CURSO DE JUnit PRUEBAS UNITARIAS

//--------------------------------------------------------//-------------------------------------------------------//

INTRODUCCIÓN:

Introducción:

Es un framework para hacer pruebas del código JAVA.

Instalación del entorno (JDK 8 + Eclipse):

Instalamos estos y verificamos su versión en el terminal de Linux = java -version.

Agregando librerías de JUnit a nuestro proyecto (manual):

Agregar librerías al classPath del proyecto.

* Creamos un proyecto JAVA
* Vamos a JUnit about para ver las librerías que debemos descargar
* Necesitamos junit.jar
* Necesitamos hamcrest-core.jar
* -----------------//----------------------
* Descargamos las librerias
* Vamos el proyecto > librerías > click derecho > BUild Path > configure Build Path > Add External JARs > agregamos las librerías > apply > apply and close

Agregando librerías de JUnit a nuestro proyecto (Add library):

* Creamos un proyecto JAVA
* Click derecho en el proyecto librerias > build path > configure Build Path > Add library > Junit > version > finish > appy > apply and close

Agregando librerías de JUnit a nuestro proyecto (MAVEN):

* Creamos un proyecto MAVEN
* Run As > Maven install
* Pom.xml
* Vamos a la pagina Junit about > dowload > codigo xml
* Pegamos el codigo xml en el pom.xml en la etiqueta dependencies
* <dependencies>
* <dependency>
* <groupId>junit</groupId>
* <artifactId>junit</artifactId>
* <version>4.13</version>
* <scope>test</scope>
* </dependency>
* </dependencies>
* APARECEN LA CARPETA Maven dependecias con los dos .jar

Intalación de MoreUnit:

* Instalar plugin que nos ayuda a la hora de construir tes, no es obligatorio pero nos ayuda a facilitar es llamado MoreUnit
* Help > Eclipse Marketplace > Find > MoreUnit > install > confirm > aceppt > finish > ok > yes > Reiniciar eclipse.

Uso de MoreUnit:

* Creamos proyecto MAVEN-JUnit
* Creamos una clase cualquiera
* Le damos dos atributos
* Creamos su constructor por defecto
* Creamos constructor con los parámetros
* Creamos sus setters y getters.
* CTRL + J = asistente MoreJunit
* Creamos el método de setUp en la interface
* Next > next > lo que queremos testear > finish.
* Preferences > MoreUnit > JAVA > Test method content > //given > //when > //then > apply > apply and close.
* CTRL + R = ejecutar código test.

//--------------------------------------------------------//-------------------------------------------------------//

JUNIT. TESTEANDO NUESTRAS CLASES:

Primer JUnit. Subjeto bajo test:

* Creamos un proyecto java con sus carpetas de clases y clasesTest y le agregamos las librerías de JUnit
* Creamos el los package de las carpetas
* Creamos la clase utilidades y una fución publica que concatenas Strings
* Creamos la clase testUtilidades con CTRL + J
* Creamos la función que testea la función del código con @Test
* Luego instanciamos la clase a probar
* Ejecutar el método a probar = concatenar.
* Validaciones atraves de los Assert
* Org.junit.Assert.assertDEMAScosas
* Parametros de junit es el primero un string que nos dice el error

Diferencia entre error y fallo de un test:

* FALLO: el método se ejecuta correctamente pero falla una de las Assert no conicide el resultado del test con lo que la función exige o esta esperando. Muestra el mensaje de que fallo
* SOLUCiON-FALLO: Detectar el fallo y arreglarlo ya sea en la clase o el test
* ERROR: el método no termina de ejecutarce, se crea un error que hace que termine normalmente > en este caso concatenar un string con un null
* SOLUcion-ERROR: lo mismo XD.

Anotaciones. Acciones previas y posteriores a los test(Fixtures):

* @Before Antes de cada test > es un método void
* @After Despues de cada test > es un método void
* @BeforeClass Antes de todos los test > es un método void
* @AfterClass Despues de todos los test > es un método void

ASERCiONES:

AssertEquals I. Introducción igualdad de objetos en java: