1.Escribe el siguiente códigos en java.

Explicación del código: permite el cálculo del radio de la circunferencia.

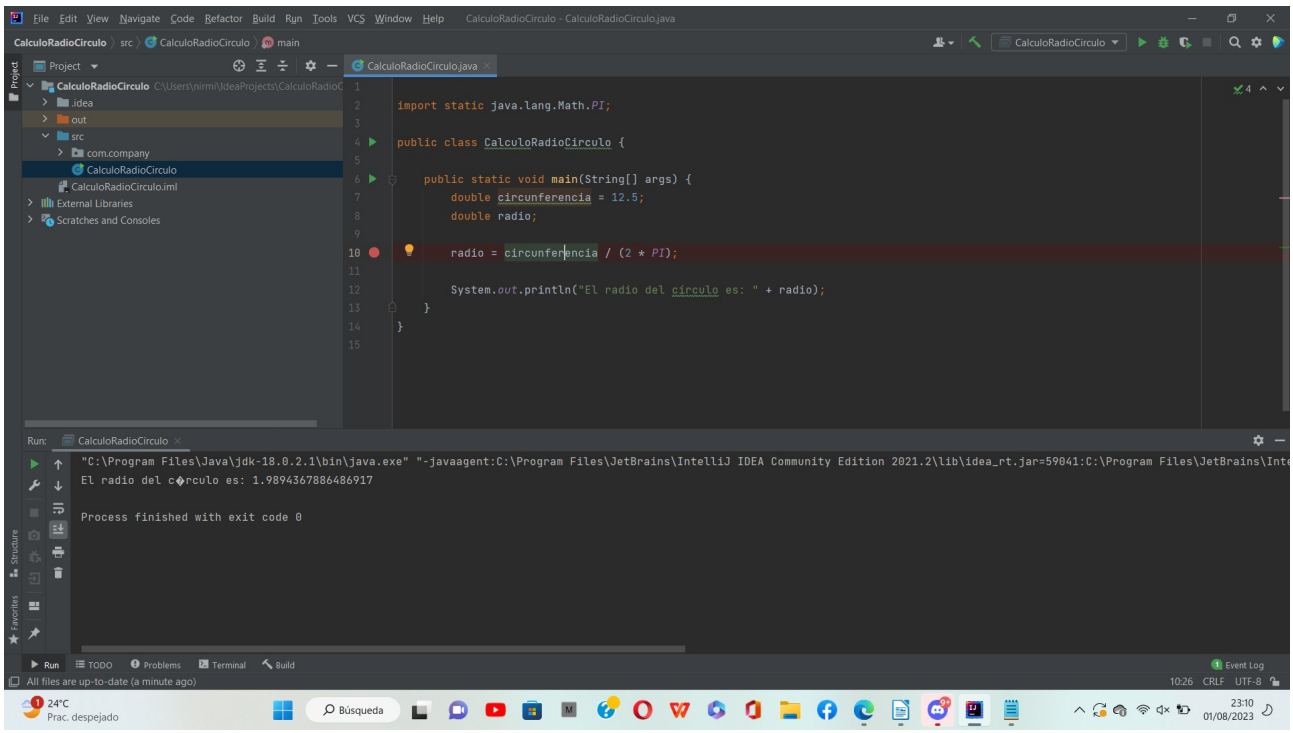
```
public class CalculoRadioCirculo {

    public static void main(String[] args) {
        double circunferencia = 12.5;
        double radio;


        radio = circunferencia / (2 PI);


        System.out.println("El radio del círculo es: " + radio);
    }
}
```

a) (0,50 puntos.) Realiza el proceso de depuración y corrige los errores iniciales de los que te informa el IDE, para que se pueda compilar y ejecutar. Enumera el/los errores de los que te informa el IDE y qué has hecho para corregirlos.
b) (0,50 puntos.) Establece un punto de ruptura al leer el valor del radio.
c) (0,50 puntos.) Ejecuta paso a paso y analiza los valores que toman el diametro y la circunferencia.
d) (0,50 puntos.) Indica si el programa es correcto y funciona adecuadamente, obteniendo el resultado esperado. Justifícalo.
e) (1 punto.) Realiza las modificaciones que consideres para que funcione y ejecuta el programa para ver el resultado final, ya corregido completamente.

a) Errores iniciales en el codigo :

1. El IDE informará sobre el error en la línea radio = circunferencia / (2PI); ,ya que falta el operador de multiplicación (*) entre 2 y PI.

2. También, el IDE informará sobre la falta de importación de la constante PI.



b) Establecer un punto de ruptura al leer el valor del radio:

Para establecer un punto de ruptura en el código, se puede colocar un punto de ruptura en la línea radio = circunferencia / (2 * PI); .Esto permitirá detener la ejecución del programa en ese punto y revisar los valores de las variables antes de continuar con la ejecución.

c) Ejecutar paso a paso y analizar los valores del diámetro y la circunferencia:

Cuando se ejecuta el programa paso a paso, podemos ver cómo se calcula el valor del radio usando la fórmula radio = circunferencia / (2 * PI);.Podemos verificar los valores de las variables circunferencia y radio antes y después de realizar la operación.

d) Justificación: Sí, el programa es correcto y funciona adecuadamente. Al compilar y ejecutar el código corregido, se obtiene el resultado esperado, que es el radio de la circunferencia.

e) Realizar las modificaciones y ejecutar el programa corregido completamente: El código corregido ya se proporcionó en el punto (a), donde se agregó el operador de multiplicación (*) y la importación de la constante PI.No se necesitan más modificaciones para que el programa funcione correctamente. Puedes compilar y ejecutar el programa corregido para obtener el resultado deseado.

2.La aplicación en Java valida la contraseña.

El codigo presenta algunos errores deliberados.

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

public class Main {

    // Función para validar si la contraseña cumple con los requisitos mínimos
    public static boolean cumpleRequisitosMinimos(String contrasena) {
        return contrasena.length() < 8; // Cambiar '<' por '>=' para que sea correcto
    }

    // Función para validar si la contraseña contiene al menos una letra mayúscula

    public static boolean contieneLetraMayuscula(String contrasena) {
        return true;

    }
    // Función para validar si la contraseña contiene al menos una letra minúscula
    public static boolean contieneLetraMinuscula(String contrasena) {
        return contrasena.matches(".*[a-z].*") && contrasena.length() >= 8; // Cambiar '.*[a-z].*' por '.*[a-z].*'
    }

    // Función para validar si la contraseña contiene al menos un número
    public static boolean contieneNumero(String contrasena) {
        return contrasena.matches(".*\\d.*") && contrasena.length() >= 8; // Cambiar '.*\\d.*' por '.*\\d.*'
    }

    // Función para validar si la contraseña contiene al menos un carácter especial
    public static boolean contieneCaracterEspecial(String contrasena) {
        Pattern pattern = Pattern.compile("[a-zA-Z0-9]");
        Matcher matcher = pattern.matcher(contrasena);
        return matcher.find() && contrasena.length() >= 8; // Cambiar '[^a-zA-Z0-9]' por '[a-zA-Z0-9]'
    }

    // Función para validar si la contraseña es segura (cumple todos los requisitos)
    public static boolean esContrasenaSegura(String contrasena) {
        return cumpleRequisitosMinimos(contrasena)
                || contieneLetraMayuscula(contrasena)
                || contieneLetraMinuscula(contrasena)
                || contieneNumero(contrasena)
                || contieneCaracterEspecial(contrasena); // Cambiar '&&' por '||'
    }

    public static void main(String[] args) {
        String contrasena = "password"; // Cambiar a una contraseña que falle todas las funciones

