| EXAMEN UD4 - ED | 16/02/2022 |
|---|---|

| CICLO: DESARROLLO DE APLICACIONES WEB CURSO: 1º<br>MÓDULO: ENTORNOS DE DESARROLLO | CALIFICACIÓN: |
|---|---|
| **Nombre:** | **Apellidos:** |

Lee detenidamente y de forma completa cada uno de los problemas planteados y responde únicamente a lo que se pide.

Crear un directorio en el directorio c:/Documentos llamado: NombreApellido_ExUD4-ED.

Una vez completada la prueba, renombrar este documento con el siguiente nombre: 'NombreApellido_Ex_UD4.odt'. Adjuntar a la entrega de la plataforma Moodle:

- 'NombreApellido_Ex_UD4.odt'
- 'NombreApellido_ExUD4-ED.zip' que incluye los directorios de trabajo de cada ejercicio.

Crear un repositorio en el usuario de GitHub. Dicho repositorio incluye un directorio para cada ejercicio del examen.

Añadir a continuación la URL de acceso al repositorio (0,5 puntos):

# Enlace a ejercicios

**Ejercicio 1 (4,5 puntos)**

Requisitos previos: SDK de .NET Core 2.1. Si aún no lo tienes instalado: descargar o utilizar el archivo 'dotnet-sdk-3.1.404-win-x64.exe' facilitado en clase.

1. Crear el proyecto usando comandos.
Abrir la consola de ejecución de comandos de Windows y en el directorio c:/Documentos/NombreAlumno_ExUD4-ED, crear el siguiente directorio: *unit-testing-using-nunit*

En este directorio, abrir un terminal Windows.

```
C:\Users\Ana González\Documents\AnaGlez_ExUD4-ED> cd unit-testing-using-nunit
C:\Users\Ana González\Documents\AnaGlez_ExUD4-ED\unit-testing-using-nunit>
```

Dentro de esa carpeta ejecutar el comando: dotnet new sln

**Adjuntar una captura con el resultado de la ejecución del comando (fecha y hora del momento de realización de la prueba, visible en la captura).**

2. Crear un directorio: *PrimeService*. Dentro de este directorio ejecutar el siguiente comando para crear el proyecto de origen: dotnet new classlib

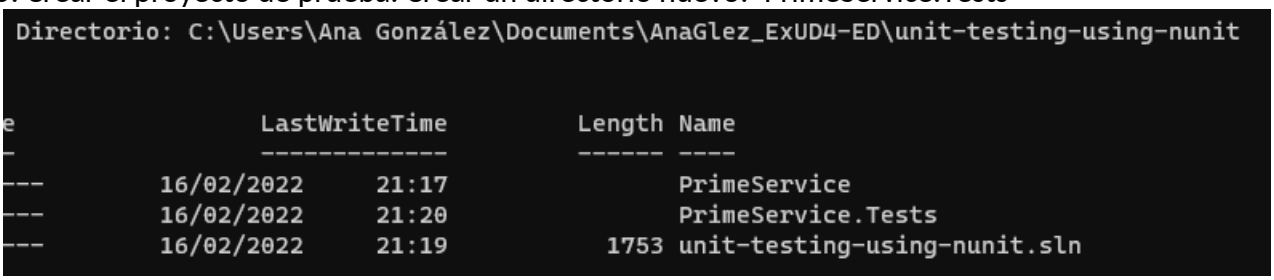
**Adjuntar una captura con el resultado de la ejecución del comando (fecha y hora del momento de realización de la prueba, visible en la captura).**

3. Cambiar el nombre del archivo 'Class1.cs' a 'PrimeService.cs', editar el archivo y añadir el siguiente código:

```
using System;
namespace Prime.Services
{
    public class PrimeService
    {
        public bool IsPrime(int candidate)
        {
            throw new NotImplementedException("Please create a test first.");
        }
    }
}
```

4. Volver a la linea de comandos, situarse en el directorio '*unit-testing-using-nunit*' (comando cd..) y ejecutar el comando:
dotnet sln add PrimeService/PrimeService.csproj

5. Crear el proyecto de prueba. Crear un directorio nuevo: 'PrimeService.Tests'



```
Directorio: C:\Users\Ana González\Documents\AnaGlez_ExUD4-ED\unit-testing-using-nunit


e                LastWriteTime         Length Name
-                -------------         ------ ----
---        16/02/2022     21:17                PrimeService
---        16/02/2022     21:20                PrimeService.Tests
---        16/02/2022     21:19           1753 unit-testing-using-nunit.sln
```

**Adjuntar una captura con la estructura de directorios generada (fecha y hora del momento de realización de la prueba, visible en la captura).**

6. Acceder al directorio '*PrimeService.Tests* ' y ejecutar el siguiente comando:

dotnet new nunit

"dotnet new agrega el SDK de prueba de Microsoft, el marco de pruebas de NUnit y el adaptador de prueba de NUnit."

