Modelos de Calidad de Software

Tema Nº7:Pruebas Unitarias con el framework JUnit – Parte 1

Indicador de logro Nº7:Construye pruebas unitarias básicas usando el framework Junit de Java.

**TEMA 01 Teoría de los**

Imagen que contiene Icono

Descripción generada automáticamente

**TEMA Nº7:**

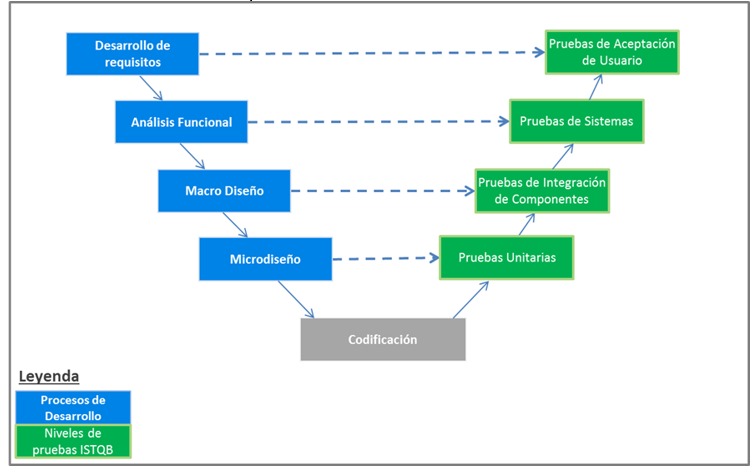
Pruebas Unitarias con el framework JUnit – Parte 1

**Subtema 7.1:**

Concepto prueba unitaria

Las pruebas unitarias considerado como el nivel más bajo de pruebas forman parte del ciclo de vida de desarrollo, suceden al finalizar la construcción, algunos puntos importantes:

* En organizaciones que usan prácticas de desarrollo agil, los desarrolladores a menudo llevan a cabo un desarrollo guiado por pruebas (TDD o Test-Driven Development) y crean pruebas unitarias automatizadas.
* Los probadores pueden revisar las pruebas unitarias para aportar sugerencias para una mayor cobertura y efectividad de estas, así como obtener un conocimmiento más profundo del software.
* En un modelo secuencial como modelo en cascada las pruebas unitarias se llevan a cabo antes de que empiece la siguiente fase.
* En algunos casos cuando el rendimiento constituye un riesgo de calidad se ejecutan las pruebas de rendimiento durante las pruebas unitarias.
* Al realizar pruebas unitarias se habrá eliminado un 50% de los defectos antes de la entrega de código al equipo de pruebas.
* El Jefe de pruebas debe tener claro que a pesar que las pruebas unitarias alcancen el 100% de su objetivo de cobertura estructural, aún quedan defectos y elementos de calidad pendientes de abordar en niveles de pruebas más altos.



Niveles ISTQB

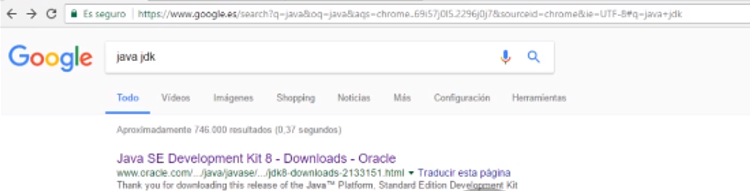
**Subtema 7.2:**

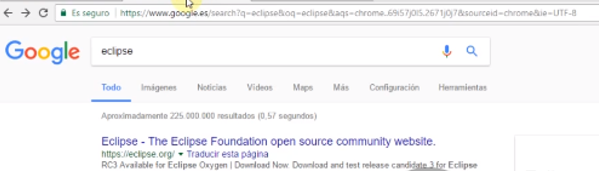
Introducción a JUNIT

El Framework JUnit Open Source nos permite realizar pruebas unitarias de nuestro código de manera controlada asegurando que el código funciona como está previsto. La buena práctica es realizar pruebas unitarias cada vez que se cambia el código, con esto abres la tendencia de reutilizar, desarrollo más rápido.

