HomeWork 1:

Consiste di progettare e implementare una suite di test black-box (specification-based testing) per un codice a vostra scelta. La classe scelta è PassowordValidator.

Questa classe ha lo scopo di verificare la validità di una password secondo alcuni criteri specificati. La classe dispone di tre attributi privati che sono:

* "minLength" (lunghezza minima richiesta per la password),
* "requiresNumber" (indica se è richiesto almeno un numero nella password)
* "requiresSpecialChar" (indica se è richiesto almeno un carattere speciale nella password).

Sono presenti i costruttori:

* uno che accetta tre parametri per inizializzare gli attributi sopra citati
* uno vuoto, probabilmente utilizzato per istanze in cui i criteri di validazione non sono rilevanti.

Sono presenti i metodi:

* Validate: prende una stringa "password" e restituisce un booleano dopo aver fatto diverse verifiche sulla password controllando:
  + se la password è nulla
  + poi se la sua lunghezza è inferiore alla "minLength".
  + Verifica se, in base alle richieste, è presente almeno un numero e\o un carattere speciale

Scopo del homework1:

Il testing basato sulle specifiche è una metodologia di testing del software in cui i test sono progettati sulla

base delle specifiche dei requisiti del sistema. Avviene quindi che invece di concentrarsi sulla logica interna o sulle implementazioni tecniche del software si concentra sulle caratteristiche e sul comportamento stesso del sistema. Infatti i test vengono vengono progettati per verificare se il software soddisfa le specifiche dei requisiti che includere informazioni come gli input attesi, gli output desiderati, i vincoli di prestazione e altre caratteristiche funzionali e non funzionali del sistema.

Questo per garantire l'obbiettivo principale del testing basato sulle specifiche è garantire che il software sia conforme alle specifiche dei requisiti e che funzioni come previsto.

Nel processo di testing basato sulle specifiche, vengono progettati casi di test che coprono diverse

combinazioni di input e scenari di utilizzo in base alle specifiche dei requisiti. I risultati dei test vengono

quindi confrontare con i risultati attesi per determinare se il software sta funzionando correttamente.

L’approccio a 7 step;

1. Understanding the requirements
   1. Il programma deve:

* Validare le password in base a determinati criteri per restituire il risultato della validazione: Il programma restituisce true se tutti i criteri specificati sono soddisfatti, indicando che la password è valida. Se uno qualsiasi dei criteri non è soddisfatto, il programma deve restituire false.
* Verifica la Lunghezza della Password: deve verificare se la lunghezza della password fornita è maggiore o uguale a una lunghezza minima specificata dalla variabile minLength
* Verifica se è soddisfatta il requisito della presenza di valori numeri: Tramite il flag requiresNumber che se è impostato su true, il programma deve controllare se la password contiene almeno un valore numerico.
* Verifica se è soddisfatta il requisito della presenza di caratteri speciali: Se il flag requiresSpecialChar è impostato su true, il programma deve controllare se la password contiene almeno un carattere speciale da un insieme predefinito (!@#$%^&\*()\_+-=[]{};':"\|,.<>/?).
* Gestione della password in caso di valore null: Il programma deve gestire il caso in cui la password fornita sia null, restituendo false per indicare che una password null non è valida.
  1. Il programma non deve:
* Modificare le Password perché il programma deve solo validare le password e non deve modificare la stringa della password o altri dati.
* Effettuare l'Autenticazione perché il programma si occupa solo della validazione delle password e quindi non deve essere utilizzato nell'autenticazione degli utenti o nella gestione degli account.
* Archiviare o Trasmettere le Password perchè non deve archiviare le password né trasmetterle su una rete.
* Interagire con l'Interfaccia Utente perché Il programma non interagisce con le interfacce utente, non acquisisce l'input dell'utente e non visualizza messaggi poiché si concentra solo sulla validazione delle password.
* Crittografare o Hashare le Password perché non deve gestire la crittografia, l'hashing o qualsiasi forma di archiviazione sicura delle password.

1. Quali sono i tipi di variabili degli input e gli output?
   * 1. INPUT

La classe accetta:

* + - * minLength di tipo int ed è la lunghezza minima richiesta per la password.
      * requiresNumber di tipo boolean e indica se la password richiede almeno un numero.
      * requiresSpecialChar di tipo boolean e indica se la password richiede almeno un carattere speciale.
    1. OUTPUT
       - Il metodo validate(String password) restituisce un valore booleano
         1. True: indica se la password fornita soddisfa tutti i criteri specificati.
         2. False: indica se la password fornita soddisfa tutti i criteri specificati.
  1. Variabili Coinvolte:
     1. Integer: minLength
     2. Boolean: requiresNumber, requiresSpecialChar
     3. String: password,// originalStr
  2. Il dominio dell’Input
* minLength è un valore intero che rappresenta la lunghezza minima della password.
* requiresNumber e requiresSpecialChar sono booleani che indicano se la password deve contenere un numero o un carattere speciale.

