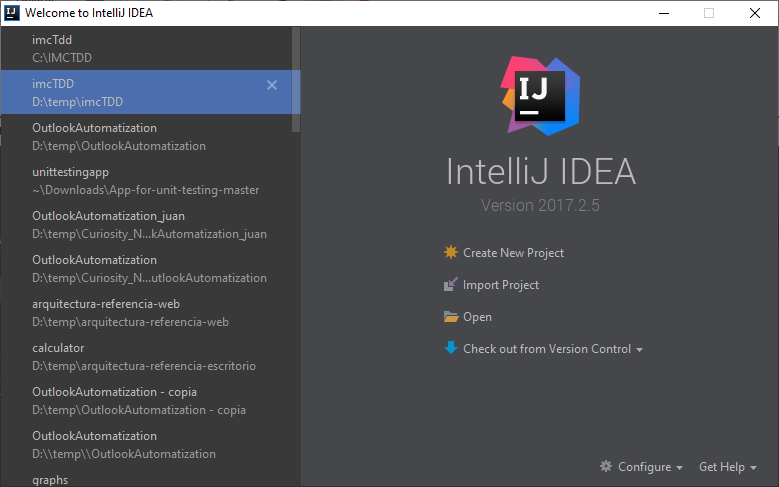
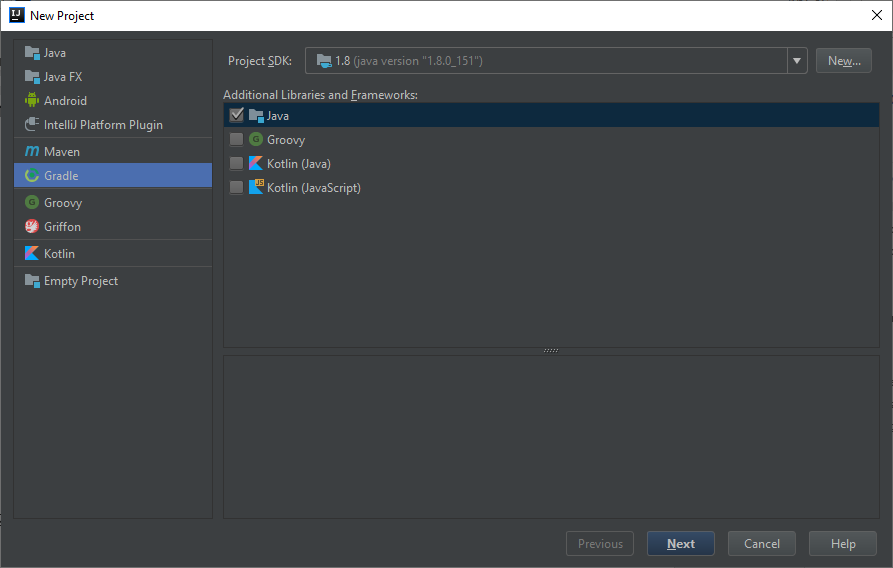
Ejercicio Pruebas Unitarias: Conceptos

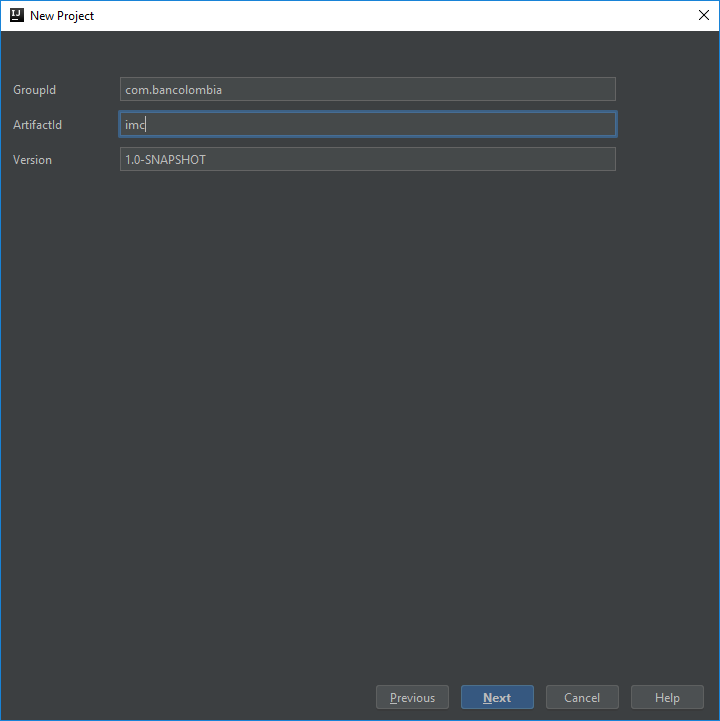
1. Abra Intellij y cree un nuevo proyecto



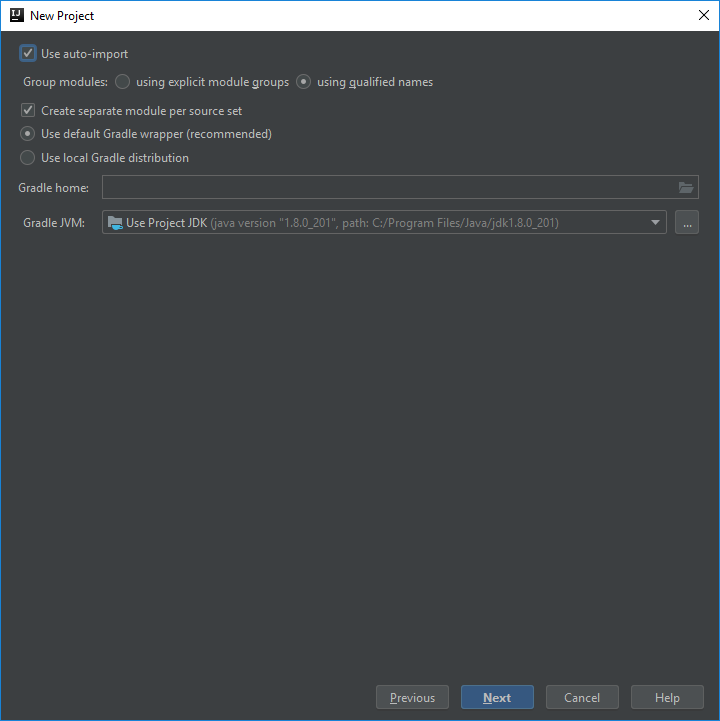
1. Seleccione la opción Gradle y posteriormente la opción Java. Finalmente presione siguiente



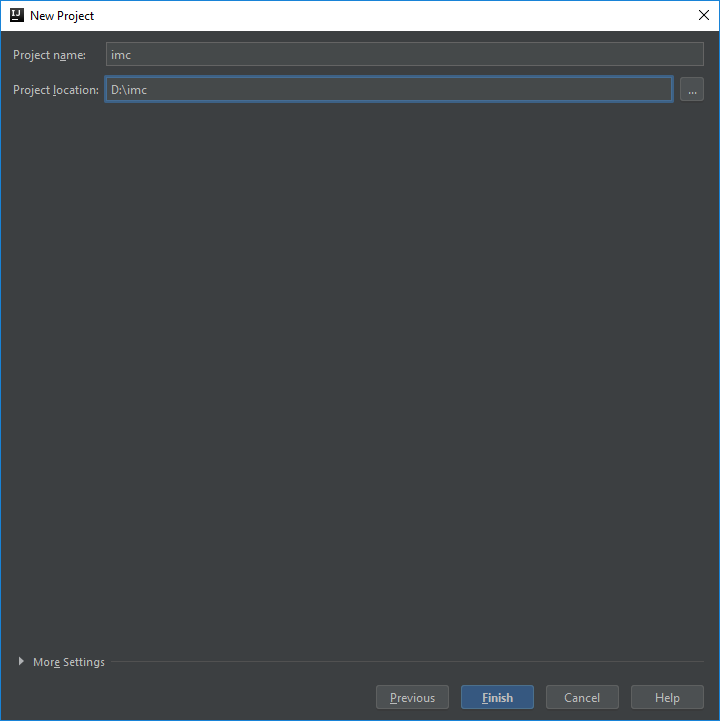
1. Agregue el GroupId, ArtifactId y Version, posteriormente presione siguiente.