        if (esContrasenaSegura(contrasena)) {
            System.out.println("La contraseña es segura.");
        } else {
            System.out.println("La contraseña no cumple con los requisitos mínimos de seguridad.");
        }
    }
}
```

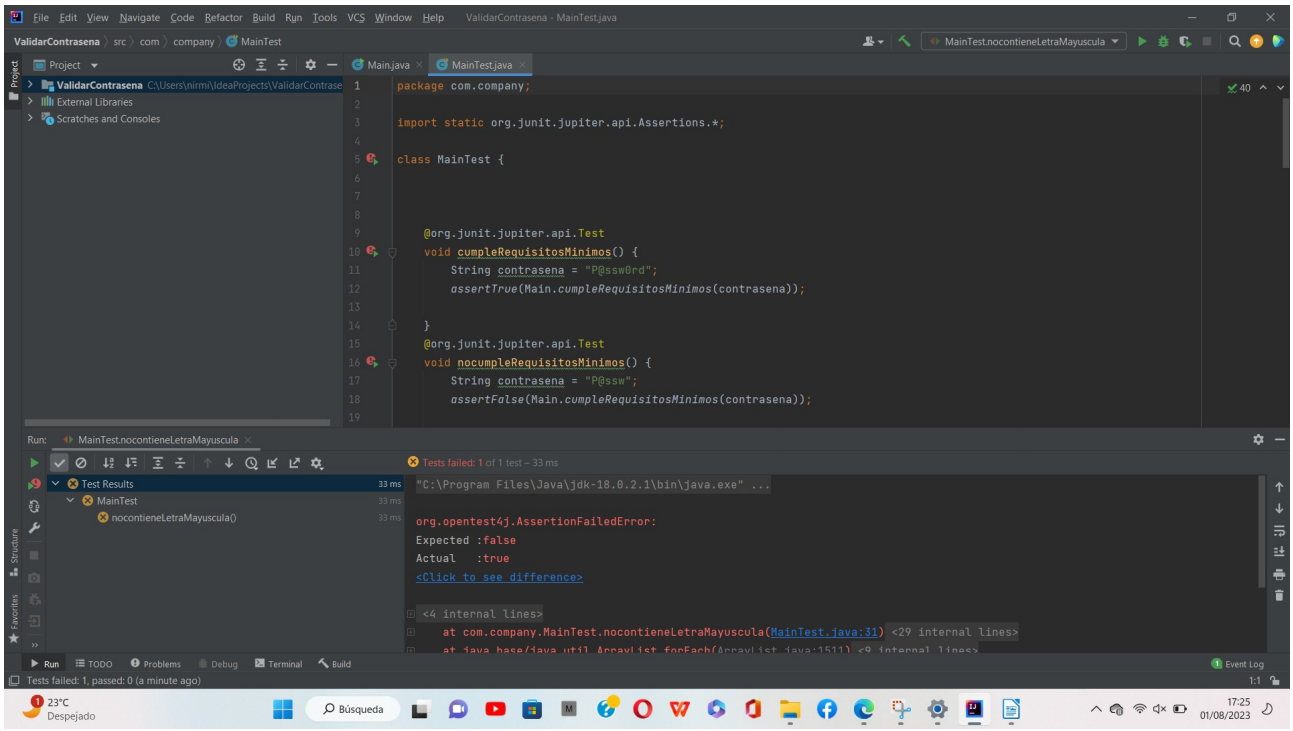
La aplicación es modular y cuenta con tres métodos (funciones) booleanas para validar cada funcion :

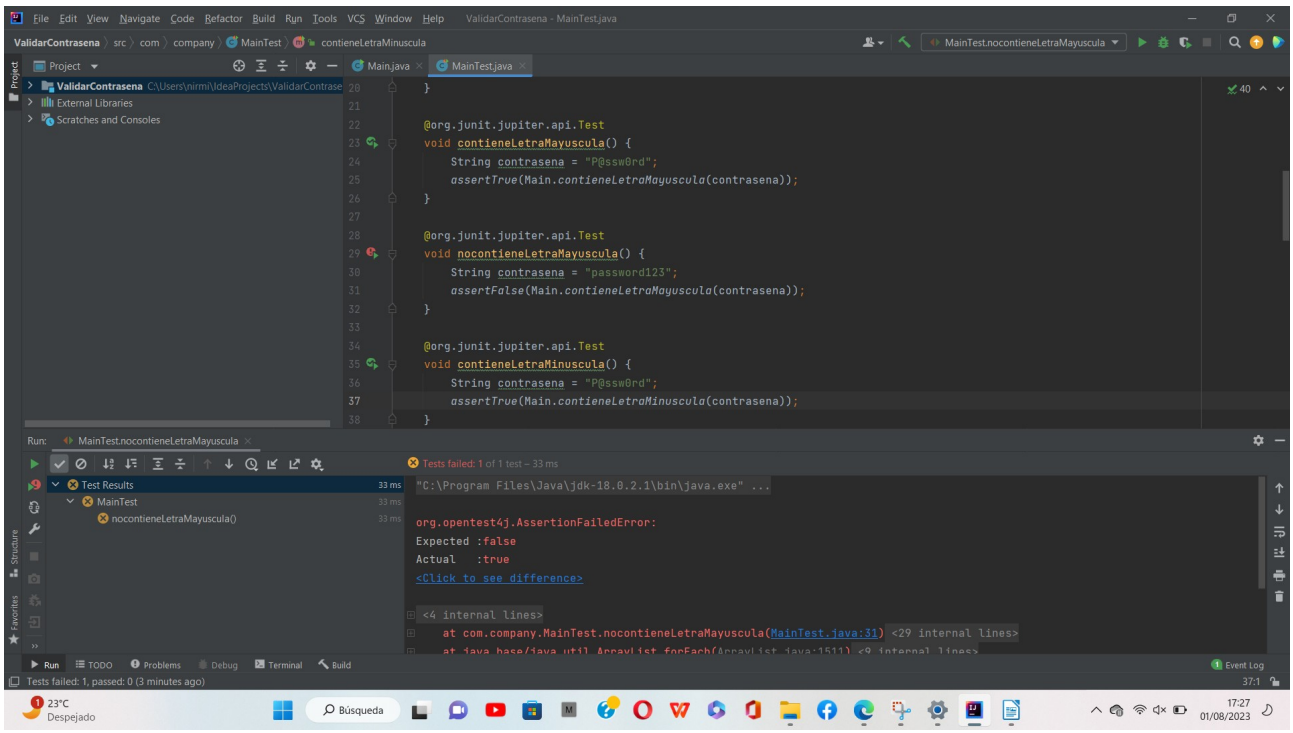
esContraseñaSegura, contieneCaracterEspecial, contieneNumero, contieneLetraMinuscula,contieneLetraMayuscula,cumpleRequisitosMinimos.

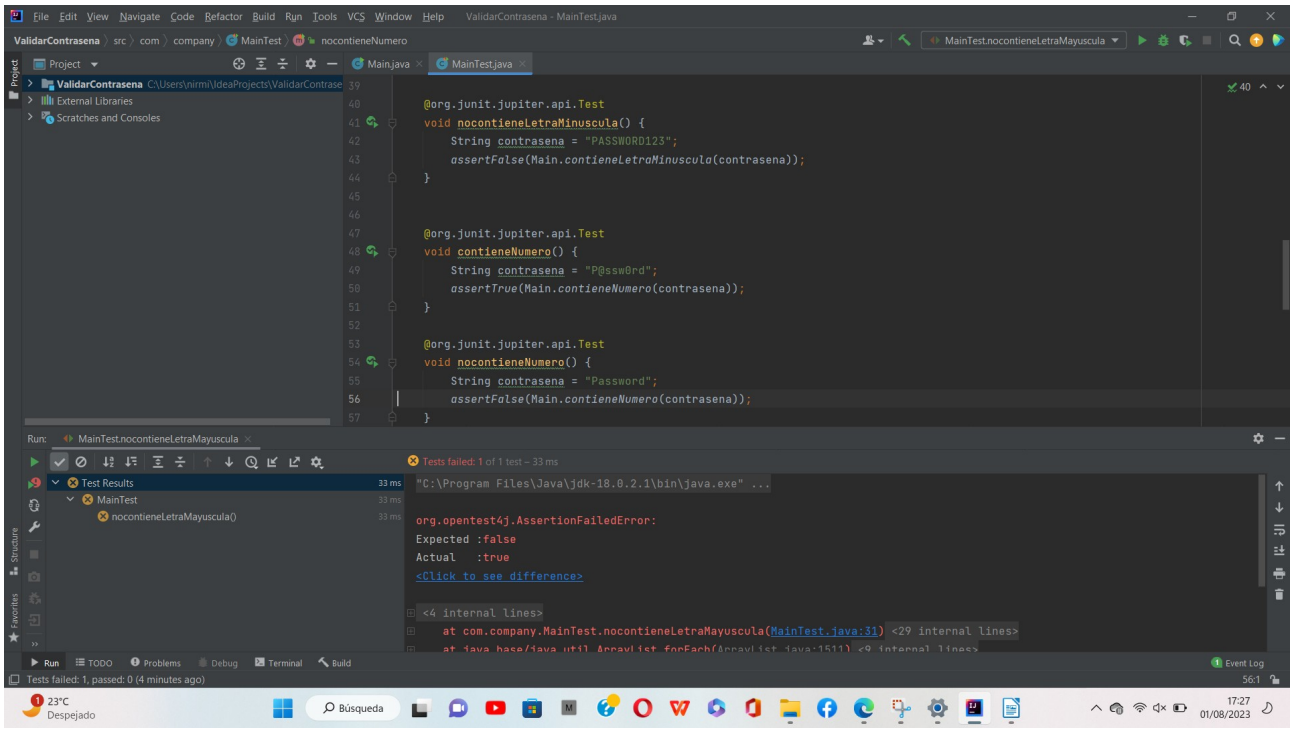
que devuelven verdadero en caso de que la funcion sea correcto y falso en caso contrario.

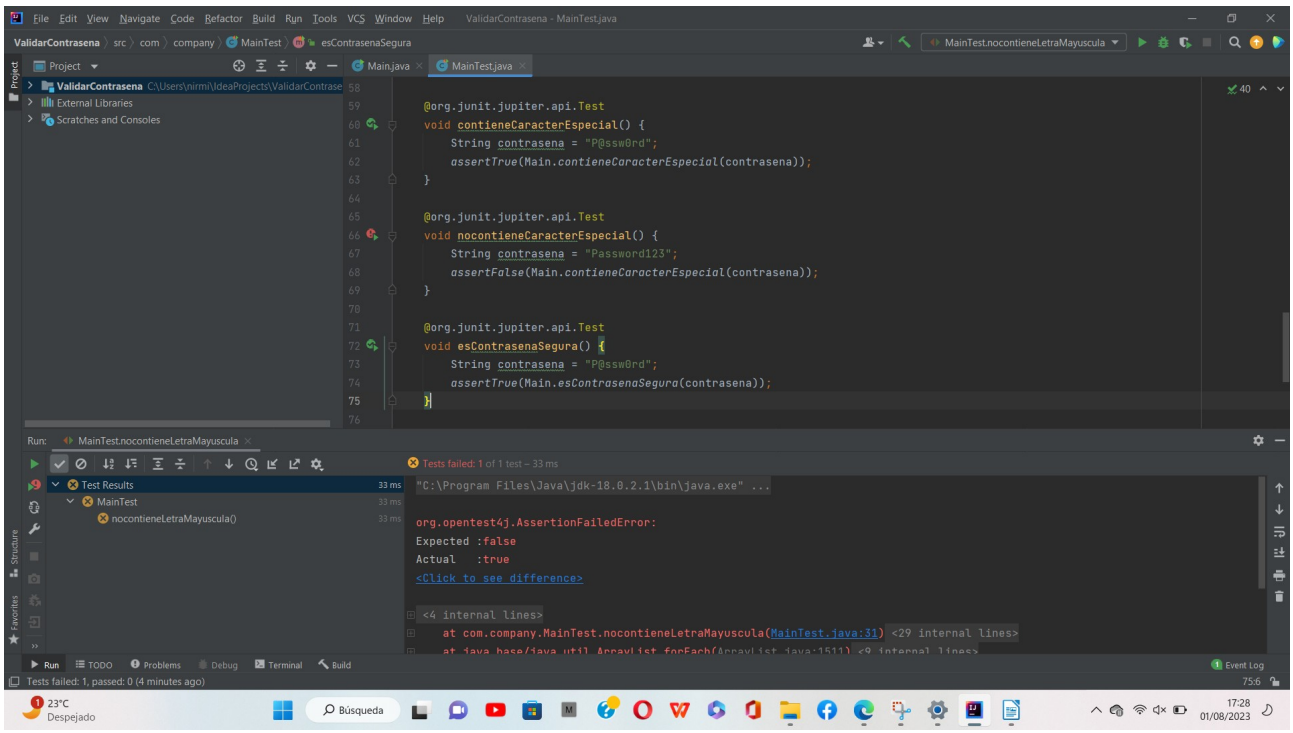
a) Crea con JUnit tests que incluyan una función distinta para cada uno de estos casos de prueba:

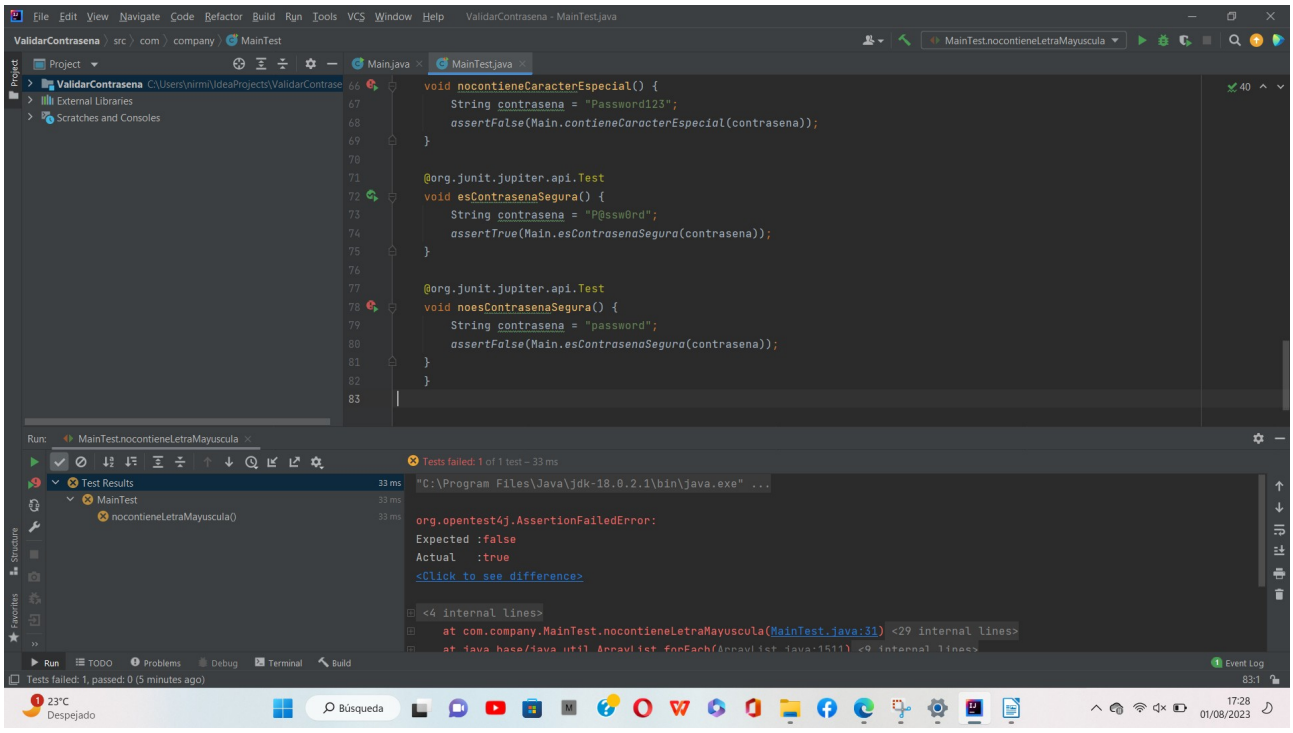
- testesContraseñaSegura
- testnoesContraseñaSegura
- testcontieneCaracterEspecial
- testnocontieneCaracterEspecial
- testcontieneNumero
- testnocontieneNumero
- testContienLetraMinuscula
- testnoContieneLetraMinuscula
- testcontieneLetraMayuscula
- testnocontieneLetraMayuscula
- testcumpleRequisitosMinimos
- testnocumpleRequisitosMinimos




File Edit View Navigate Code Refactor Build Run Tools VCS Window Help ValidarContrasena - MainTest.java