JUnit es un conjunto de librerias para realizar pruebas unitarias de desarrollos en Java con definición de clases de prueba, metodos.

Los pre requisitos de instalación antes del uso son: JDK , Eclipse.

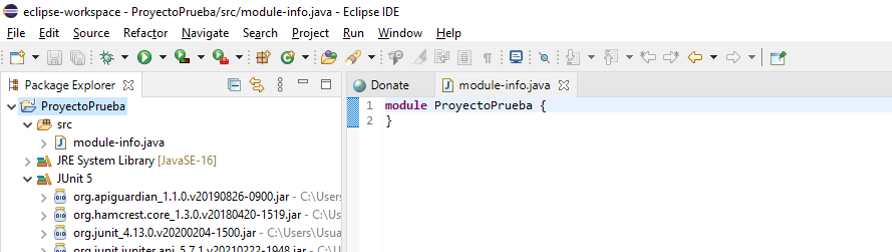




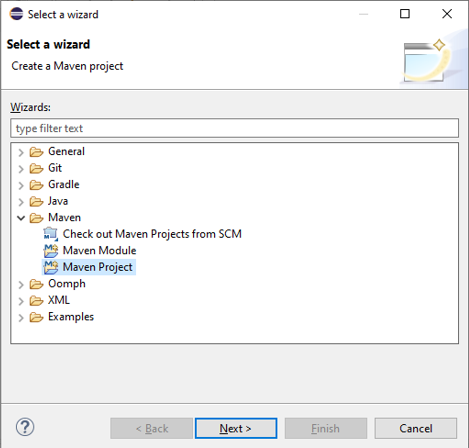


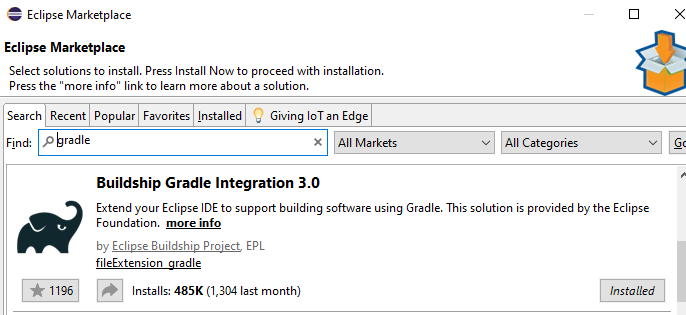
Importante adicionar librerias para un uso más rápido y seguro como Junit.jar, existen varios.



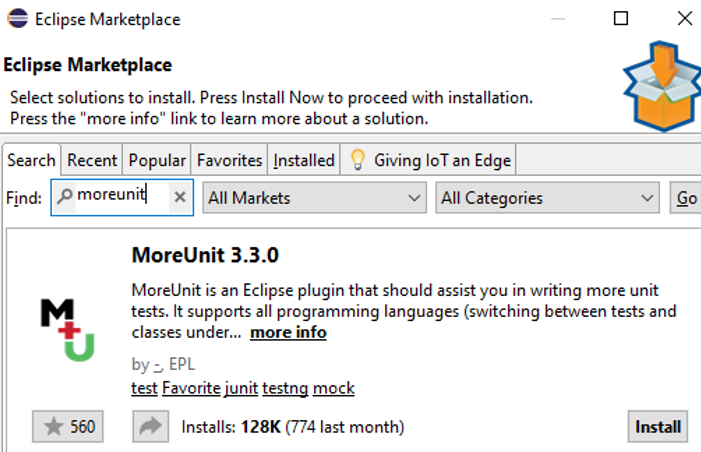


Existe varias herramientas como Maven o como el Gradle la cual se encarga de la gestión y construcción de proyectos en Java , entre todas la bondades que tiene es capaz de descargar todas las dependencias que necesitamos en nuestro proyecto, y hasta las dependencias de las dependencias, osea toda una cadena de dependencias transitivas, bastante usado enlas corporaciones.





