LABORATORIO: PRUEBAS DE SOFTWARE

OBJETIVOS

En este laboratorio se probará si la implementación de 2 clases es correcta:

- La clase Esprimo del paquete pruebas nos dice si el número que le pasamos como parámetro es primo o no.
- La clase SiguientePrimo del paquete pruebas nos dice cuál es el siguiente número primo empezando en el que le pasemos como parámetro.

Los objetivos de este laboratorio son:

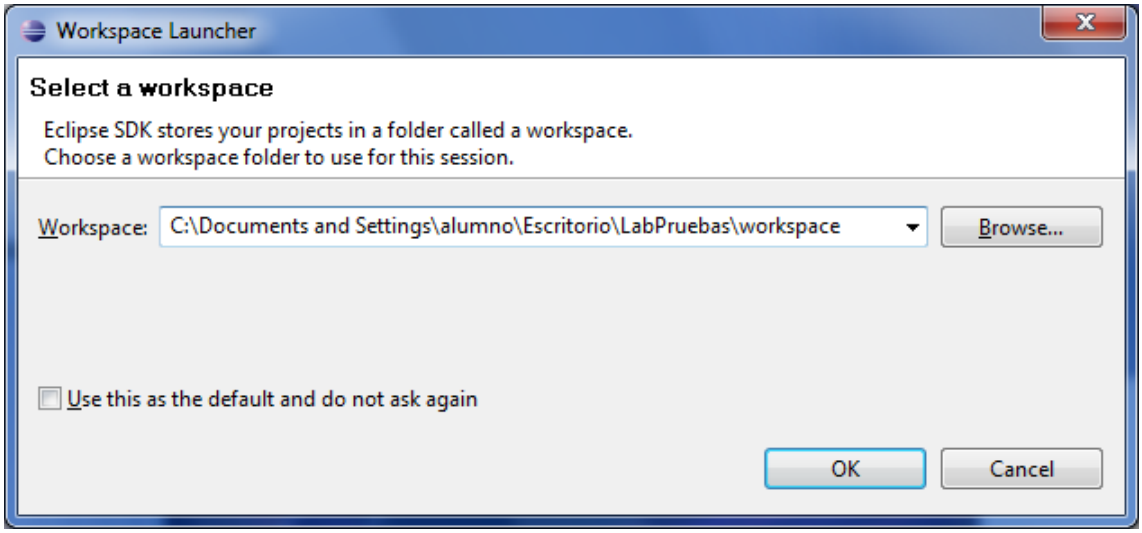
- Mostrar cómo se puede realizar una prueba de unidad construyendo un componente que pruebe de manera automática un conjunto de casos de prueba.
- Mostrar distintas maneras de construir componentes de prueba
    - Escribiendo los casos de prueba en forma de código Java
    - Escribiendo los casos de prueba en tablas de bases de datos
    - Utilizando una herramienta de ayuda como JUnit. En este caso los casos de prueba se escriben en Java; la herramienta proporciona clases para visualizar los resultados de la ejecución de los mismos.
- Mostrar cómo se puede realizar una prueba de integración de varios módulos de manera ascendente (probando primero los módulos hoja) y de manera descendente (construyendo módulos "resguardo").
- Mostrar distintas maneras de construir los resguardos: utilizando una base de datos o usando código Java.

TAREAS A REALIZAR

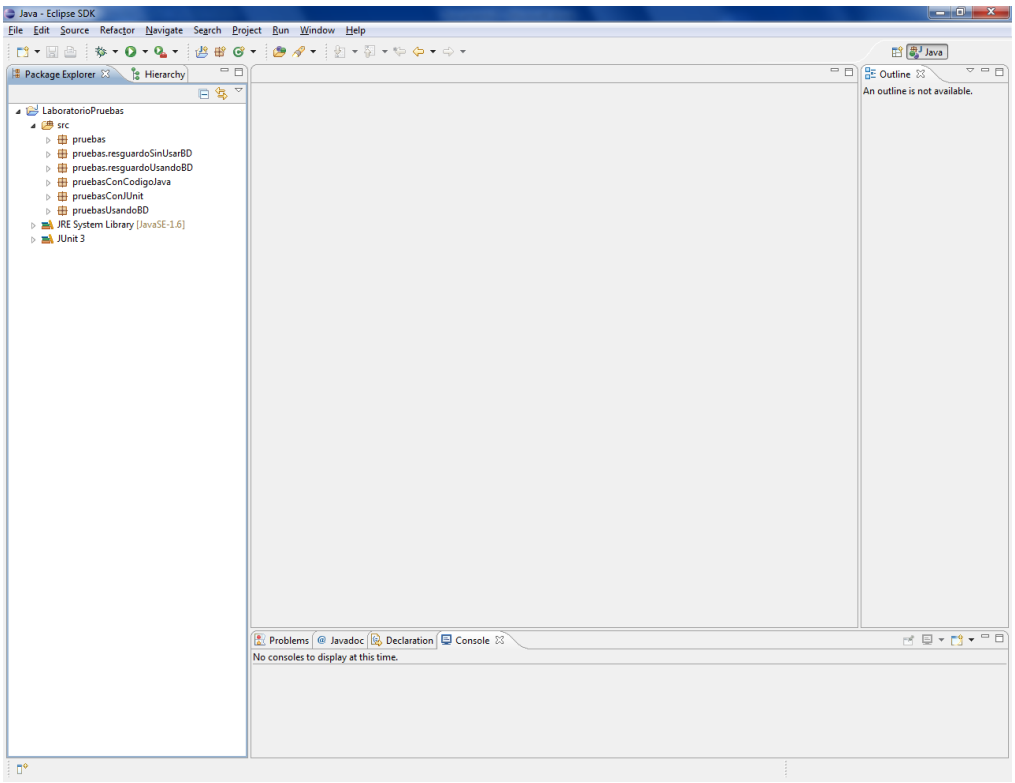
1) **Configurar las aplicaciones que se proporcionan.**

Descargar, descomprimir el fichero Pruebas.zip y configurar la aplicación en Eclipse.

Abrir eclipse y poner como path del workspace, el de la carpeta "workspace" que se encuentra dentro de la carpeta LabPruebas que se ha creado al descomprimir.



Deberá aparecer un proyecto llamado “LaboratorioPruebas”, y dentro de él, en la carpeta src, los distintos paquetes que vamos a usar en el laboratorio.



En el código de conexión a la base de datos, habrá que sustituir el path existente por el path en el que se ha copiado la base de datos al descargarla.

```
Connection conexion=DriverManager.getConnection
("jdbc:ucanaccess://X:/XXXXXXXXXX/Pruebas.mdb");
```

2) Realizar las pruebas de unidad del módulo Esprimo.esPrimo(…)

2.1) Visualizar y Ejecutar el componente de prueba PruebasConCodigoJava.ComponentePruebaEsprimo, que se encuentra en el paquete pruebasConCodigoJava, y que ejecuta una serie de casos de prueba. Comprobar que los casos de prueba se encuentran "codificados" en Java.

// Caso de prueba 1: sin pasar parámetros

```
num = new String[0];
try {result= Esprimo.esPrimo(num);
      System.out.println("Caso de prueba 1 incorrecto");}
      catch (Exception e) {System.out.println("Caso de prueba
1correcto");}
```

Comprobar el resultado de dicha ejecución, viendo los mensajes que aparecen en la salida estándar.

```
Caso de prueba 1 correcto
Caso de prueba 2 correcto
...
```

2.2) Comprobar los casos de prueba, que se encuentran definidos en la tabla PRU_UNIDAD_ESPRIMO de la base de datos proporcionada.

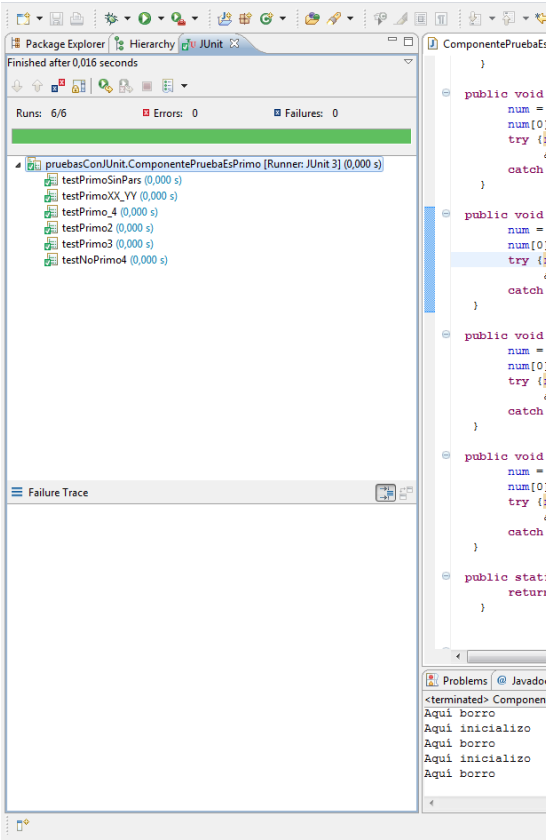
Para borrar los valores de los atributos SalReal y ResPrueba se proporciona una consulta de actualización BorrarPRU_UNIDAD_ESPRIMO

2.3) Ejecutar el componente de prueba PruebasUsandoBD.ComponentePruebaEsprimo, que se encuentra en el paquete pruebasUsandoBD, y que ejecuta los casos de prueba anteriores. Comprobar el resultado de dicha ejecución, visualizando la tabla PRU_UNIDAD_ESPRIMO. Se puede verificar que el módulo Esprimo.esPrimo falla cuando los números son muy grandes.

2.4) Escribir el código Java correspondiente a otros casos de prueba como el 6,7, 8 y 24, los cuales no aparecen en PruebasConCodigoJava.ComponentePruebaEsprimo.

2.5) Ejecutar el componente de prueba PruebasConJUnit.ComponentePruebaEsprimo, que se encuentra en el paquete pruebasConJUnit, y que ejecuta también los casos de prueba en él definidos.

Como resultado de esa ejecución aparece la siguiente interfaz gráfica de usuario, que muestra el resultado de ejecutar todas las pruebas definidas en PruebasConJUnit.ComponentePruebaEsPrimo.



Se puede comprobar que los 6 casos de prueba se han ejecutado de manera satisfactoria.

En la clase PruebasConJUnit.ComponentePruebaEsPrimo, cada caso de prueba se define como un método de la forma testXXX()

```
public void testPrimoSinPars() {
      num = new String[0];
      try {result= Esprimo.esPrimo(num);
            assertTrue(false);}
      catch (Exception e)
      {assertTrue(e instanceof ErrorFaltaParametro);}
}
```

El método *`asserTrue()`* recibe como parámetro una condición o un booleano. Si la condición se cumple, o el booleano es true, la prueba es correcta. En caso contrario, la prueba se considera incorrecta.

Se pueden además definir acciones que se ejecuten antes de probar los casos de prueba (en el método *`setUp`*)

```
protected void setUp() {
      System.out.println("Aquí inicializo");
}
```

y acciones que se ejecuten después de probar los casos de prueba (en el método *`tearDown`*)

```
protected void tearDown() {
      System.out.println("Aquí borro");
```

```
}
```

El método *suite()* devuelve un objeto de una clase Test, con todos los casos de prueba, que están definidos como métodos testXXX en la clase pasada como parámetro a *TestSuite*.

```
public static Test suite() {
      return new TestSuite(ComponentePruebaEsPrimo.class);
}
```

En el método *main()* se pueden ejecutar los casos de prueba extraídos con el método *suite()* usando una interfaz textual.

3) Realizar las pruebas de integración de los módulos Esprimo.esPrimo(…) y SiguientePrimo.siguientePrimo()

3.1) Integración ascendente

Para ello aseguraros de que la clase Pruebas.SiguientePrimo tiene descomentado el import adecuado (aquél que usa el módulo real Esprimo)

```
import Pruebas.Esprimo;
//import Pruebas.resguardoUsandoBD.Esprimo;
// import Pruebas.resguardoSinUsarBD.Esprimo;
```

Y ejecutar los componentes PruebasUsandoBD.ComponentePruebaSiguientePrimo y PruebasConCodigoJava.ComponentePruebaSiguientePrimo.

3.2) Integración descendente, usando un resguardo para el módulo Esprimo.

Para ello aseguraros de que la clase Pruebas.SiguientePrimo tiene descomentado el import de uno de los resguardos y no el módulo real Esprimo.

```
//importPruebas.Esprimo;
import Pruebas.resguardoUsandoBD.Esprimo;
//import Pruebas.resguardoSinUsarBD.Esprimo;
```

3.3) Comprobar cómo se han definido los dos resguardos. Lo que hacen es devolver el valor esperado cuando la entrada coincide con la entrada de alguno de los casos de prueba, y si no, devuelven “no primo”

Pruebas.resguardoUsandoBD.Esprimo lo hace consultando a la base de datos.

Pruebas.resguardoSinUsarBD.Esprimo lo hace usando código Java, como aparece a continuación

```
if (args.length==0)
      throw new ErrorFaltaParametro();
```

```
if (args.length==2 && args[0].equals("xx") && args[1].equals("yy"))
       throw new ErrorSolo1Parametro();
if (args.length==1 && args[0].equals("-4"))
       throw new ErrorNoNumeroPositivo();
if (args.length==1 && args[0].equals("2"))
       return true;
```

4) Crear un nuevo TestCase de JUnit con varios casos de prueba para probar el módulo SiguientePrimo.