1. Explore what the program does for various inputs

Per esplorare il comportamento del programma nei vari input, sono stati pensati:

* Diversi scenari e valori di input
* Risposta attesa

I casi di test sono:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| minLenght | RequiresNumber | requiresSpecialChar | password | Esito Previsto | Esito Atteso |
| password.length() >= 8 && password.length() <= 24 |  |  | ciaomichiamonico | si | si |
| password.length() >= 8 && password.length() <= 24 |  |  | Parabola1 | si | si |
| password.length() >= 8 && password.length() <= 24 |  |  | Mipiaceilcioccolato | si | si |
|  | True |  | ciao1 | Si | Si |
|  | True |  | Parabola1 | Si | Si |
|  | True |  | 123456 | Si | si |
|  |  | True | ciao@ | Si | Si |
|  |  | True | Parabola1? | Si | Si |
|  |  | True | @ | Si | Si |
|  |  |  | “empty” | Si | Si |
|  |  |  | “null” | Si | si |
| password.length() >= 8 && password.length() <= 24 | True | True | Pianura123? | Si | Si |
| password.length() >= 8 && password.length() <= 24 | True | True | "Parabola1@" | Si | Si |
| password.length() >= 8 && password.length() <= 24 | True | True | Catastrofe99\* | Si | Si |
| password.length() >= 8 && password.length() <= 24 | True | True | Pasquale001@ | Si | si |

1. Explore inputs, outputs and identify partitions
   1. INPUT
      1. minLength: Intero (int)

Intervallo: Interi positivi maggiori di zero.

Dipendenze: non ha nessuna dipendenza

* + 1. requiresNumber: Booleano (boolean)

Valori: true se sono richiesti numeri o false se non lo sono richiesti

Dipendenze: non ha nessuna dipendenza

* + 1. requiresSpecialChar: Booleano (boolean)

Valori: true se sono richiesti caratteri speciali o false se non lo sono richiesti Dipendenze: non ha nessuna dipendenza

* + 1. password (usata come input in vari metodi di test): Stringa (String)

Valori: stringhe fornite nei metodi di test.

Dipendenze: non ha nessuna dipendenza

* 1. Output
     1. validate(String password): Booleano (boolean)

Restituisce true se la password soddisfa tutti i criteri specificati, altrimenti restituisce false.

Dipendenze: stringa di password, minLength, requiresNumber e requiresSpecialChar.

* 1. identify partitions
     1. Lunghezza della password minLength:
        + Valori positivi superiore a 1
        + Valorni negativi: non ammessi
        + Zero: non ammesso
     2. requiresNumber
        + true: il programma verifica la presenza di numeri
        + false: il programma verifica che non sono presenti numeri
     3. Presenza di caratteri speciali:
        + true: il programma verifica la presenza di caratteri speciali
        + false: il programma verifica che non sono prensenti caratteri speciali
     4. password
        + Stringa vuota: “”
        + Stringa nulla: “null”
     5. Interazioni tra variabili

I vari metodi di test (testPasswordLengthRequirement, testNumberRequirement, ecc.) interagiscono con la variabile password e verificano se essa soddisfa i criteri specificati in base ai valori di input di minLength, requiresNumber e requiresSpecialChar.

* + 1. Possibili tipi di output

L’output dei metodi di test è un valore booleano e indica che se la passord è valida

1. Identify boundary cases

Occorre osservare gli input e le partizioni precedentemente individuate e identificare i casi in cui gli input si trovano ai limiti delle partizioni. Questi sono spesso i punti critici in cui possono verificarsi errori o comportamenti inattesi.

Assolutamente, identificare i casi limite è fondamentale per rilevare eventuali bug. Ecco una panoramica dei casi limite per le diverse parti del tuo codice:

* testPasswordLengthRequirement:
  1. Boundary Case: Lunghezza minima consentita (minLength = 8).
* testNumberRequirement:

1. Boundary Case: Password con un requisito: presenza di numeri

* testSpecialCharacterRequirement:

1. Boundary Case: Password con un requisito: presenza di caratteri speciali

* testEmptyPassword:

1. Boundary Case: Password vuota ("").

* testNullPassword:

1. Boundary Case: Password nulla (null).

* testValidPassword:

1. Boundary Case: Lunghezza minima e almeno un solo numero e un carattere speciale (soddisfa il requisito inseriti).
2. Automate test cases

|  |  |
| --- | --- |
| T1 | public void testPasswordLengthRequirement(String password) {  Assertions.*assertTrue*(password.length() >= 8 && password.length() <= 24); } |
| T2 | public void testNumberRequirement(String password) {  Assertions.assertTrue(password.matches(".\*\\d.\*"));  } |
| T3 | public void testSpecialCharacterRequirement(String password) {  Assertions.assertTrue(password.matches(".\*[!@#$%^&\*()\_+\\-=\\[\\]{};':\"\\\\|,.<>\\/?].\*"));  } |
| T4 | public void testEmptyPassword(String password) {  Assertions.assertFalse(passwordValidator.validate(password));  } |
| T5 | public void testNullPassword(String password) {  Assertions.assertFalse(passwordValidator.validate(password));  } |
| T6 | public void testValidPassword(String password) {  Assumptions.assumeTrue(password.matches(".\*[a-z].\*"));  Assertions.assertAll(  () -> Assertions.assertTrue(password.length() >= 8 && password.length() <= 24, "Lunghezza minima non rispettata"),  () -> Assertions.assertTrue(password.matches(".\*\\d.\*"), "Assenza di almeno 1 numero"),  () -> Assertions.assertTrue(password.matches(".\*[!@#$%^&\*()\_+\\-=\\[\\]{};':\"\\\\|,.<>\\/?].\*"), "Assenza di almeno 1 carattere speciale"));  } |

1. Augment the test suite with creativity and experience

La suite di test sembra completa e copre diversi casi per verificare il corretto funzionamento della classe `PasswordValidator`.

In questa sezione, abbiamo inserito alcuni test che si potevamo aggiungere per cercare di avere una copertura maggiore di controllo.

Alcune di queste idee possono essere:

|  |  |
| --- | --- |
| * Test di invalidità con caratteri speciali: Si può aggingere un test in cui la password contiene solo caratteri speciali ma non soddisfa gli altri requisiti. | Il test deve fallire |
| * Test di validità: Se la password inserita ha la lunghezza e requisiti abilitati ma non sono presenti caratteri | Il test deve fallire |
| * Test con `minLength` negativo: Bisogna verificare che è presente un valore negativo per `minLength` sollevi un'eccezione appropriata | Il test deve fallire |
| * Test con `null` come Valore di Requisiti: Verifica che passare `null` come valore per `requiresNumber` o `requiresSpecialChar` | Il test deve fallire |
|  |  |
|  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | originalStr | minLength | requiresNumber | requiresSpecialChar | Scopo | expectedStr | Risultato  Effettivo |
| T1 | Null | 8 | Falso | falso | La stringa è di tipo null | Falso | Falso |
| T2 | Vuoto | 8 | Falso | Falso | La stringa è di tipo empty | Falso | Falso |
| T3 | Parabola | 6 | Falso | Falso | La stringa deve contenere 6 caratteri minimi con nessun requisito | Vero | Vero |
| T4 | Palo | 6 | Falso | falso | La stringa deve contenere 6 caratteri minimi con nessun requisito | falso | FALSO |
| T5 | Mitico | 6 | Falso | Falso | La stringa deve contenere 6 caratteri minimi con nessun requisito | True | VERO |
| T6 | Parabola | 6 | True | False | La stringa deve contenere 6 caratteri minimi e deve avere almeno un valore numerico | False | FALSO |
| T7 | Parabola1 | 6 | True | False | La stringa deve contenere 6 caratteri minimi e deve avere almeno un valore numerico | True | VERO |
| T8 | Parabola | 6 | Falso | Falso | La stringa deve contenere 6 caratteri minimi con nessun requisito | vero | VERO |
| T9 | Parabola1 | 6 | Falso | FALSO | La stringa deve contenere 6 caratteri minimi con nessun requisito | Falso | VERO |
| T10 | parabola | 6 | Falso | Vero | La stringa deve contenere 6 caratteri minimi e deve avere almeno un carattere speciale | falso | FALSO |
| T11 | Parabola@ | 6 | Falso | Vero | La stringa deve contenere 6 caratteri minimi e deve avere almeno un carattere speciale | Vero | VERO |
| T12 | Parabola | 6 | Falso | falso | La stringa deve contenere 6 caratteri minimi con nessun requisito | vero | VERO |
| T13 | parabola@ | 6 | FALSO | FALSO | La stringa deve contenere 6 caratteri minimi con nessun requisito | Falso | VERO |

Attenzione: I test T9 e T13 restituiscono un valore aspettato diverso da quello desiderato a causa di un errore di progettazione.