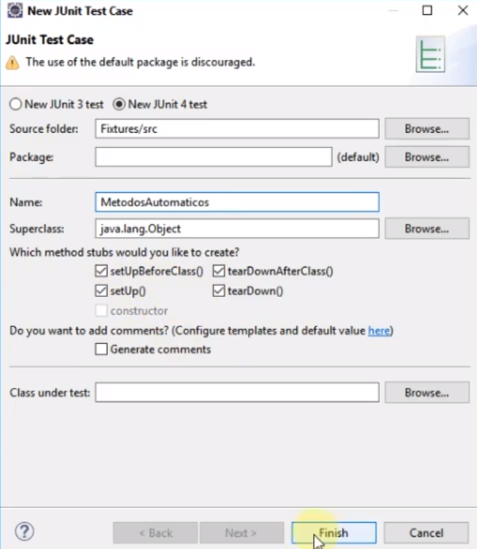
Por último también es sugerido instalar Plugin para ayudar a construir texto, brinda ventajas pero no es obligatorio, existen varios en este caso el MoreUnit.

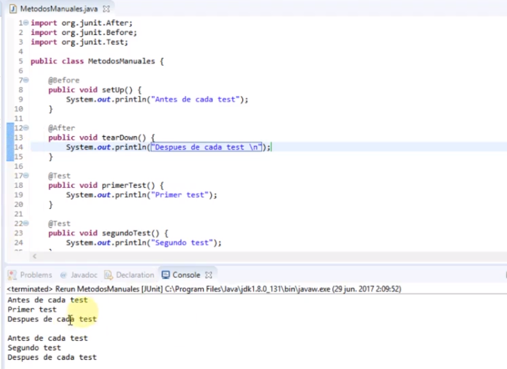


**Subtema 7.3:**

Anotaciones (After, Test, Before)

Ejecución de métodos antes y después del test, los cuales pueden hacerse de forma manual o automática.





Y para que sirve todo esto.

Por ejemplo:

* Antes de todas las clases para abrir conexión a base de datos y cerrar conexión al final de todas las clases.
* Antes de cada clase instanciar o acciones comunes

**Subtema 7.4:**

Principales Asserts en JUNIT

Los Asserts o aserciones son la parte más importante de las pruebas unitarias son las validaciones de flujos básicos o alternativos.

Esto valida nuestra implementación, existe una amplia colección de metodos Asserts, entre algunos tipos tenemos:

* AssertEquals: Evalúa si dos objetos pasados por parámetro son iguales.
* AssertTrue: Evalúa si la condición pasada es verdadera (Boolean).
* AssertFalse: Evalúa si la condición pasada es falsa (Boolean).
* AssertNull: Comprueba si un objeto es nulo.
* AssertNotNull: Comprueba que la condición no sea nula.
* AssertSame: Comprueba que dos objetos no sean la misma instancia.
* AssetNotSame: Opuesto a AssertSame, valida que dos objetos de entrada sean la misma instancia del objeto.
* AssetThat: Tipo especial, permite que se aparezcan más al Lenguaje Natural (Leer y escribir de forma sencilla).
* Fail: Se usa para hacer fallar un test explícitamente (manejo de excepciones).

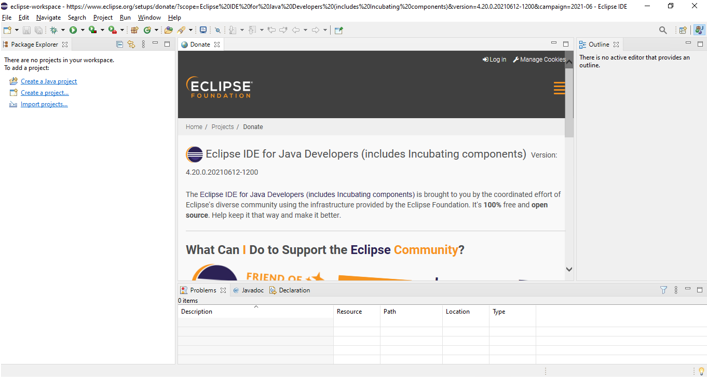
**Subtema 7.5:**

Creación y Ejecución de una prueba unitaria simple con Junit en NetBeans o Eclipse.

