POLITÉCNICO
ESTELLA

ASIGNATURA: ETORNOS DE DESARROLLO

GRADO SUPERIOR EN DESARROLLO DE APLICACIONES MULTIPLATAFORMA

TAREA EDDE 06

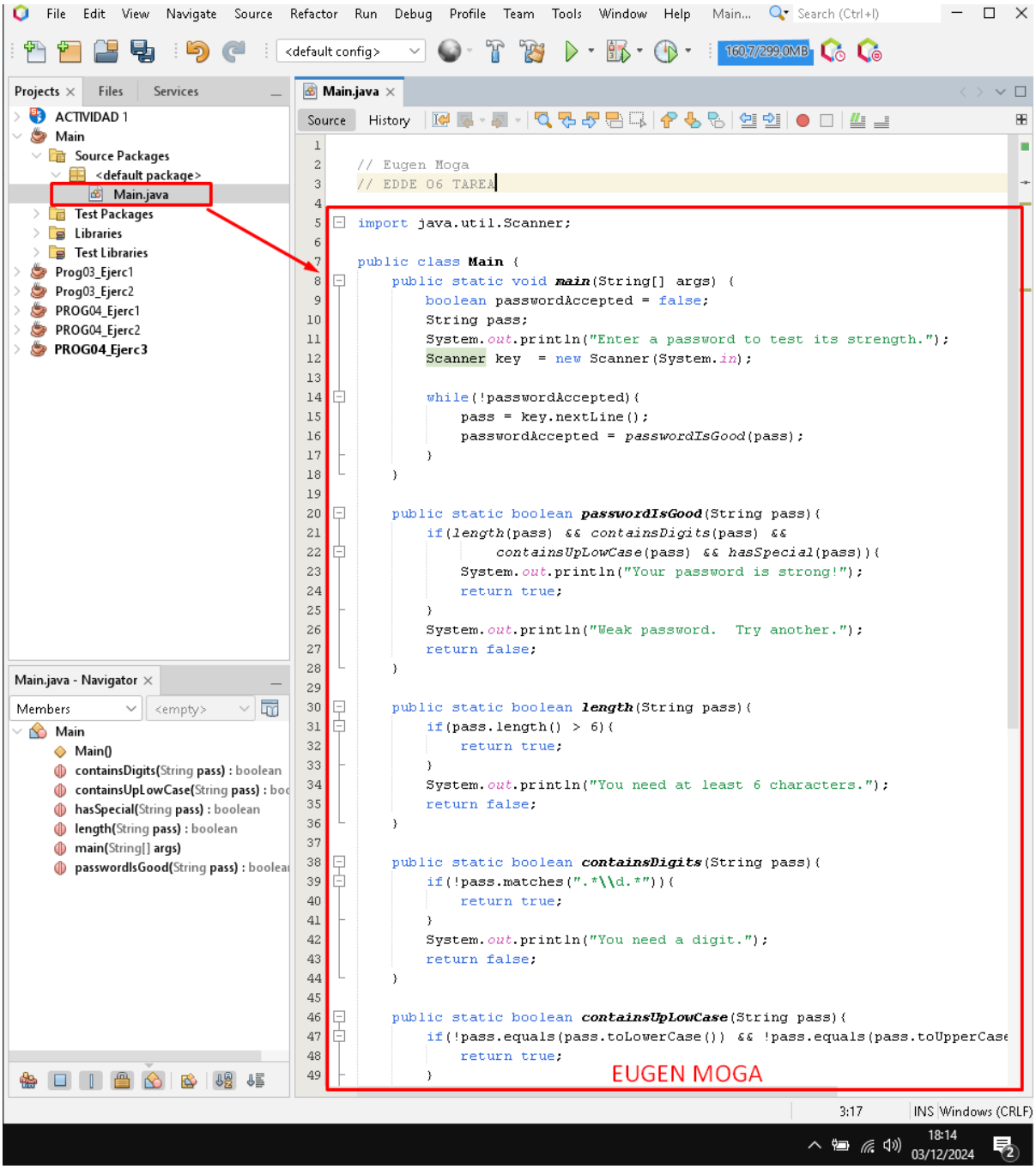
Presentado por: Eugen Moga

Índice

Ejercicios 1

1/ Desde Netbeans, crear el fichero Main.java y copia el código del fichero adjuntado a la tarea.

Creo la clase Main.java y copio el código proporcionado.



Ejercicios 2

2/ Abre el fichero Main.java: - Estudia su código y añade líneas de comentarios que permiten ayudar a la comprensión del código (un comentario para cada método mínimamente).

boolean passwordAccepted = false;

Se inicia la variable passwordAccepted de tipo boleano para comprobar si la contraseña cumple con los criterios de seguridad.

String pass;

Variable pass para almacenar la contraseña ingresada por el usuario.

System.out.println("Enter a password to test its strength.");

Scanner key = new Scanner(System.in);

Solicita al usuario que ingrese una contraseña para probar su fortaleza.

while(!passwordAccepted){

Repite hasta que se ingrese una contraseña válida.

pass = key.nextLine();

Lee la contraseña desde la entrada del usuario.

passwordAccepted = passwordIsGood(pass);

Verifica si la contraseña cumple con los criterios definidos.

public static boolean passwordIsGood(String pass){

if(length(pass) && containsDigits(pass) && containsUpLowCase(pass) && hasSpecial(pass))

Comprueba si la contraseña cumple con todos los criterios: longitud, presencia de dígitos, mayúsculas/minúsculas y caracteres especiales.

return true;

Contraseña válida.

