {
        assertTrue(PasswordValidator.isValid("Abcdef1!")); // Should pass
    }

    @Test
    void testPasswordTooShort() {
        assertFalse(PasswordValidator.isValid("Ab1!")); // Too short, should fail
    }

    @Test
    void testPasswordNoUppercase() {
        assertFalse(PasswordValidator.isValid("abcdef1!")); // No uppercase, should
fail
    }

    @Test
    void testPasswordNoLowercase() {
        assertFalse(PasswordValidator.isValid("ABCDEF1!")); // No lowercase, should
fail
    }

    @Test
    void testPasswordNoDigit() {
        assertFalse(PasswordValidator.isValid("Abcdefg!")); // No digit, should fail
    }

    @Test
    void testPasswordNoSpecialChar() {
        assertFalse(PasswordValidator.isValid("Abcdef12")); // No special char, should
fail
    }

    @Test
    void testPasswordWithSpace() {
        assertFalse(PasswordValidator.isValid("Abc def1!")); // Contains space, should
fail
    }
```

```java
    @Test
    void testPasswordNull() {
        assertFalse(PasswordValidator.isValid(null)); // Null password, should fail
    }

    @Test
    void testPasswordEmpty() {
        assertFalse(PasswordValidator.isValid("")); // Empty password, should fail
    }

    // Intentionally failing test for demonstration
    @Test
    void testIncorrectValidPasswordCheck() {
        assertTrue(PasswordValidator.isValid("abcdefg1!")); // ❌ No uppercase, should actually fail but expected true
    }
}
```



INFERENCE

This approach dynamically verifies the Password Validator against multiple real-world test cases loaded from a CSV file using JUnit 5 parameterized testing. It ensures easy scalability and separation of test data from the test logic.