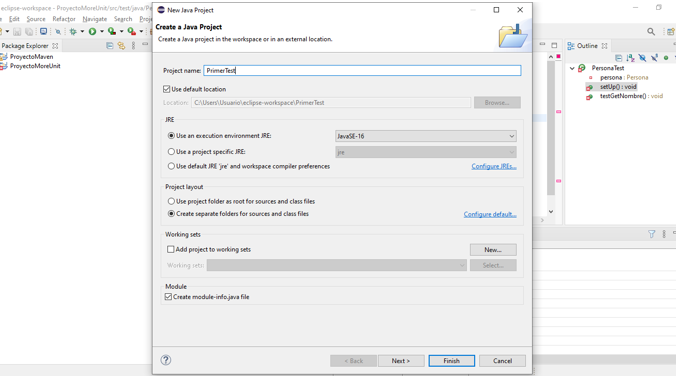
Para esta prueba simple vamos a utilizar Eclipse.

Previamente garantizamos la instalación del JDK y el Eclipse.

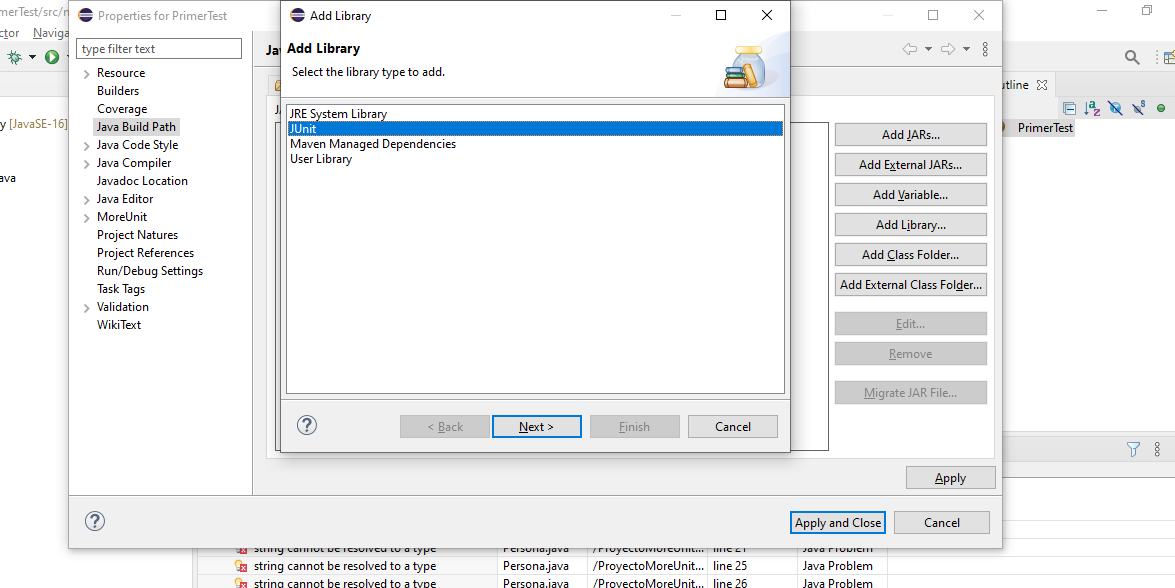