System.out.println("Weak password. Try another.");

return false;

Si no cumple, se informa que es débil y se solicita otra.

public static boolean length(String pass)

if(pass.length() > 6)

return true;

Verifica si la contraseña tiene al menos 6 caracteres.

System.out.println("You need at least 6 characters.");

return false;

Contraseña demasiado corta. Y el resultado es falso

public static boolean containsDigits(String pass)

if(!pass.matches(".*\\d.*"))

return true;

Comprueba si la contraseña contiene al menos un dígito.

System.out.println("You need a digit.");

return false;

Tiene que tener al menos un numero en la contraseña. Y el resultado es falso

public static boolean containsUpLowCase(String pass)

if (!pass.equals(pass.toLowerCase()) &&
!pass.equals(pass.toUpperCase())) {

return true;

Verifica si la contraseña tiene letras mayúsculas y minúsculas. Si las tiene el resultado es verdadero.

System.out.println("You need a lower and upper case letter.");

return false;

// No contiene al menos una mayúscula y una minúscula, y el resultado es falso.

public static boolean hasSpecial(String pass){

if(!pass.matches("[A-Za-z0-9]*")){

return false;

// Comprueba si la contraseña contiene caracteres especiales.

System.out.println("You need a special character.");

return false;

No hay caracteres especiales en la contraseña. Y el resultado es falso.

```
public class Main {
    public static void main(String[] args) {

        /* Se inicia la variable passwordAccepted de tipo boleano
        para comprobar si la contraseña cumple con los criterios de seguridad
        */
        boolean passwordAccepted = false;
        String pass;                                                    // Variable pass para almacenar la contraseña ingresada por el usuario.

        // Solicita al usuario que ingrese una contraseña para probar su fortaleza.
        System.out.println("Enter a password to test its strength.");
        Scanner key  = new Scanner(System.in);

        while(!passwordAccepted){                                       // Repite hasta que se ingrese una contraseña válida.
            pass = key.nextLine();                                      // Lee la contraseña desde la entrada del usuario.
            passwordAccepted = passwordIsGood(pass);                    // Verifica si la contraseña cumple con los criterios definidos.
        }
    }

    public static boolean passwordIsGood(String pass){                  // Comprueba si la contraseña cumple con todos los criterios:
        if(length(pass) && containsDigits(pass) &&                      // longitud, presencia de dígitos, mayúsculas/minúsculas y caracteres especiales.
                containsUpLowCase(pass) && hasSpecial(pass)){
            System.out.println("Your password is strong!");
            return true;                                                // Contraseña válida.
        }
        System.out.println("Weak password.  Try another.");
        return false;                                                   // Si no cumple, se informa que es débil y se solicita otra.
    }

    public static boolean length(String pass){
        if(pass.length() > 6){                                          // Verifica si la contraseña tiene al menos 6 caracteres.
            return true;
        }
        System.out.println("You need at least 6 characters.");          // Te dice que necesita al menos 6 caracteres
        return false;
    }

    public static boolean containsDigits(String pass){
        if(!pass.matches(".*\\d.*")){                                   // Comprueba si la contraseña contiene al menos un dígito.
            return true;
        }
        System.out.println("You need a digit.");                        // Tiene que tener almenos un numero
        return false;
    }
                                                                        EUGEN MOGA
    public static boolean containsUpLowCase(String pass){
```

Ejercicios 3

3/ De manera general, ¿cuáles son los criterios para tener una contraseña segura? ¿El programa parece cumplir con esos requisitos?

Según Microsoft los criterios son:

- Al menos 12 caracteres de longitud
- Una combinación de letras mayúsculas, minúsculas, números y símbolos.
- No utilizar palabras de diccionario, nombre de persona, producto u organización.
- Que sea diferente a contraseñas anteriores.
- Fácil de recordar, pero difícil de adivinar para otros.

Cumple con los requisitos excepto en la longitud mínima que es de 6 caracteres.

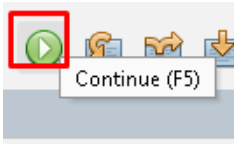
Tampoco detecta patrones predecibles o palabras comunes, que seria crucial para mayor seguridad.

Ejercicios 4

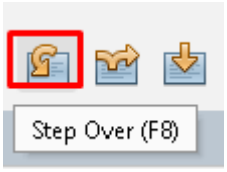
4/ - Pon puntos de ruptura (breakpoint) en cada método de comprobación de la contraseña: líneas 27, 35, 43 y 51 del código inicial.

- ¿Cuáles son las diferencias entre cada tipo de avance de debuggueo? (stepover, stepinto…).

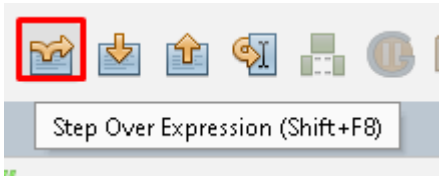
Continue avanza al siguiente breakpoint.



