

Pregunta 1 (2,5 puntos): Resolver el cuestionario de la plataforma.

Pregunta 2 (7,5 puntos): Para la realización de ésta y las siguientes preguntas, debes clonar (mediante un *fork* previo) el siguiente repositorio:

<https://github.com/jrgs/DAW-ExamenED-2EV-2324>

Una vez clonado, se debe cambiar el nombre de la clase *Form1* a *Examen2EV* (vuestras iniciales)2324.

También habrá que modificar el nombre de la clase *comprobadorDePassword* para que incluya vuestras iniciales y el curso actual (2324).

El proyecto que acabáis de clonar tiene dos clases: una clase *comprobadorDePassword*, que verifica la fortaleza de una contraseña, y un formulario (*Form1*), que permite al usuario realizar todas las comprobaciones que ofrece la clase.

Los pasos a realizar en el examen serán los siguientes:

1. **(DOSSIER - 1 pt)** Encontrar cinco errores de normas de estilo en el fichero *comprobadorDePassword.cs*, indicando número de línea, error encontrado y solución. NO SE PUEDE REPETIR ERROR, DEBEN SER DIFERENTES.

	Línea	Error	Solución
1	14 16 17 18 26 28 40 41 42 76	Nombres poco descriptivos <pre> 14 public string pwd; 15 16 private bool mins; 17 private bool mays; 18 private bool nums; 26 26 ✓ public int test(string p) 41 42 int f=0; 76 </pre>	Poner nombres autoexplicativos: Pwd, p – password mins – lowerCase mays – upperCase nums – numbers test – TestPassword f – force <i>(Refactorizar -> Cambiar nombre)</i>
2	14 - 19	Estilo de los nombres de campos no sigue las normas	Añadir _ delante de cada nombre
3	30 73	Faltan espacios alrededor de los operadores <pre> 30 if (pwd==null pwd.Length<=0) 73 int f=0; </pre>	Añadir dichos espacios <i>(Editar -> Avanzados -> Dar formato al documento)</i>
4	12	Estilo del nombre de clase no sigue las normas <pre> 12 ✓ public class comprobadorDePassword </pre>	Reescribir en PasCal
5	26	Estilo del nombre de método no sigue las normas <pre> 26 ✓ public int test(string p) </pre>	Reescribir en PasCal
	31 34 77 - 80	Faltan llaves	Poner llaves – cada una en línea reservada

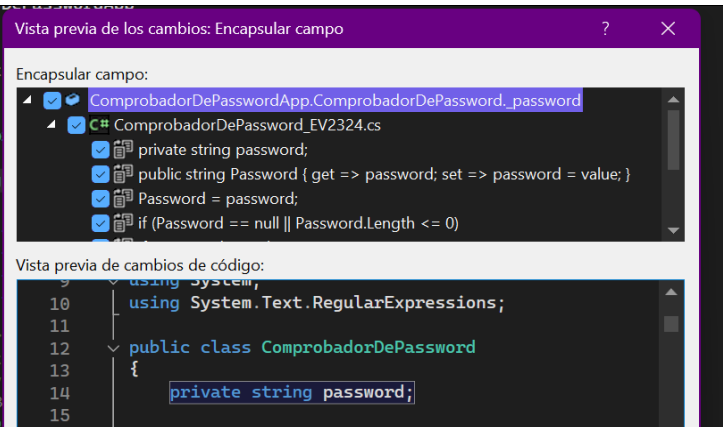
		<pre> 30 if (pwd==null pwd.Length<=0) 31 return -1; // Si la contrase 32 33 if (pwd.Length < 6) 34 return 0; // No tiene la lon </pre>	
--	--	---	--

2. **(DOSSIER - 1 pt)** Realizar el diseño de pruebas (caja negra) para el método *comprobadorDePassword.test()*.

Prueba	Clase de equivalencia	Casos de prueba	Salida
Ok	_password.Lendth >= 6	A2c4e6 C0ntr@s C0ntr@s3ñ@S3gur@	Ok (> 0)
Error1	_password == null	null	Error (-1)
	_password == ""	""	Error (-1)
Error2	_password.Length < 6	A abc A2c4e	Error (0)

3. **(COMMIT - 1,5 pt)** Crear los métodos de prueba que correspondan a los siguientes valores de prueba: "", "abc", "C0ntr@s3ñ@S3gur@"

4. **(DOSSIER+COMMIT - 1,5 pt)** Si existen, detectar y aplicar al menos tres patrones de refactorización DISTINTOS en el fichero *comprobadorDePassword.cs*, indicando el patrón que se aplica y, si es posible aplicarlo con Visual Studio, la opción que se usa.