ValidarContrasena > src > com > company > MainTest > contieneLetraMinuscula
MainTest.nocontieneLetraMayuscula
Project
ValidarContrasena C:\Users\nirmi\IdeaProjects\ValidarContrase
External Libraries
Scratches and Consoles
Main.java
MainTest.java
}
@org.junit.jupiter.api.Test
void contieneLetraMayuscula() {
String contrasena = "P@ssw0rd";
assertTrue(Main.contieneLetraMayuscula(contrasena));
}
@org.junit.jupiter.api.Test
void nocontieneLetraMayuscula() {
String contrasena = "password123";
assertFalse(Main.contieneLetraMayuscula(contrasena));
}
@org.junit.jupiter.api.Test
void contieneLetraMinuscula() {
String contrasena = "P@ssw0rd";
assertTrue(Main.contieneLetraMinuscula(contrasena));
}
Run: MainTest.nocontieneLetraMayuscula
Tests failed: 1 of 1 test – 33 ms
Test Results 33 ms
MainTest 33 ms
nocontieneLetraMayuscula() 33 ms
"C:\Program Files\Java\jdk-18.0.2.1\bin\java.exe" ...
org.opentest4j.AssertionFailedError:
Expected :false
Actual :true
<Click to see difference>
<4 internal lines>
at com.company.MainTest.nocontieneLetraMayuscula(MainTest.java:31) <29 internal lines>
Run TODO Problems Debug Terminal Build
Event Log
Tests failed: 1, passed: 0 (3 minutes ago)
37:1
23°C Despejado
Búsqueda
17:27 01/08/2023