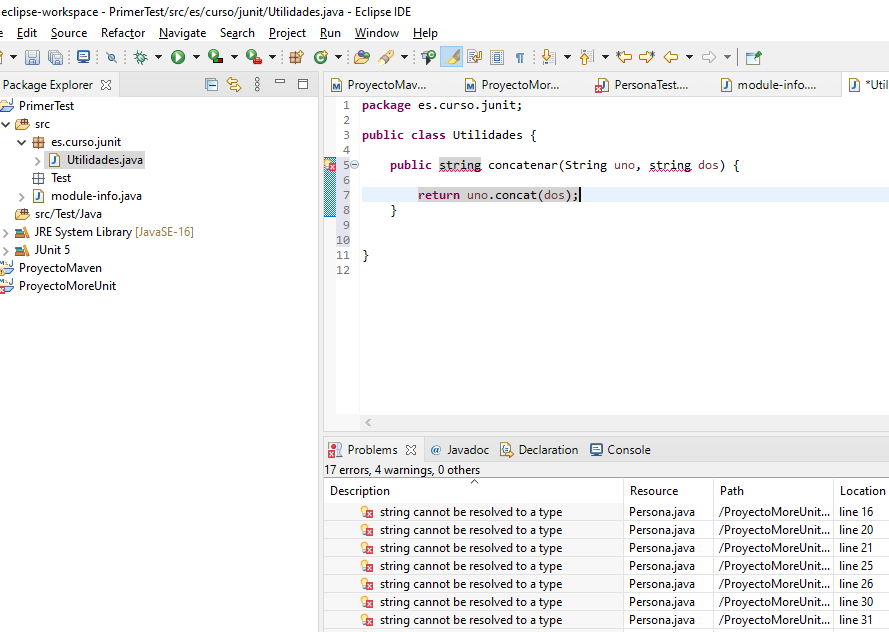
Creamos un proyecto de prueba en Java.



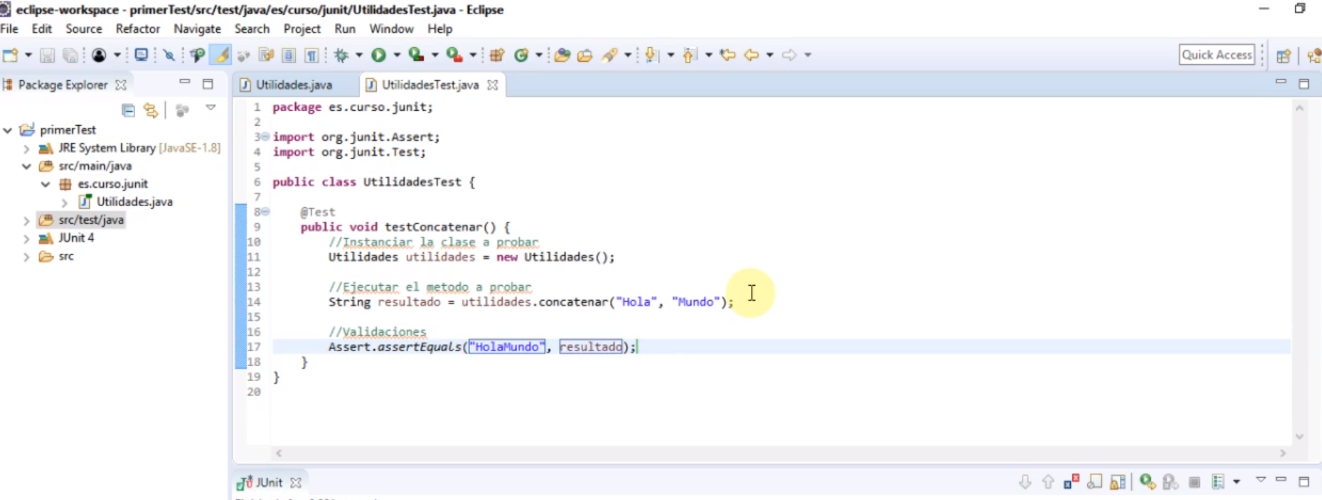
Luego de adicionar las librerías o Plugins que se desee.