	Patrones de refactorización	Solución
1	Campo público <pre> 12 public class ComprobadorDePassword 13 { 14 public string _password; </pre>	 <p>Editar → Refactorizar → Encapsular campo</p>
2	Método TestPassword demasiado largo.	Colocar la parte de valoración dentro de setter
3	Ausencia de excepciones	Y agregar casos de excepción

		<pre> public string Password { get => _password; set { _password = value; if (value == null value.Length <= 0) { throw new Exception(ERROR_EMPTY_PASSWORD); } else if (value.Length < MIN_LENGTH) { throw new Exception(ERROR_SHORT_PASSWORD); } } } </pre>
4	Números mágicos	<p>Crear constantes</p> <pre> private const int MIN_LENGTH = 6; private const int SAFE_LENGTH = 12; </pre>
5	Campos innecesarios	<pre> private bool _lowerCase; private bool _upperCase; private bool _numbers; private bool _length; public ComprobadorDePassword() { _lowerCase = _upperCase = _numbers = _length = false; } </pre>
6	Corregir código en la clase Form para que espere las excepciones	
		<pre> try { int resultado = miComprobador.TestPassword(txtPassword.Text); if (resultado == 1) MessageBox.Show("Contraseña débil"); else if (resultado == 2) MessageBox.Show("Contraseña normal"); else if (resultado == 3) MessageBox.Show("Contraseña fuerte"); else if (resultado == 4) MessageBox.Show("Contraseña muy fuerte"); } catch (Exception error) { if (error.Message.Contains(ComprobadorDePassword.ERROR_EMPTY_PASSWORD)) { MessageBox.Show(ComprobadorDePassword.ERROR_EMPTY_PASSWORD); } else if (error.Message.Contains(ComprobadorDePassword.ERROR_SHORT_PASSWORD)) { MessageBox.Show(ComprobadorDePassword.ERROR_SHORT_PASSWORD); } } </pre>

5. **(DOSSIER + COMMIT - 1,5 pt)** En base a los cambios realizados en el punto (4), modificar los métodos de prueba creados en el punto (3).

Prueba	Antes / Después
Ok	Sin cambios

		<pre> [TestMethod] [DataRow("A2c4e6 ")] [DataRow("C0ntr@s ")] [DataRow("C0ntr@s3ñ@S3gur@")] 0 referencias public void TestMethodOk(string password) { ComprobadorDePassword ComprobadorOk = new ComprobadorDePassword(); Assert.IsTrue(ComprobadorOk.TestPassword(password) > 0); } </pre>
Error1	<p>Antes</p> <pre> [TestMethod] [DataRow("")] [DataRow(null)] 0 referencias public void TestMethodError1(string password) { int resultExpected = -1; ComprobadorDePassword ComprobadorNull = new ComprobadorDePassword(); Assert.AreEqual(resultExpected, ComprobadorNull.TestPassword(password)); } </pre> <p>Después</p> <pre> [TestMethod] [DataRow("")] [DataRow(null)] 0 referencias public void TestMethodError1(string password) { ComprobadorDePassword ComprobadorNull = new ComprobadorDePassword(); try { ComprobadorNull.TestPassword(password); } catch (Exception error) { StringAssert.Contains(error.Message, ComprobadorDePassword.ERROR_EMPTY_PASSWORD); return; } Assert.Fail("Error. Se debía haber producido una excepción."); } </pre>	
Error2	<p>Antes</p> <pre> [TestMethod] [DataRow("A")] [DataRow("abc")] [DataRow("A2c4e")] public void TestMethod3(string password) { int resultExpected = 0; ComprobadorDePassword ComprobadorShort = new ComprobadorDePassword(); Assert.AreEqual(resultExpected, ComprobadorShort.TestPassword(password)); } </pre> <p>Después</p>	

```

[TestMethod]
[DataRow("A")]
[DataRow("abc")]
[DataRow("A2c4e")]
...
| 0 referencias
public void TestMethodError2(string password)
{
    ComprobadorDePassword comprobadorShort = new ComprobadorDePassword();

    try
    {
        comprobadorShort.TestPassword(password);
    }
    catch (Exception error)
    {
        StringAssert.Contains(error.Message, ComprobadorDePassword.ERROR_SHORT_PASSWORD);
        return;
    }

    Assert.Fail("Error. Se debía haber producido una excepción.");
}

```

6. **(COMMIT - 1 pt)** Documentar el fichero *comprobadorDePassword.cs*. Sólo se debe documentar los constructores y los métodos públicos.

Después de cada paso indicado como COMMIT se deberá de realizar un *commit* (sólo de manera local). Una vez terminado, deberéis de subir vuestro proyecto a GitHub y enviar la *PullRequest* correspondiente.

Como en el examen anterior y las prácticas, se deberá incluir un dossier explicativo para los puntos (1), (3) y (4). Os recomiendo guardar el dossier en el mismo directorio del proyecto para que automáticamente se haga también *commit* del mismo.