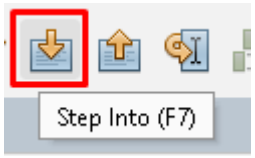
Step Over ejecuta la línea actual y pasa a la siguiente.



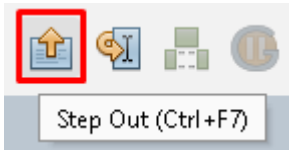
Step Over Expression evalúa subexpresiones dentro de una línea de código antes de pasar a la siguiente instrucción.



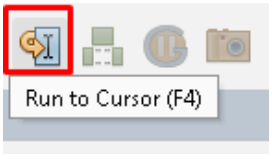
Step Into ingresa en la ejecución de funciones llamadas en la línea actual.



Step Out finaliza la ejecución del método actual y vuelve al nivel superior.



Run to cursor ejecuta el programa hasta llegar a la línea en la que colocaste el cursor



- Prueba el comportamiento del programa entrando contraseñas de diferentes tipos hasta recoger cada caso, es decir hasta llegar hasta cada breakpoint puesto.

No entiendo la lógica de este punto del ejercicio. ¿solo hay que ejecutar el programa y mostrar cada breakpoint y ver que funciona correctamente?

Pero hay 2 errores en el código, asi que no va a funcionar correctamente. Primero arreglaría los errores del código y luego mostraría cómo funciona cada método.

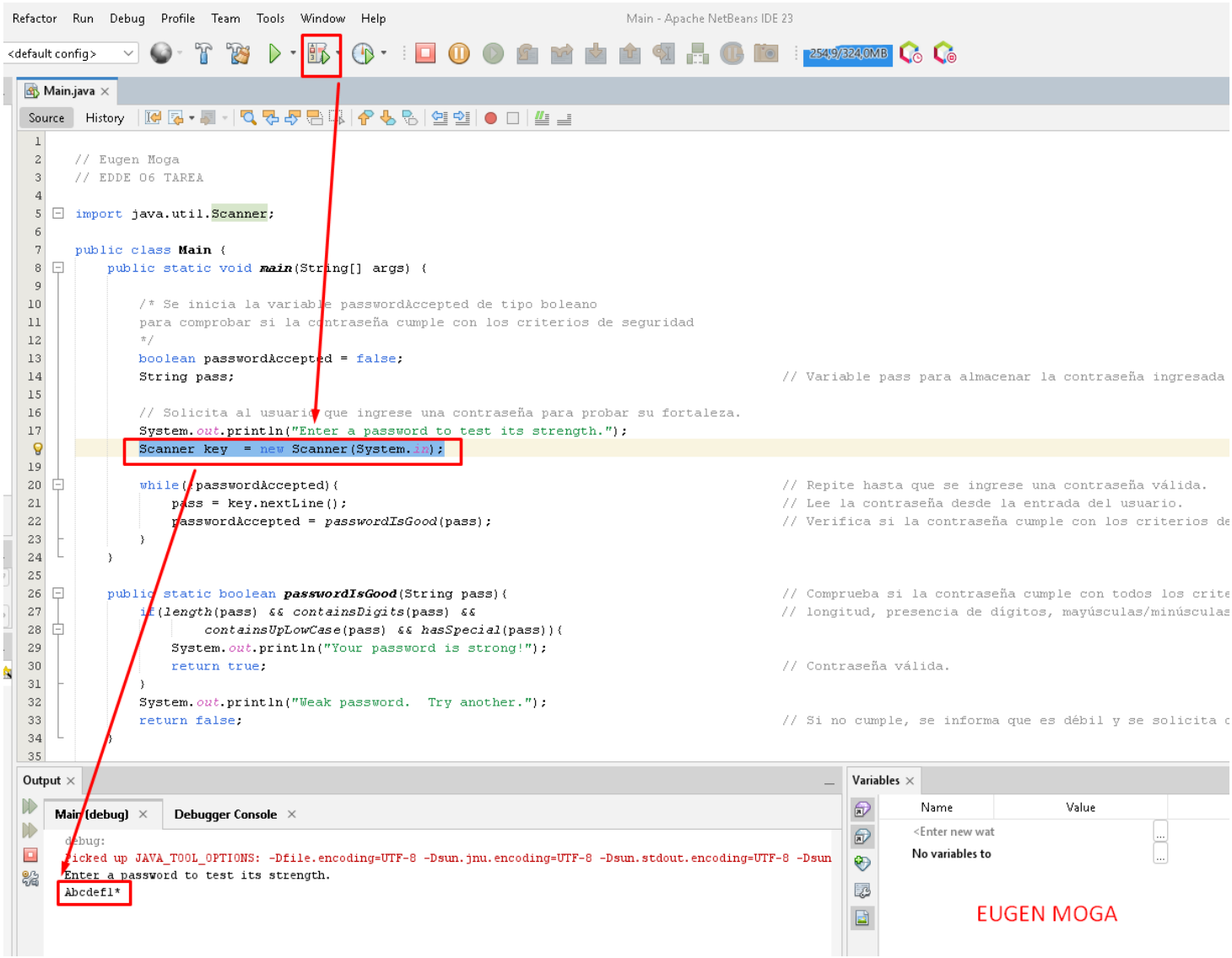
Voy a intentar explicarme lo mejor posible.