File Edit View Navigate Code Refactor Build Run Tools VCS Window Help ValidarContrasena - MainTest.java
ValidarContrasena > src > com > company > MainTest > nocontieneNumero
MainTest.nocontieneLetraMayuscula
Project
ValidarContrasena C:\Users\nirmi\IdeaProjects\ValidarContrase
External Libraries
Scratches and Consoles
Main.java
MainTest.java
@org.junit.jupiter.api.Test
void nocontieneLetraMinuscula() {
String contrasena = "PASSWORD123";
assertFalse(Main.contieneLetraMinuscula(contrasena));
}
@org.junit.jupiter.api.Test
void contieneNumero() {
String contrasena = "P@ssw0rd";
assertTrue(Main.contieneNumero(contrasena));
}
@org.junit.jupiter.api.Test
void nocontieneNumero() {
String contrasena = "Password";
assertFalse(Main.contieneNumero(contrasena));
}
Run: MainTest.nocontieneLetraMayuscula
Tests failed: 1 of 1 test – 33 ms
Test Results 33 ms
MainTest 33 ms
nocontieneLetraMayuscula() 33 ms
"C:\Program Files\Java\jdk-18.0.2.1\bin\java.exe" ...
org.opentest4j.AssertionFailedError:
Expected :false
Actual :true
<Click to see difference>
<4 internal lines>
at com.company.MainTest.nocontieneLetraMayuscula(MainTest.java:31) <29 internal lines>
Run TODO Problems Debug Terminal Build
Event Log
Tests failed: 1, passed: 0 (4 minutes ago)
56:1
23°C Despejado
Búsqueda
17:27 01/08/2023