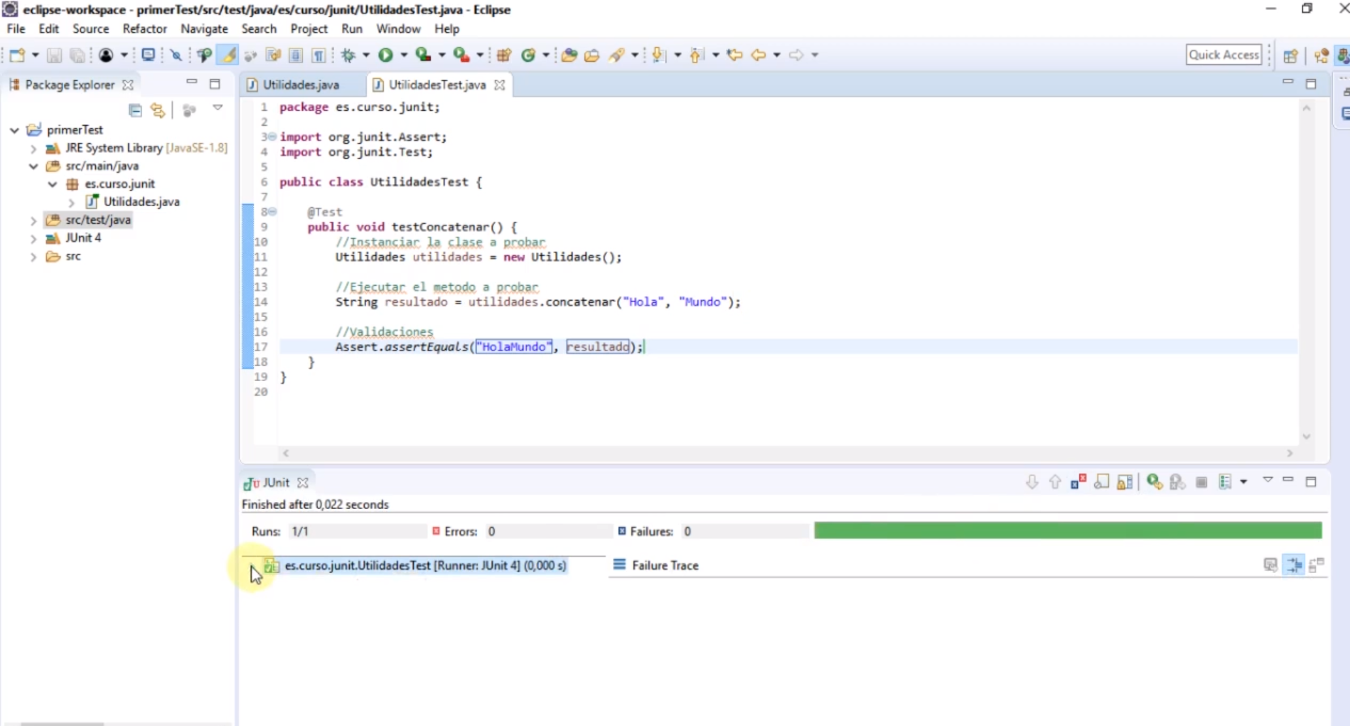
Vamos a hacer la prueba unitaria a una clase llamada Utilidades, la cual concatena.



Creamos una clase test.



Ejecutamos y podemos ver que el resultado del flujo básico de prueba es positivo.



Sólo como comentario podemos adicionar varios Assert y si modificamos Hola Mundo con espacio en blanco tendríamos otro flujo alternativo que brindaría error.

Actividad:

Revisa y analiza el enlace sugerido\* en la plataforma virtual, y luego realiza la actividad propuesta:

· ¿Qué es una Prueba Unitaria? Brindar ejemplos prácticos.

· ¿Para qué me sirven las Pruebas Unitarias?

· Crear una prueba unitaria en java con JUNIT usando gradle como motor de librerías.

\*Cada docente deberá proporcionar el enlace que considere apropiado para la ejecución de la actividad.