Ejecuto el programa desde el botón Debug Main Project y se para en la línea

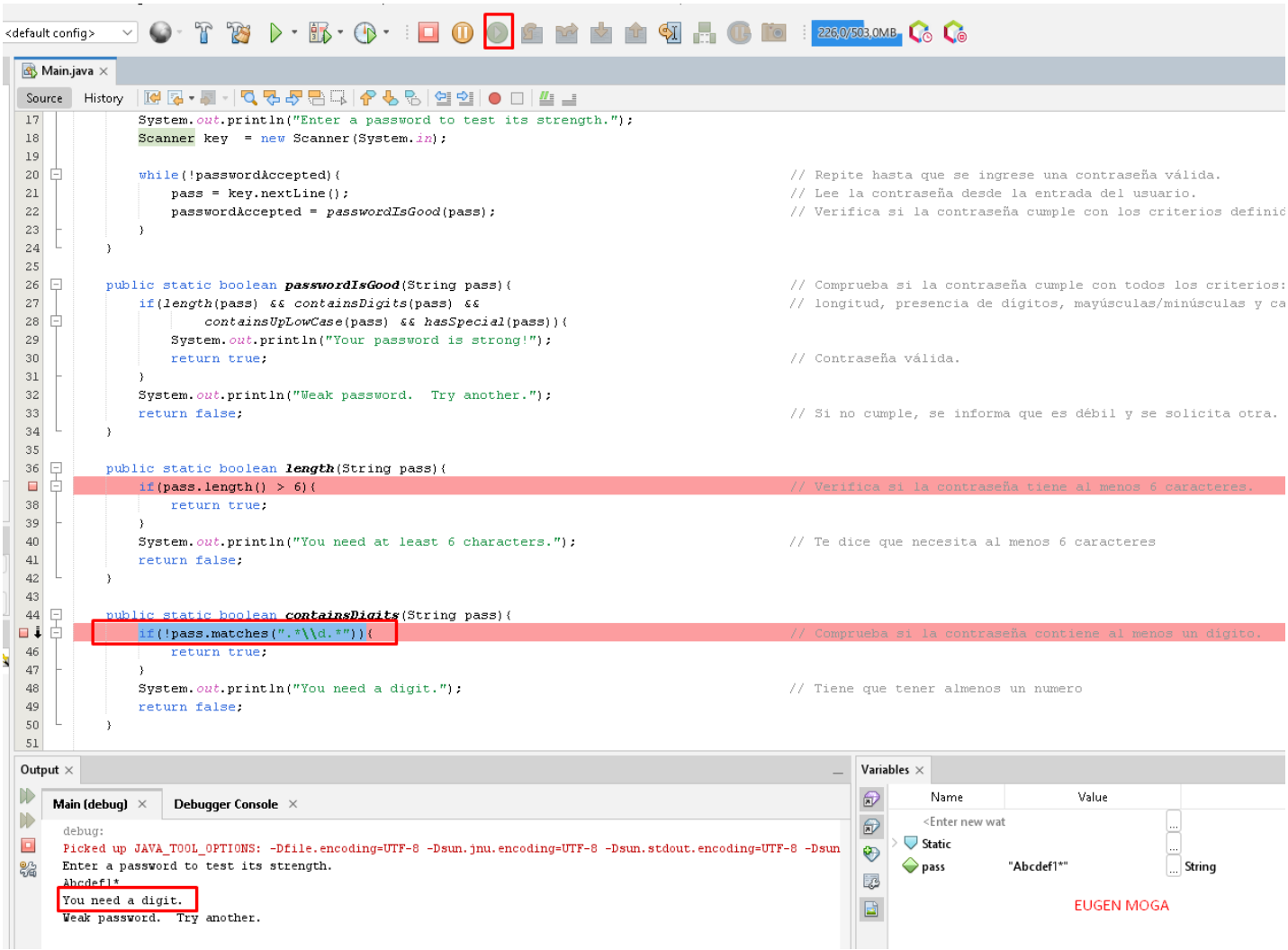
Scanner key = new Scanner(System.in);

Esperando la entrada desde teclado

Le pongo la contraseña Abcdef1* y le doy enter.