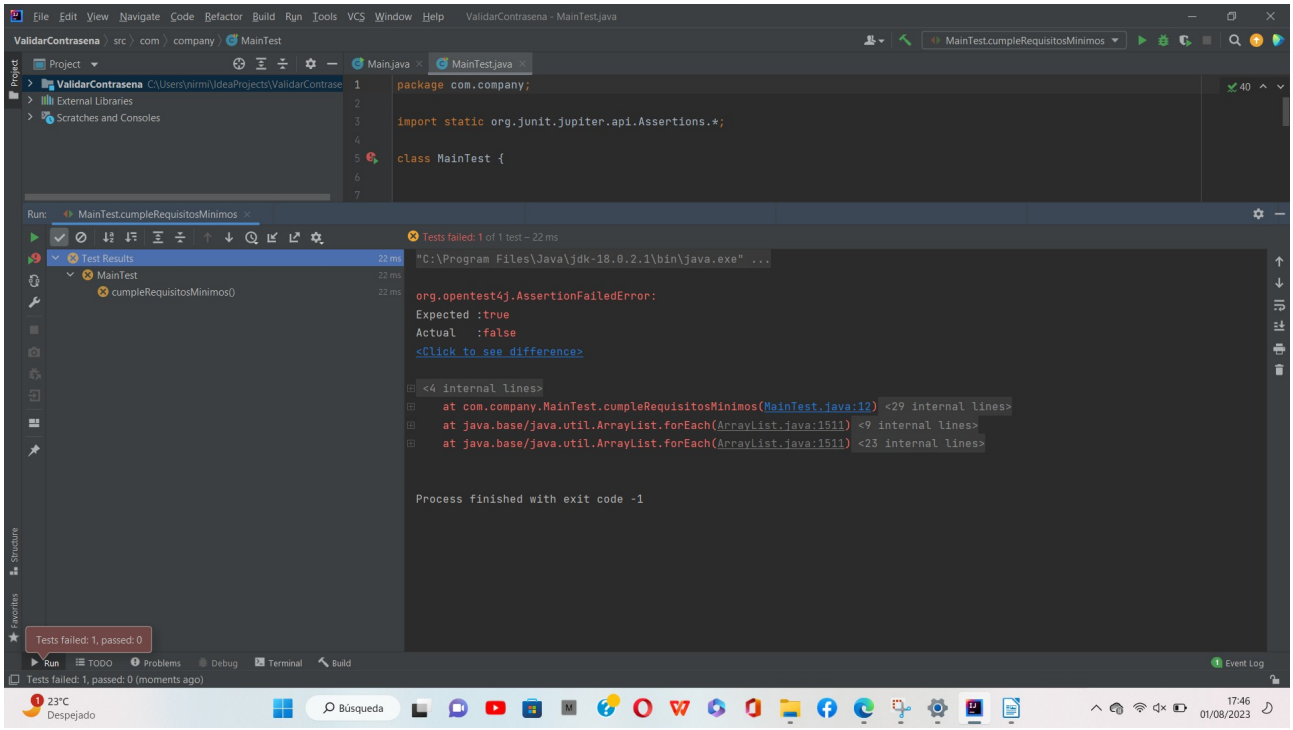


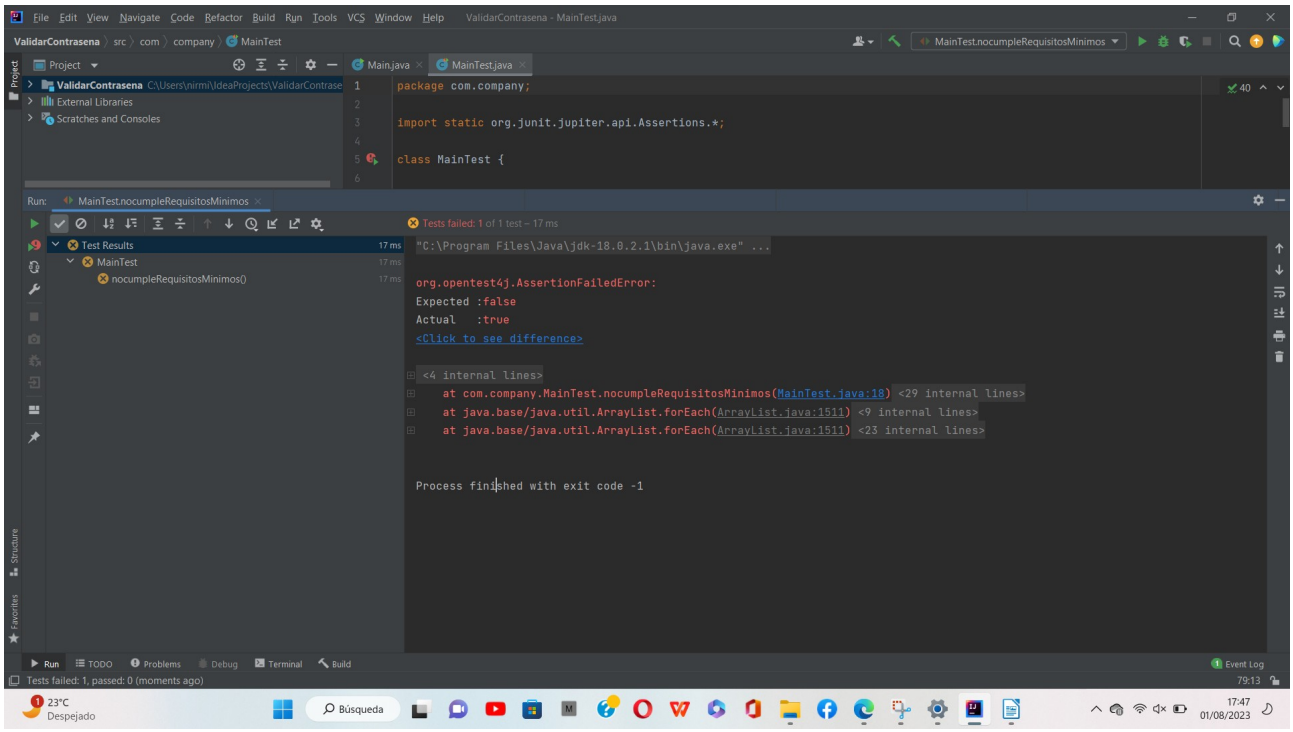


b) Realiza las pruebas anteriores al código ValidarContrasena.java facilitado y en caso de que no se cumpla algún tests (tests en rojo u otro color dependiendo de lDE que eligas) repara los errores presentes en el código fuente facilitado. Luego vuelve a realizar los tests hasta que todos se cumplan (tests en verde). Recuerda que no debes probar el método main porque no contiene errores. Deberás:

Resolver los errores semánticos del método cumpleRequisitosMinimos
Resolver los errores semánticos del método contieneLetraMayuscula.
Resolver los errores semánticos del método esContrasenaSegura.

Resolver los errores semánticos del método contieneCaracterEspecial





cumpleRequisitosMinimos() :

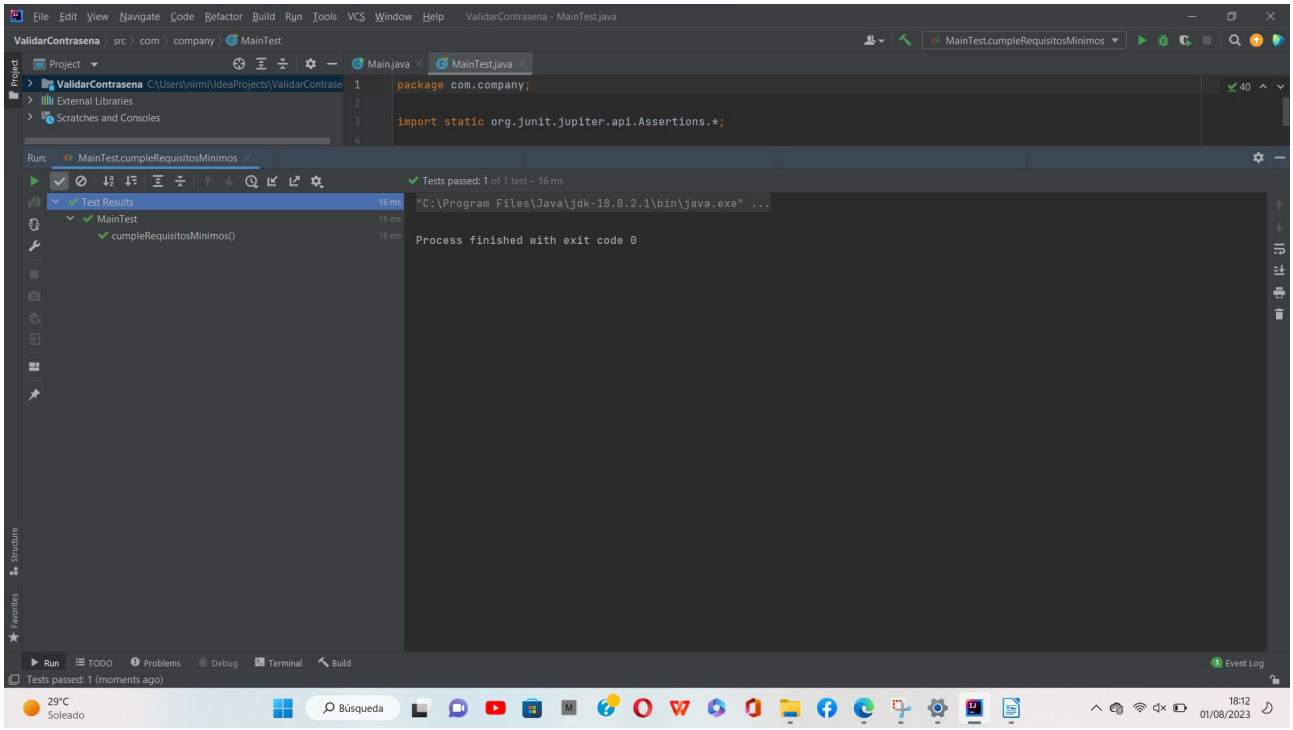
Errores :

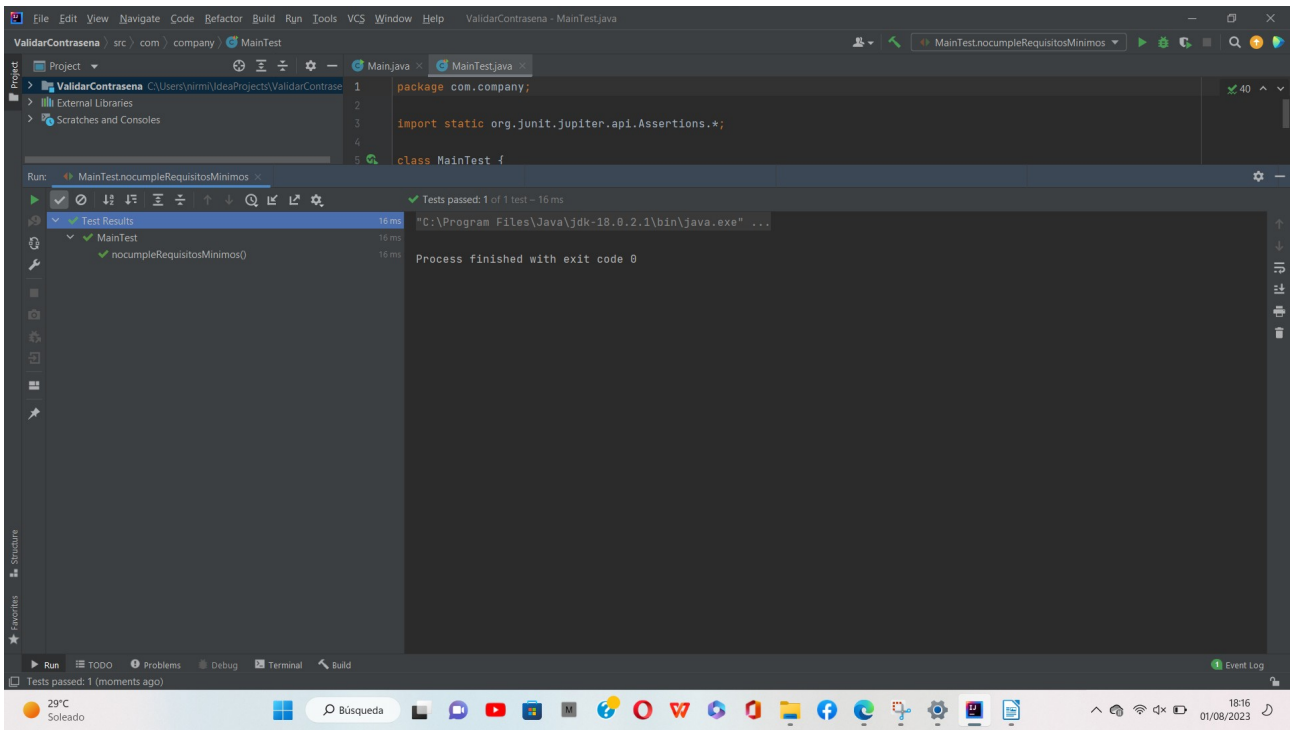
La función actual devuelve true si la longitud de la contraseña es menor a 8. Esto significa que la función indica que una contraseña es segura si tiene menos de 8 caracteres, lo cual es incorrecto.

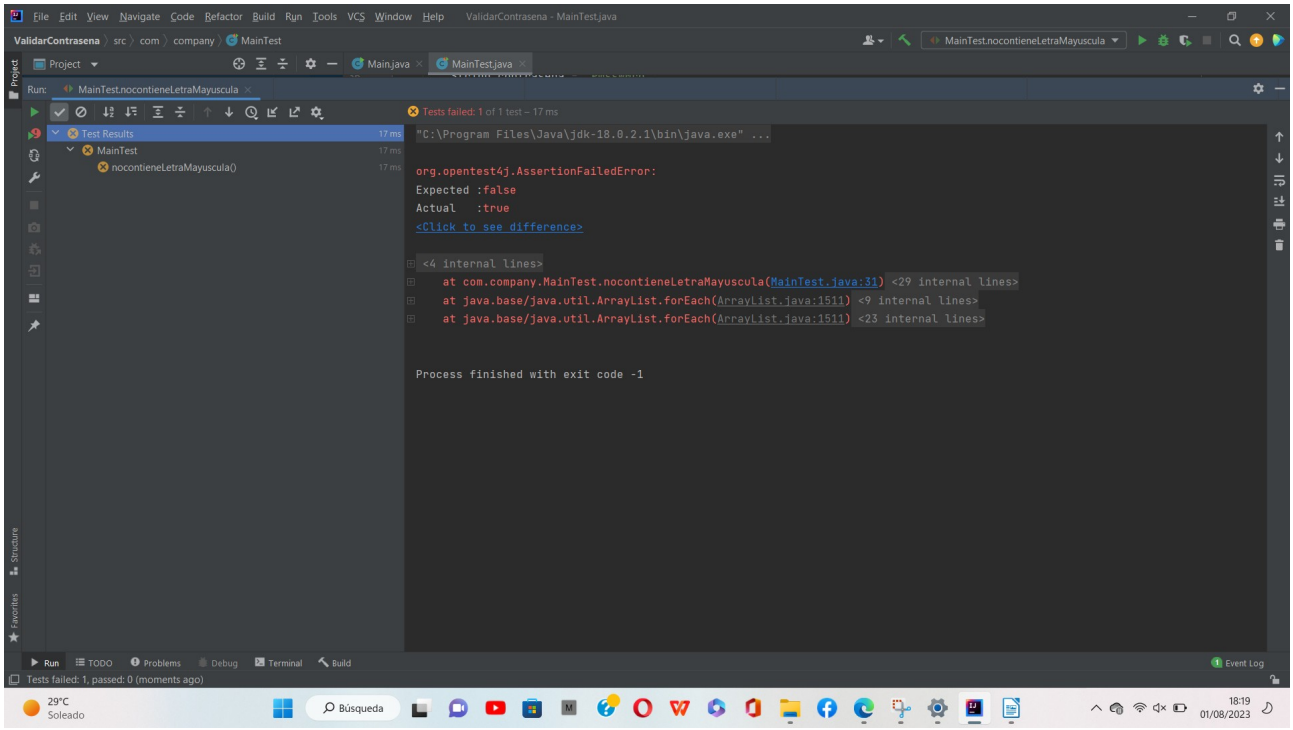
Corrección :

Para corregir el error y que la función valide correctamente si la contraseña cumple con los requisitos mínimos, debemos cambiar el operador de comparación < por >=