**Adjuntar una captura con el resultado de la ejecución del comando (fecha y hora del momento de realización de la prueba, visible en la captura).**

*7.* Agregar la biblioteca de clases PrimeService como otra dependencia al proyecto:

dotnet add reference ../PrimeService/PrimeService.csproj

**Adjuntar una captura con el resultado de la ejecución del comando (fecha y hora del momento de realización de la prueba, visible en la captura).**

8. En el directorio unit-testing-using-nunit, ejecutar:
dotnet sln add ./PrimeService.Tests/PrimeService.Tests.csproj

9. En el directorio *PrimeService.Tests*, cambie el nombre del archivo *UnitTest1.cs* por *PrimeService_IsPrimeShould.cs* y reemplace todo su contenido por el código siguiente:

```
using NUnit.Framework;
using Prime.Services;

namespace Prime.UnitTests.Services
{
   public class PrimeService_IsPrimeShould
   {
      private PrimeService _primeService;

      public void SetUp()
      {
         _primeService = new PrimeService();
      }

      public void IsPrime_InputIs1_ReturnFalse()
      {
         var result = _primeService.IsPrime(1);
         Assert.IsFalse(result, "1 should not be prime");
      }

   }
}
```

Añadir al código anterior los atributos NUnit para los siguientes requisitos de la clase prueba:

- El atributo [TestFixture] para indicar que la clase **PrimeService_IsPrimeShould** contiene pruebas unitarias.
- El atributo [Test] que indica que el método **IsPrime_InputIs1_ReturnFalse()** es un método de prueba.
- El atributo [SetUp] que indica que el método **SetUp()** es el código de inicialización.

Una vez actualizada la clase principal, ejecutar: dotnet test
**Adjuntar una captura con el resultado de la ejecución del comando (fecha y hora del momento de realización de la prueba, visible en la captura).**
**Añadir una interpretación del motivo del error recibido**

11. Añadir el directorio *unit-testing-using-nunit* al repositorio Git

**Ejercicio 2 ( 5 puntos)**

1. Crear el proyecto usando Netbeans:

**Categoría**: Java with Maven

**Tipo**: Java Application

**Nombre proyecto**: NomApe_Ej2-ExUD4

**Localización**: c:/Documentos/NombreAlumno_ExUD4-ED

2. Crear una clase java: conversor.java

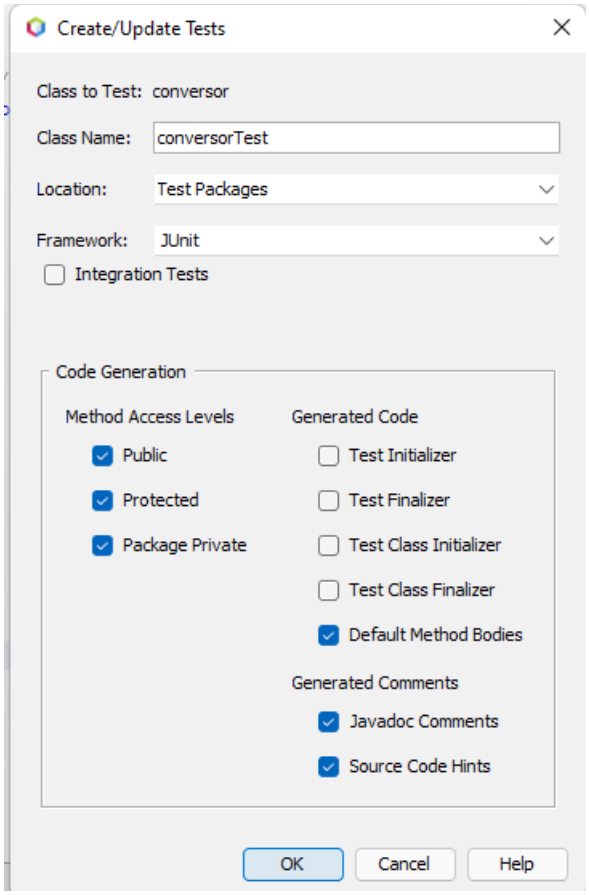
Añadir el siguiente código:

```
/**
 *
 * @author Ana González
 */
public class conversor {
    public static void main(String[] argumentos) {
        float celsius = 0f;
        // Cálculo conversión
        float fahrenheit = celsiusAFahrenheit(celsius);
        System.out.printf("%f grados celsius son %f grados fahrenheit", celsius, fahrenheit);
    }

    public static float celsiusAFahrenheit(float celsius) {
        return (celsius * 1.8f) + 32.0f;
    }

    public static float fahrenheitACelsius(float fahrenheit) {
        return (fahrenheit - 32.0f) / 1.8f;
    }
}
```

3. Crear una prueba test:




Modificar ***conversorTest.java*** con el siguiente código:

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

/**
 *
 * @author Ana González
 */
public class conversorTest {

    public conversorTest() {
    }

    /**
     * Test of main method, of class conversor.
     */
    @Test
    public void testMain() {
        System.out.println("main");
        String[] argumentos = null;
        conversor.main(argumentos);
        // TODO review the generated test code and remove the default call to fail.
        fail("The test case is a prototype.");
    }
```

```
    /**
     * Test of celsiusAFahrenheit method, of class conversor.
     */
    @Test
    public void testCelsiusAFahrenheit() {
        System.out.println("celsiusAFahrenheit");
        float celsius = 0.0f;
        float expResult = 32.0f;
        float result = conversor.celsiusAFahrenheit(celsius);
        assertEquals(expResult, result, 0.0);
        // TODO review the generated test code and remove the default call to fail.
        fail("The test case is a prototype.");
    }

    /**
     * Test of fahrenheitACelsius method, of class conversor.
     */
    @Test
    public void testFahrenheitACelsius() {
        System.out.println("fahrenheitACelsius");
        float fahrenheit = 0.0f;
        float expResult = -17.777779f;
        float result = conversor.fahrenheitACelsius(fahrenheit);
        assertEquals(expResult, result, 0.0);
        // TODO review the generated test code and remove the default call to fail.
        fail("The test case is a prototype.");
    }

}
```

5. Ejecutar las pruebas y adjuntar a continuación una captura del resultado de las pruebas

6. Modificar ***conversorTest.java*** y comentar todas las llamadas al siguiente método:

```
fail("The test case is a prototype.");
```

7. Ejecutar las pruebas y adjuntar a continuación una captura del resultado de las pruebas