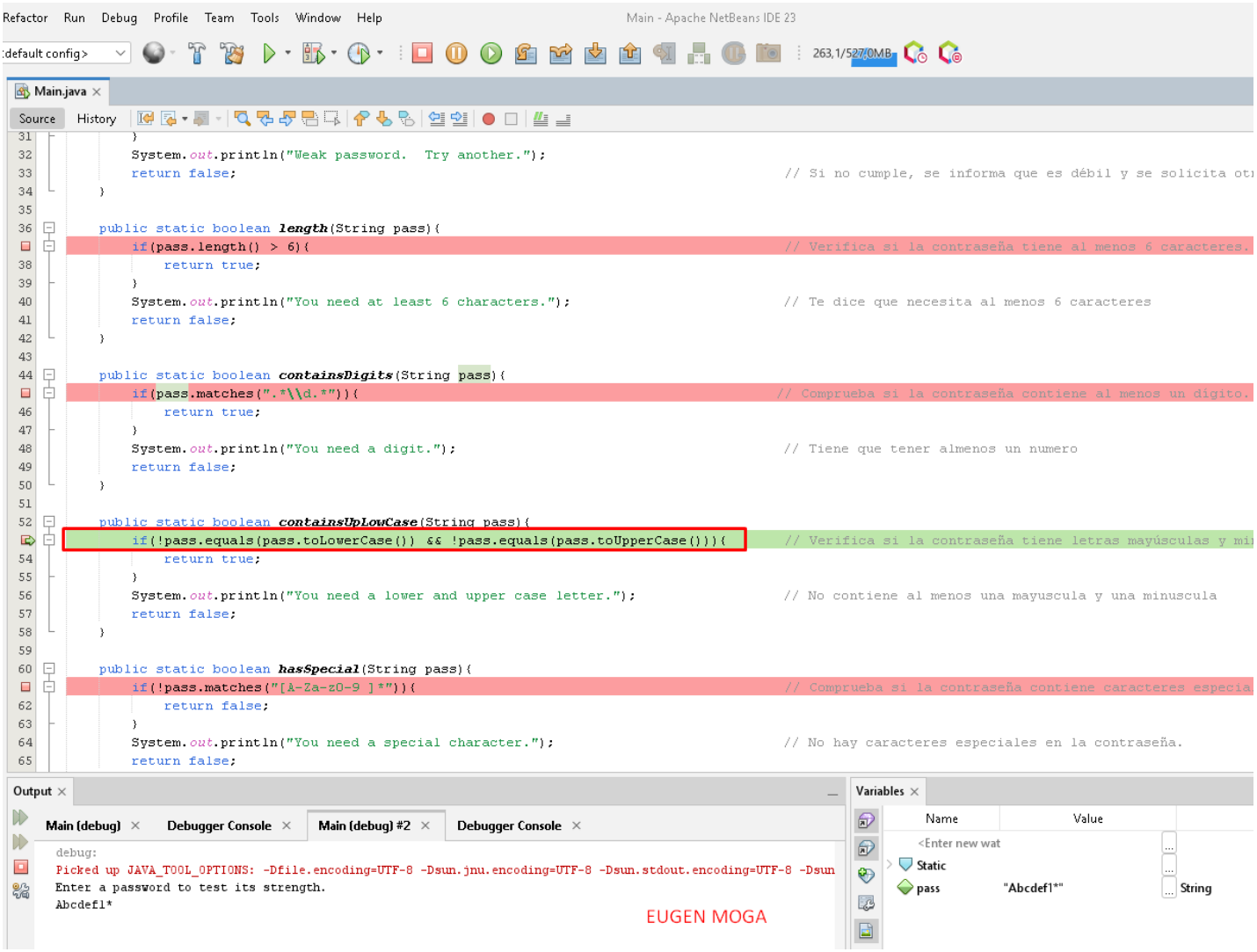


Ahora esta parado en el primer breakpoint en la línea if(pass.length() > 6) para continuar le doy al botón Continue y pasa al siguiente breakpoint en la linea if(!pass.matches(".*\\d.*")) y aquí se encuentra el primer error. Esta línea debería comprobar si hay al menos un numero pero como tiene el signo ! delante de la variable pass hace exactamente lo contrario. Y arroja el error de que necesita un digito



Soluciono el error dejando la línea de la siguiente manera if(pass.matches(".*\\d.*"))

Una vez solucionado el error pasa al tercer breakpoint y se para en la línea if(!pass.equals(pass.toLowerCase()) && !pass.equals(pass.toUpperCase())) comprobando que tiene al menos una letra minúscula y una mayúscula, como esta condición es verdadero pasa al ultimo breakpoint al darle al botón Continue



Ahora esta detenido en el ultimo breakpoint exactamente en la línea if(!pass.matches("[A-Za-z0-9]*")) que comprueba si la contraseña tiene caracteres especiales. Y aquí esta el segundo error del código ya que una vez que comprueba que el código tiene caracteres especiales devuelve falso en la línea 62 return false;

```
    public static boolean hasSpecial(String pass){
        if(!pass.matches("[A-Za-z0-9 ]*")){                                    // Compr
            return false;
        }
        System.out.println("You need a special character.");                   // No ha
        return false;
    }
}
```

```
lain (debug) ×   Debugger Console ×   Main (debug) #2 ×   Debugger Console ×
debug:
Picked up JAVA_TOOL_OPTIONS: -Dfile.encoding=UTF-8 -Dsun.jnu.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun
Enter a password to test its strength.
Abcdefl*
```