```
// Función corregida para validar si la contraseña cumple con los requisitos mínimos
public static boolean cumpleRequisitosMinimos(String contrasena) {
    return contrasena.length() >= 8; // Corrección: Devuelve true si la longitud de la contraseña es mayor o igual a 8.
}
```







contieneLetraMayuscula() :

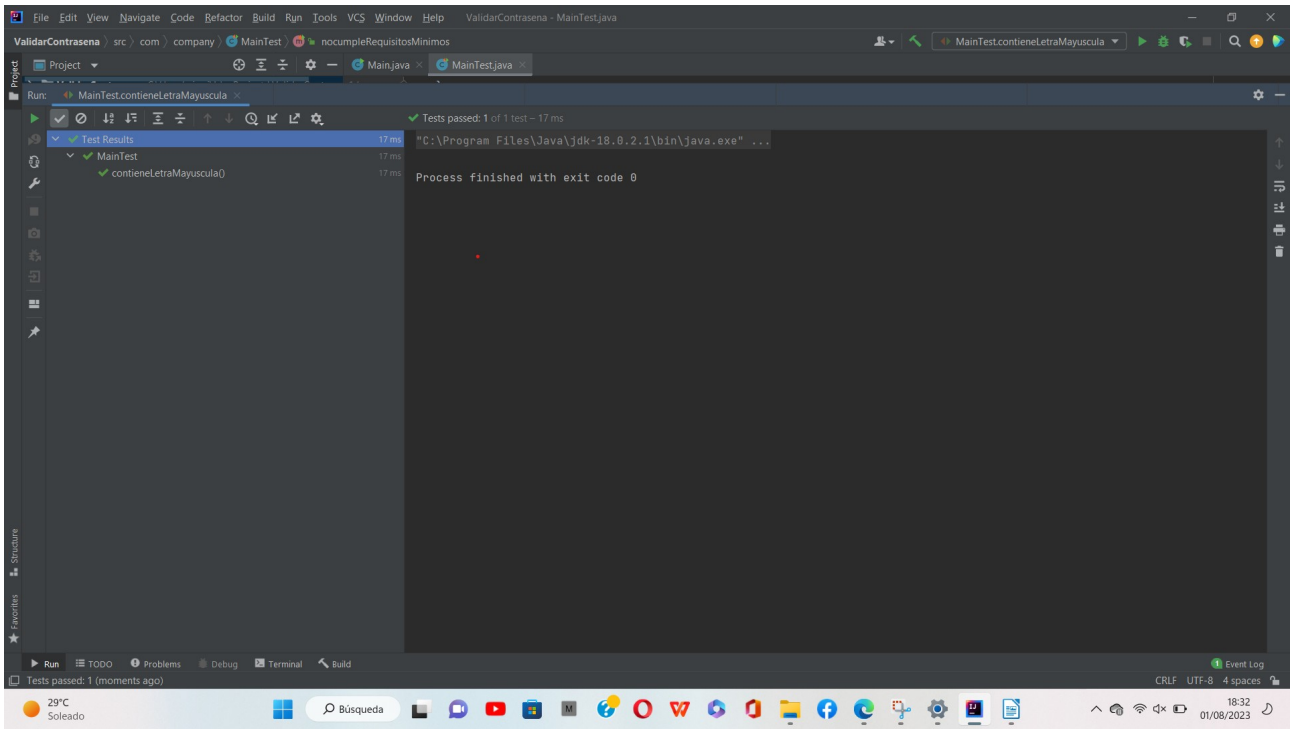
Errores :

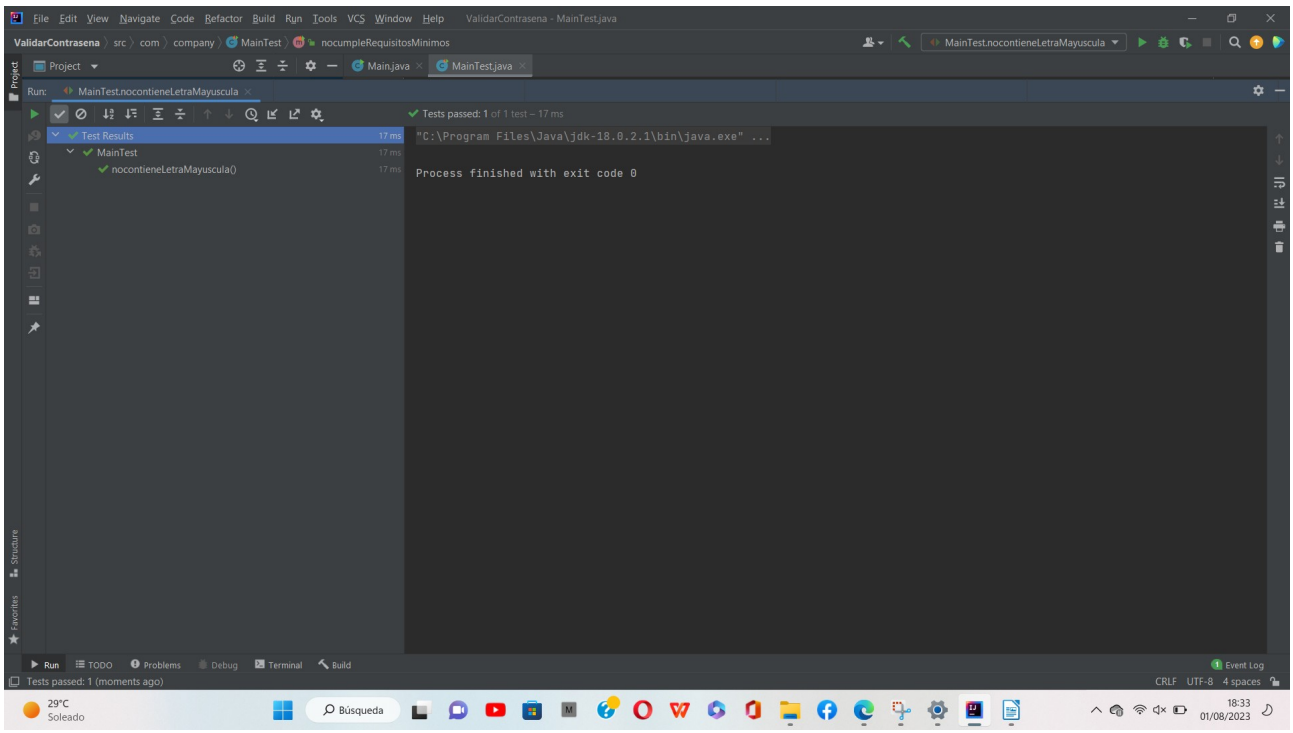
La función actual siempre devuelve true,sin importar si la contraseña contiene letras mayúsculas o no. Esto no cumple con el objetivo de validar si la contraseña contiene al menos una letra mayúscula.

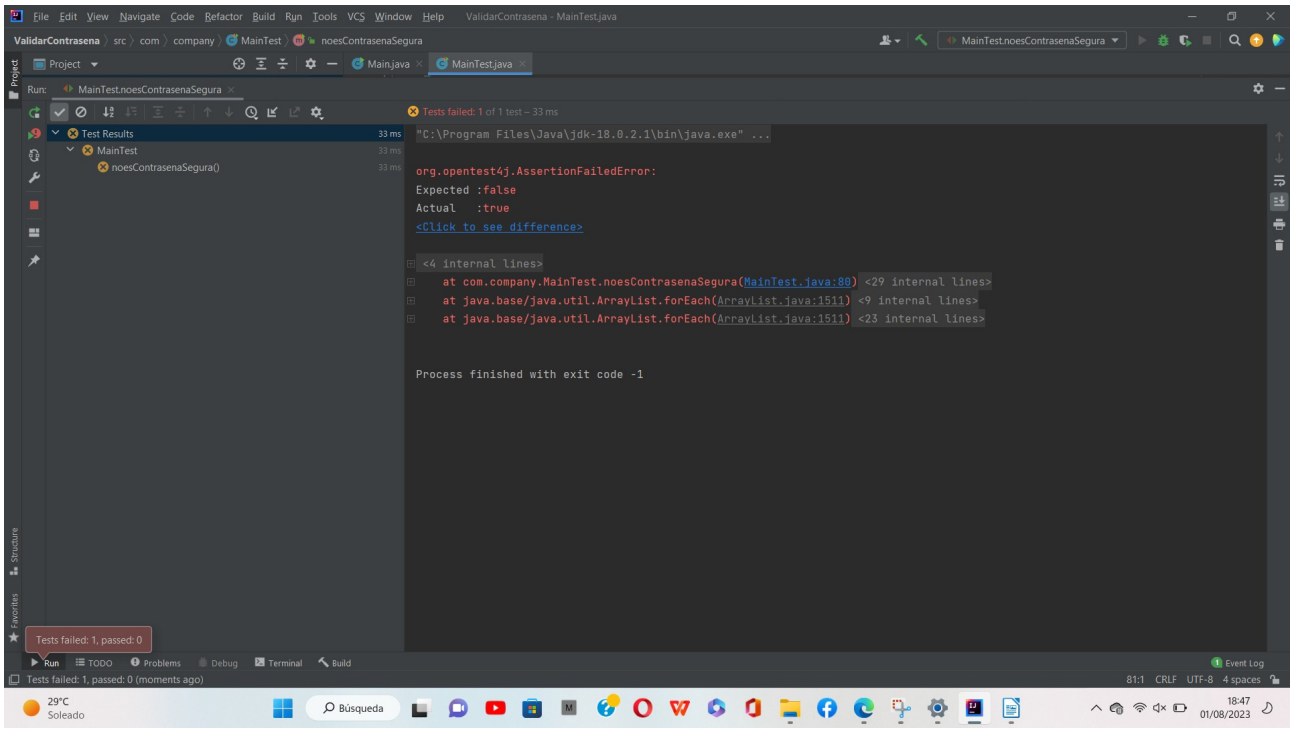
Corrección :

Para corregir el error y validar correctamente si la contraseña contiene al menos una letra mayúscula, necesitamos verificar si hay alguna letra mayúscula en la cadena contrasena. Podemos hacerlo utilizando expresiones regulares o simplemente recorriendo los caracteres de la contraseña.

```
12
13      // Función para validar si la contraseña contiene al menos una letra mayúscula
14  @   public static boolean contieneLetraMayuscula(String contrasena) {
15          return contrasena.matches( regex: ".*[A-Z].*"); // Corrección: Devuelve true si la contraseña contiene al menos una letra mayús
16      }
17
```







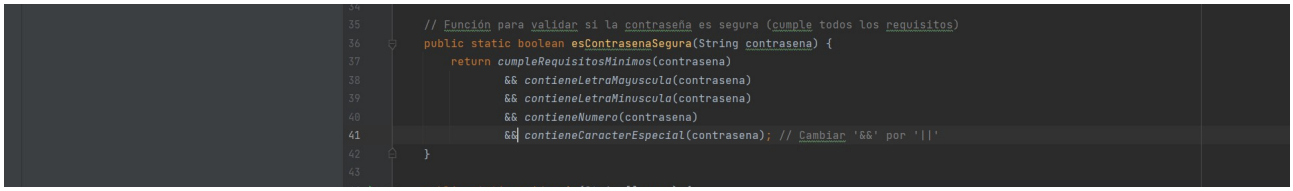
esContrasenaSegura() :

Errores :

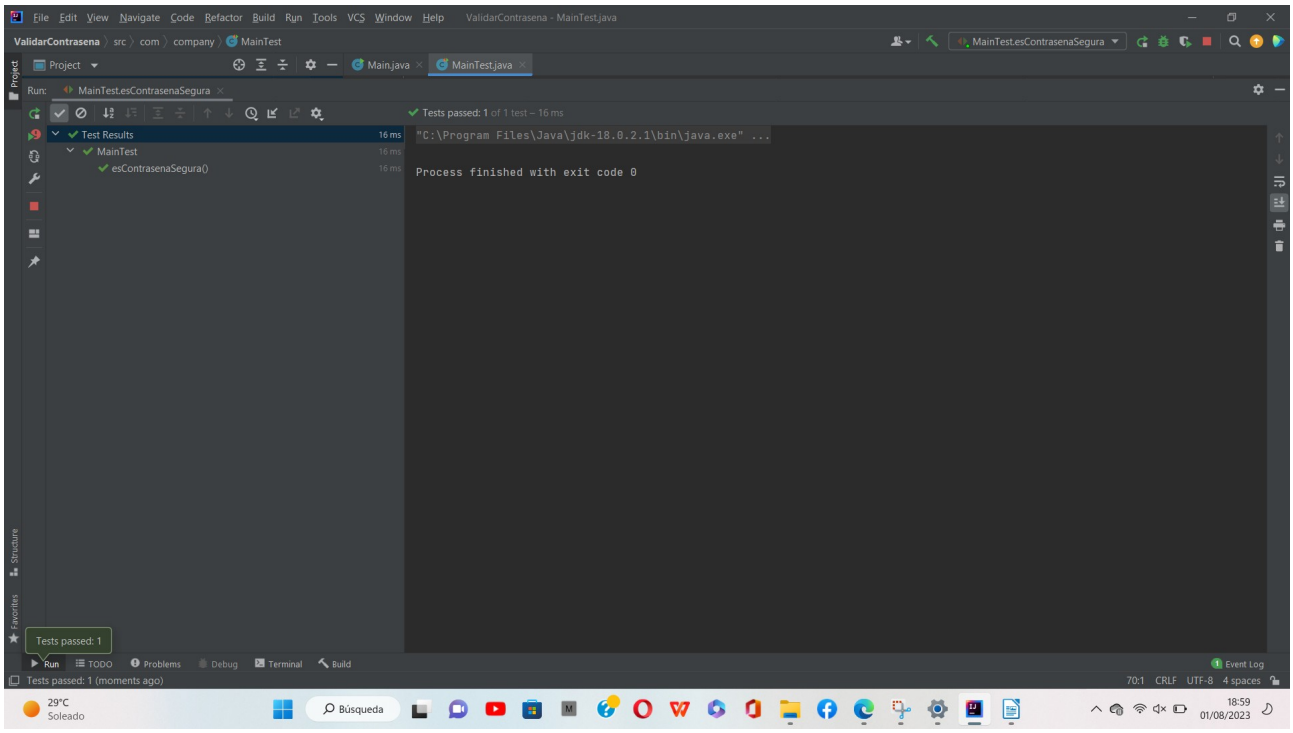
La función actual utiliza el operador lógico || (OR) en lugar del operador && (AND) para combinar las validaciones de cada requisito. Esto significa que la función considerará que una contraseña es segura si cumple con al menos uno de los requisitos, lo cual no es el comportamiento correcto para validar que la contraseña cumpla con todos los requisitos al mismo tiempo.

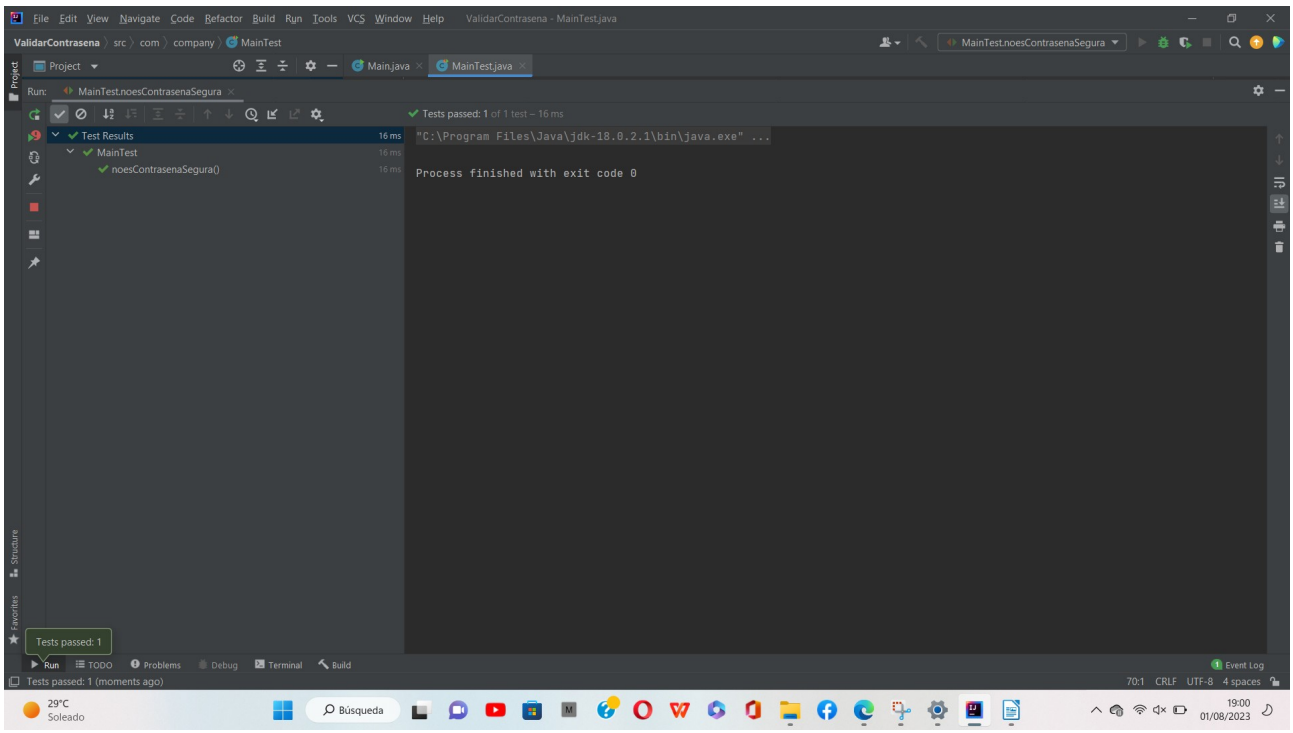
Corrección :

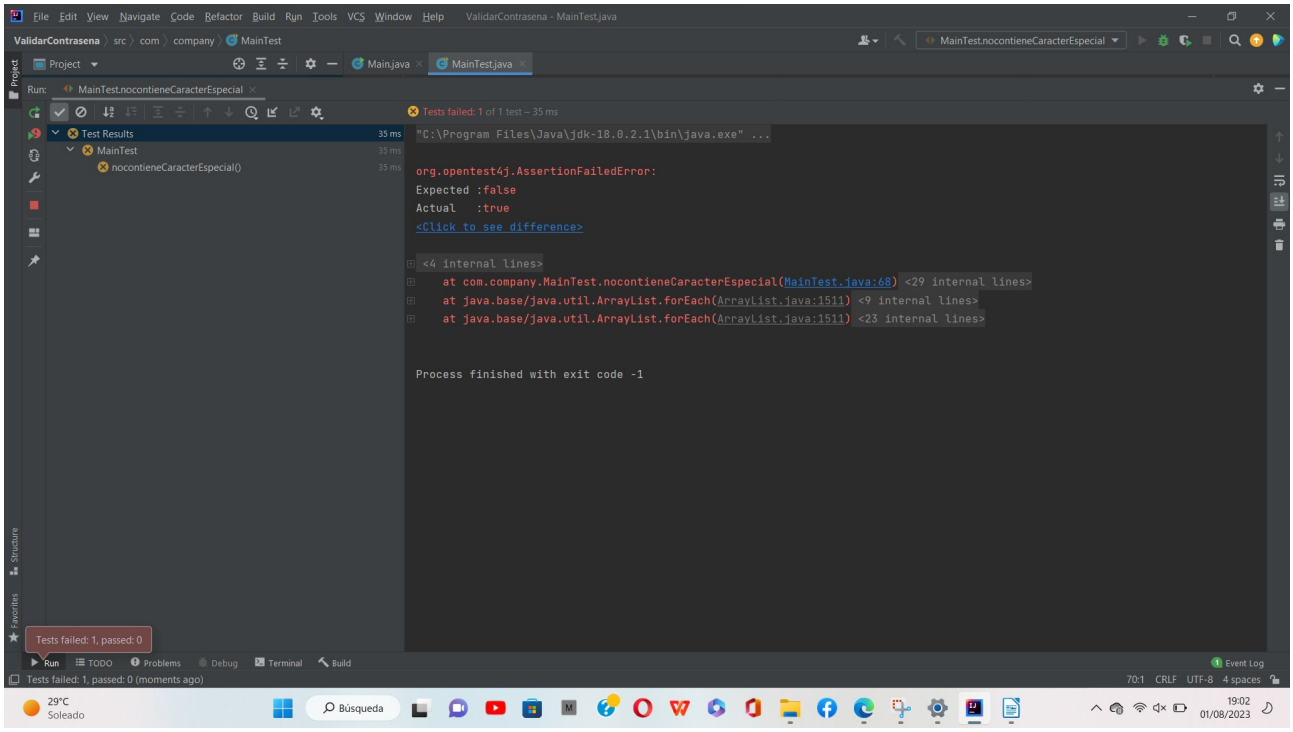
Para corregir el error y validar correctamente que la contraseña cumpla con todos los requisitos al mismo tiempo, necesitamos utilizar el operador && en lugar de | |.