Devuelve falso tanto si se cumple la condición como si no se cumple para solucionarla cambio la línea 62 a return true;

```
Refactor  Run  Debug  Profile  Team  Tools  Window  Help        Main - Apache NetBeans IDE 23
<default config>
Main.java ×
Source  History
34      }
35
36      public static boolean length(String pass){
37          if(pass.length() > 6){                                  // Verifica si la contraseña tiene al menos
38              return true;
39          }
40          System.out.println("You need at least 6 characters.");  // Te dice que necesita al menos 6 caractere
41          return false;
42      }
43
44      public static boolean containsDigits(String pass){
45          if(pass.matches(".*\\d.*")){                            // Comprueba si la contraseña contiene al men
46              return true;
47          }
48          System.out.println("You need a digit.");                // Tiene que tener almenos un numero
49          return false;
50      }
51
52      public static boolean containsUpLowCase(String pass){
53          if(!pass.equals(pass.toLowerCase()) && !pass.equals(pass.toUpperCase())){   // Verifica si la contraseña tiene letras ma
54              return true;
55          }
56          System.out.println("You need a lower and upper case letter.");   // No contiene al menos una mayuscula y una
57          return false;
58      }
59
60      public static boolean hasSpecial(String pass){
61          if(!pass.matches("[A-Za-z0-9 ]*")){                     // Comprueba si la contraseña contiene carac
62              return true;
63          }
64          System.out.println("You need a special character.");    // No hay caracteres especiales en la contra
65          return false;
66      }
67  }
68
Output ×                                                    Variables ×
Main (debug) ×  Debugger Console ×  Main (debug) #2 ×  Debugger Console ×  Main (debug) #3 ×  Debugger C >
debug:                                                      Name  Value
Picked up JAVA_TOOL_OPTIONS: -Dfile.encoding=UTF-8 -Dsun.jnu.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun   <Enter new wat
Enter a password to test its strength.                      No variables to
Abcdefl*
Your password is strong!
BUILD SUCCESSFUL (total time: 33 seconds)
```

EUGEN MOGA

- Encuentra los dos errores en el código, y corrígelos. - Visualiza las variables mientras haces el debug. ¿Cambia alguna de valor durante el proceso?

Los errores los solucione en el apartado anterior para poder llegar hasta cada breakpoint. Que se encontraban en las líneas

45: if(!pass.matches(".*\\d.*")) para solucionarlo le quite el signo ! antes de la variable pass quedando de la siguiente manera if(pass.matches(".*\\d.*"))

62: return false; lo cambie por return true;

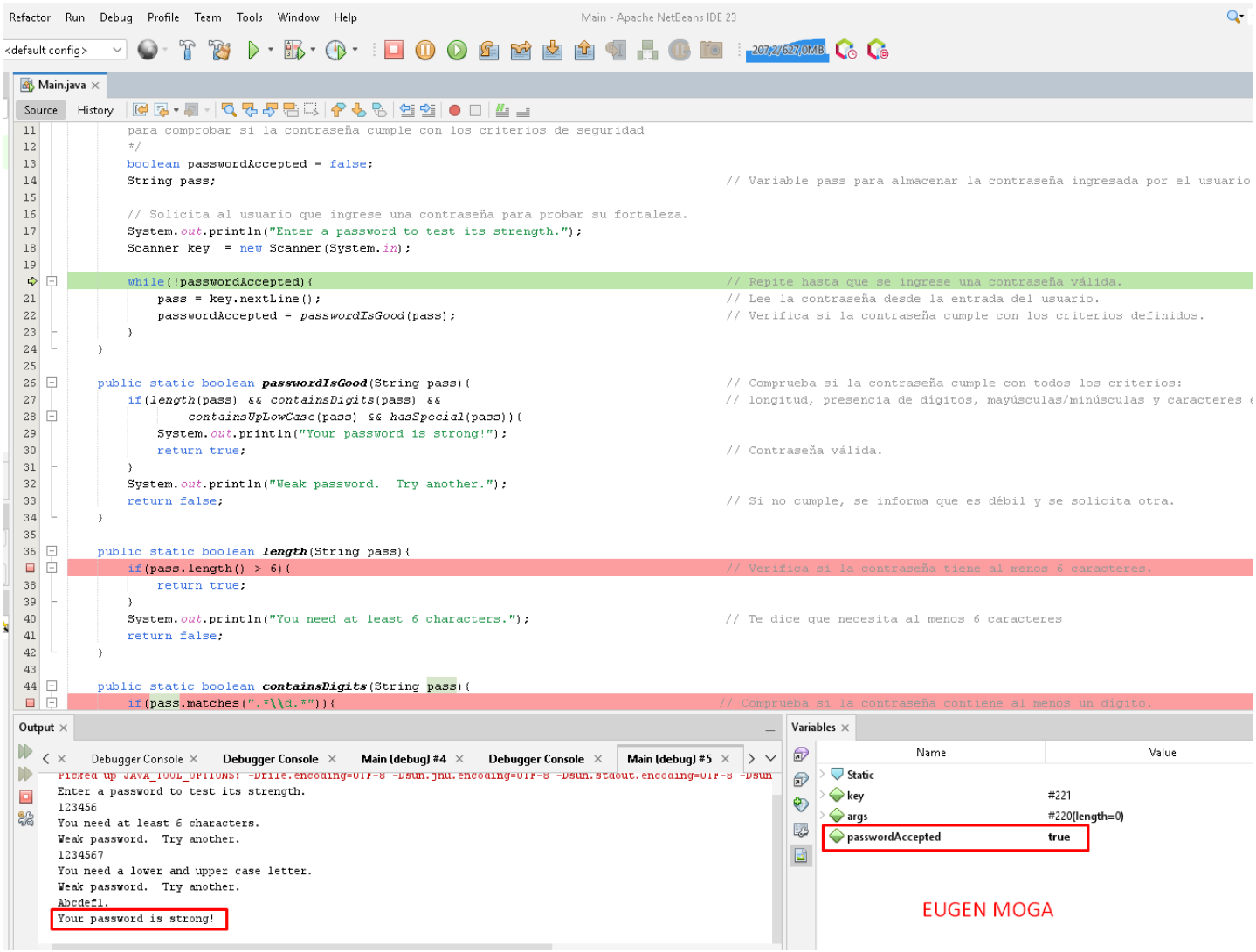
Captura con los errores solucionados

```
35
36      public static boolean length(String pass){
37          if(pass.length() > 6){
38              return true;
39          }
40          System.out.println("You need at least 6 characters.");
41          return false;
42      }
43
44      public static boolean containsDigits(String pass){
45          if(pass.matches(".*\\d.*")){
46              return true;
47          }
48          System.out.println("You need a digit.");
49          return false;
50      }
51
52      public static boolean containsUpLowCase(String pass){
53          if(!pass.equals(pass.toLowerCase()) && !pass.equals(pass.toUpperCase())){
54              return true;
55          }
56          System.out.println("You need a lower and upper case letter.");
57          return false;
58      }
59
60      public static boolean hasSpecial(String pass){
61          if(!pass.matches("[A-Za-z0-9 ]*")){
62              return true;
63          }
64          System.out.println("You need a special character.");
65          return false;
66      }
67  }
68
```