```
// Función para validar si la contraseña es segura (cumple todos los requisitos)
public static boolean esContrasenaSegura(String contrasena) {
    return cumpleRequisitosMinimos(contrasena)
            && contieneLetraMayuscula(contrasena)
            && contieneLetraMinuscula(contrasena)
            && contieneNumero(contrasena)
            && contieneCaracterEspecial(contrasena); // Cambiar '&&' por '||'
}
```







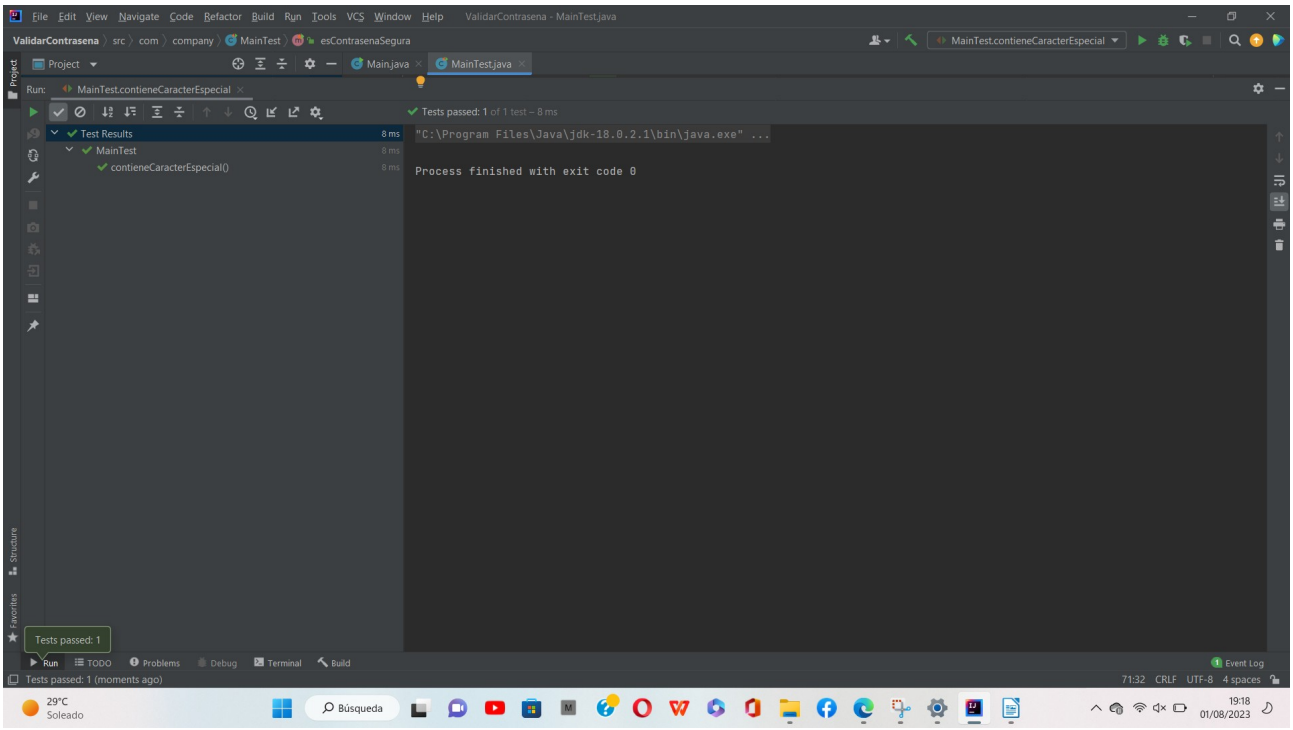
contieneCaracterEspecial() :

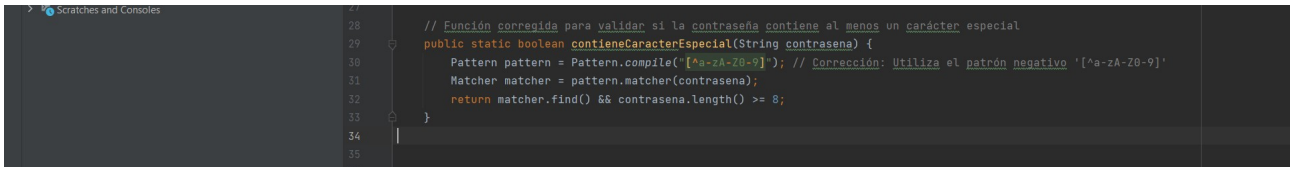
Errores :

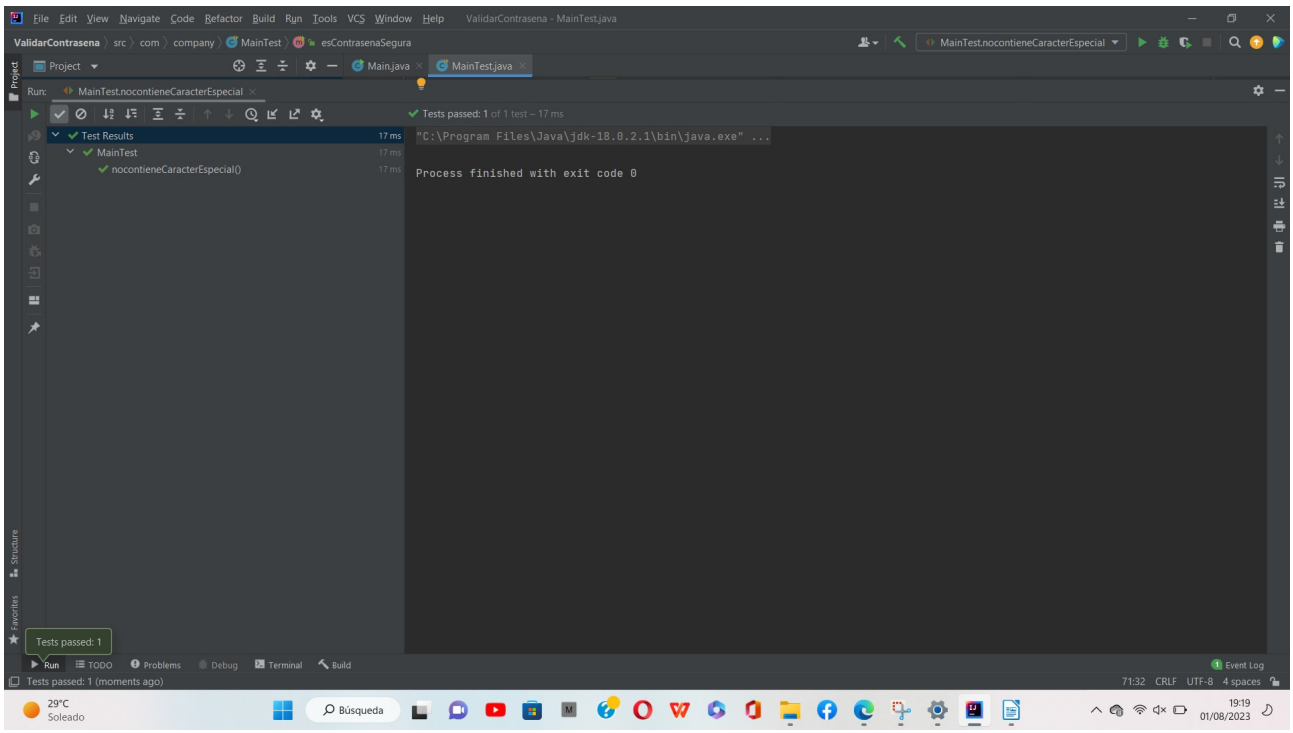
La función actual utiliza el patrón [a-zA-Z0-9] en lugar del patrón negativo [^a-zA-Z0-9].Esto significa que la función devolverá true si encuentra cualquier letra o número en la contraseña, en lugar de buscar un carácter especial.

Corrección :

Para corregir el error y validar correctamente si la contraseña contiene al menos un carácter especial, necesitamos utilizar el patrón negativo [^a-zA-Z0-9] en lugar de [a-zA-Z0-9].






File Edit View Navigate Code Refactor Build Run Tools VCS Window Help ValidarContrasena - MainTest.java
ValidarContrasena src com company MainTest esContrasenaSegura
MainTest.nocontieneCaracterEspecial
Project
Main.java
MainTest.java
Run: MainTest.nocontieneCaracterEspecial
Tests passed: 1 of 1 test – 17 ms
Test Results 17 ms
MainTest 17 ms
nocontieneCaracterEspecial() 17 ms
"C:\Program Files\Java\jdk-18.0.2.1\bin\java.exe" ...
Process finished with exit code 0
Tests passed: 1
Run TODO Problems Debug Terminal Build
Event Log
Tests passed: 1 (moments ago)
71:32 CRLF UTF-8 4 spaces
29°C
Soleado
Búsqueda
19:19
01/08/2023