EUGEN MOGA

En cuanto a si las variables cambian durante el proceso de debug la respuesta es sí, pass se actualiza con cada nueva contraseña que se ingresa. Y en el proceso intermedio de cada validación las variables de tipo booleano pueden cambiar de verdadero a falso al hacer las comprobaciones

Por ejemplo, la variable paswordAccepted esta iniciada en false, para que muestre que la contraseña es correcta necesita cambiar a true una vez que se comprueban todas las validaciones.



Ejercicios 5

5/ Crear una clase de test JUnit automáticamente con Netbeans (desde minutos 3:22)

https://www.youtube.com/watch?v=dKW1FFgTa84

¿Con qué texto clave empieza cada test?

Cada test empieza con el texto @test.

Crear un par de test simples para probar el código.

Para crear la clase MainTest en el menú de la izquierda sobre Main.java clic derecho, Tools, Create/Ubdate Tests selecciono JUnit4 y le doy OK

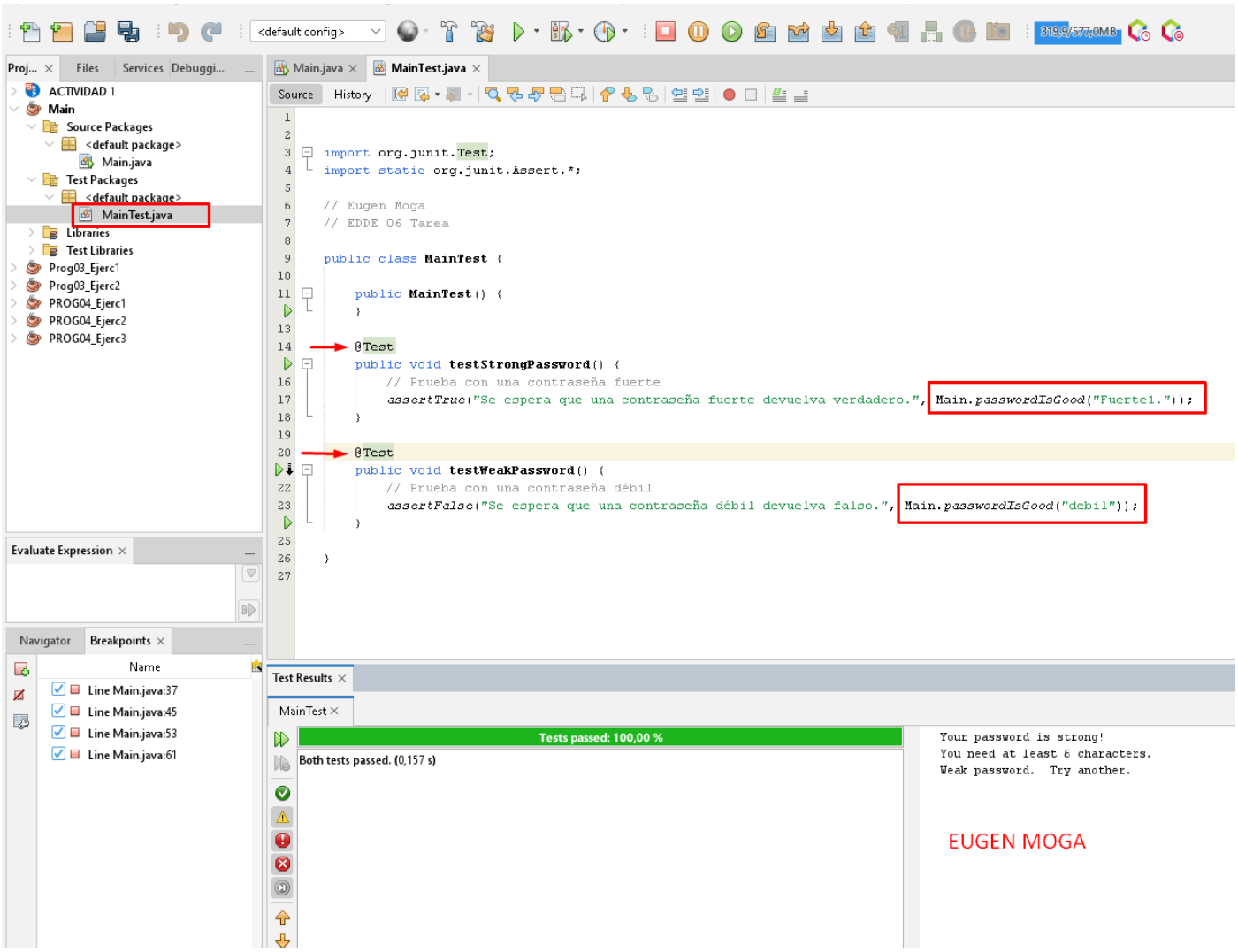
Ahora paso a poner una comprobación simple para verificar el comportamiento del método passwordIsGood de la clase Main.

```
@Test
public void testStrongPassword() {
    assertTrue("Se espera que una contraseña fuerte devuelva verdadero.",
Main.passwordIsGood("Fuerte1."));
```

@Test: indica que este método es un caso de prueba.

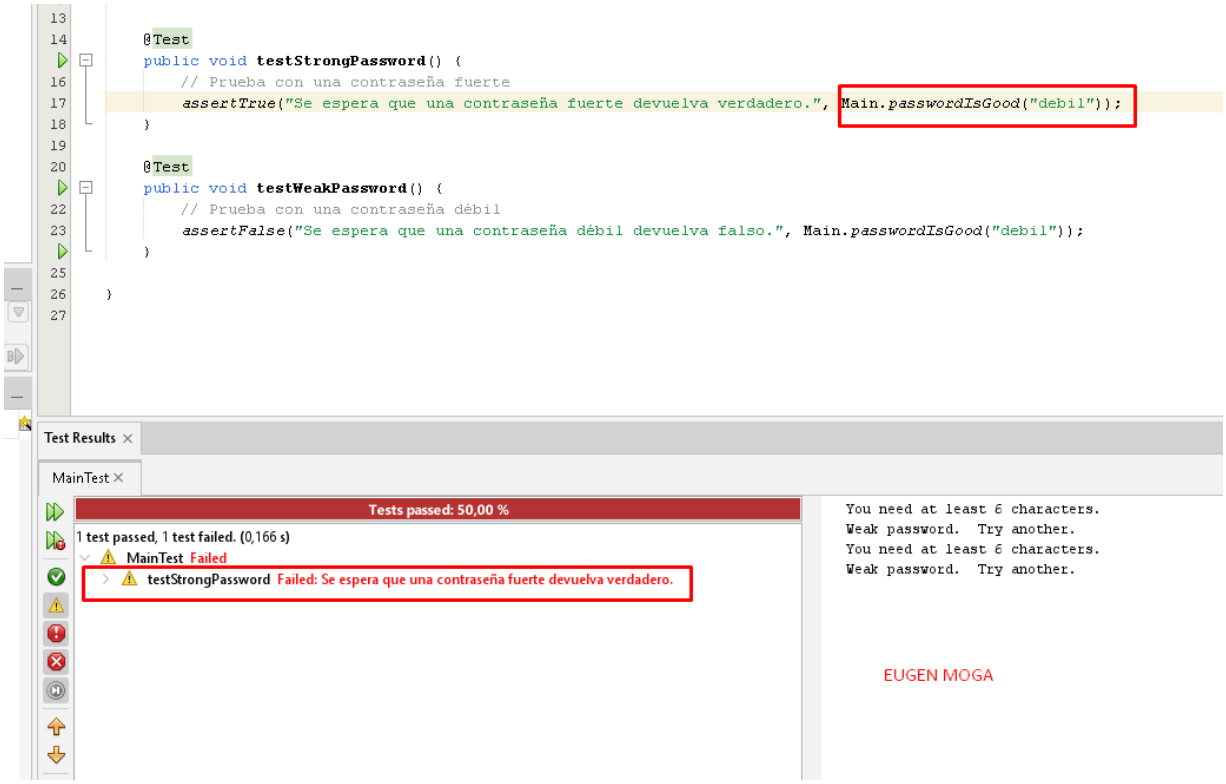
assertTrue: verifica que el resultado de Main.passwordIsGood(“Fuerte1.”) sea verdadero porque cumple todas las restricciones. Si el método no devuelve verdadero (true) el test falla y muestra el mensaje “Se espera que una contraseña fuerte devuelva verdadero”

Creo otro test para comprobar las contraseñas débiles con el nombre testWeakPassword pero esta vez con assertFalse que devuelve verdadero si la contraseña es débil.



Se puede ver que el test ha funcionado correctamente

Pero si por ejemplo le cambio el valor de la variable a débil el test muestra un error y enseña el mensaje



PD: la tarea es muy confusa, espero que se entienda todo bien,