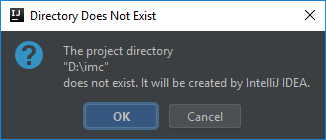
1. Realice la configuración del gradle, es decir, si va a trabajar con el gradle de la máquina o con el del proyecto. Se recomienda trabajar con el del proyecto que es Gradle Wrapper, además se recomienda seleccionar *use auto import*, pero es una configuración opcional. Por último, seleccione siguiente.



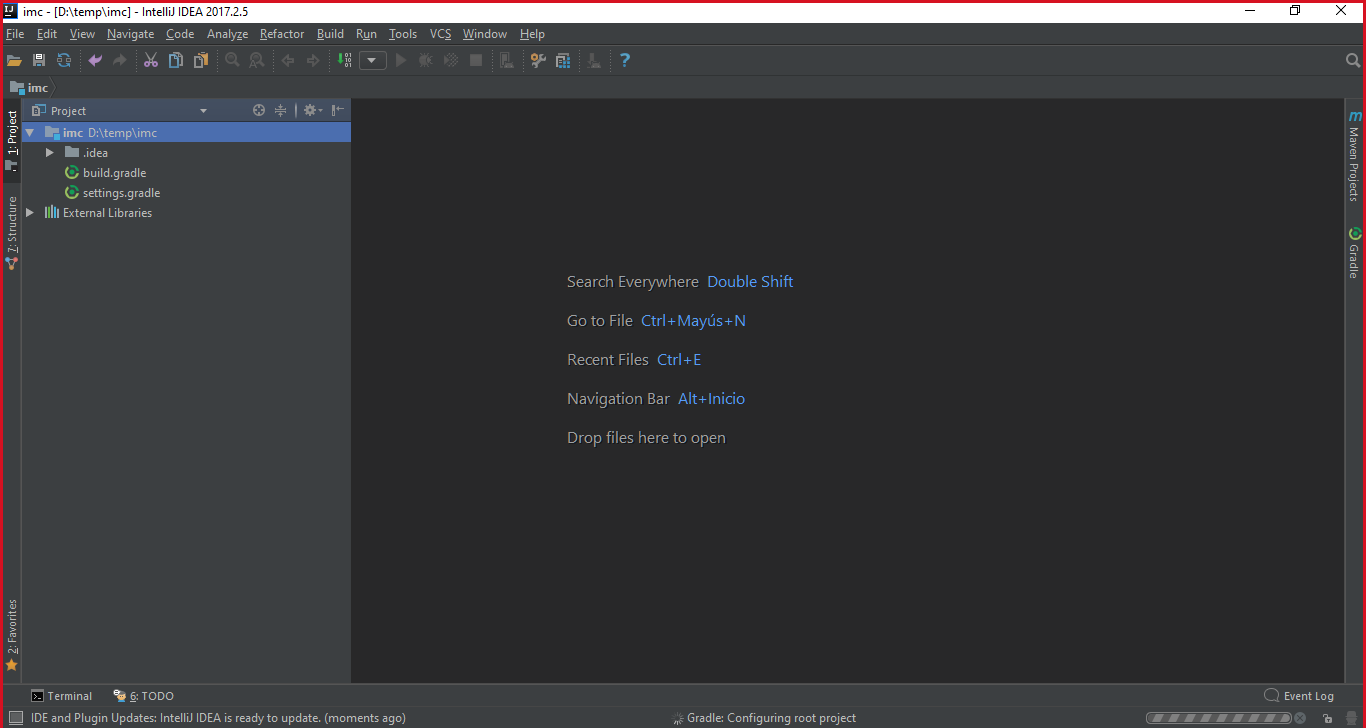
1. Seleccione el nombre de la carpeta donde se almacenará el proyecto y la ubicación de esta. Posteriormente seleccione finalizar.



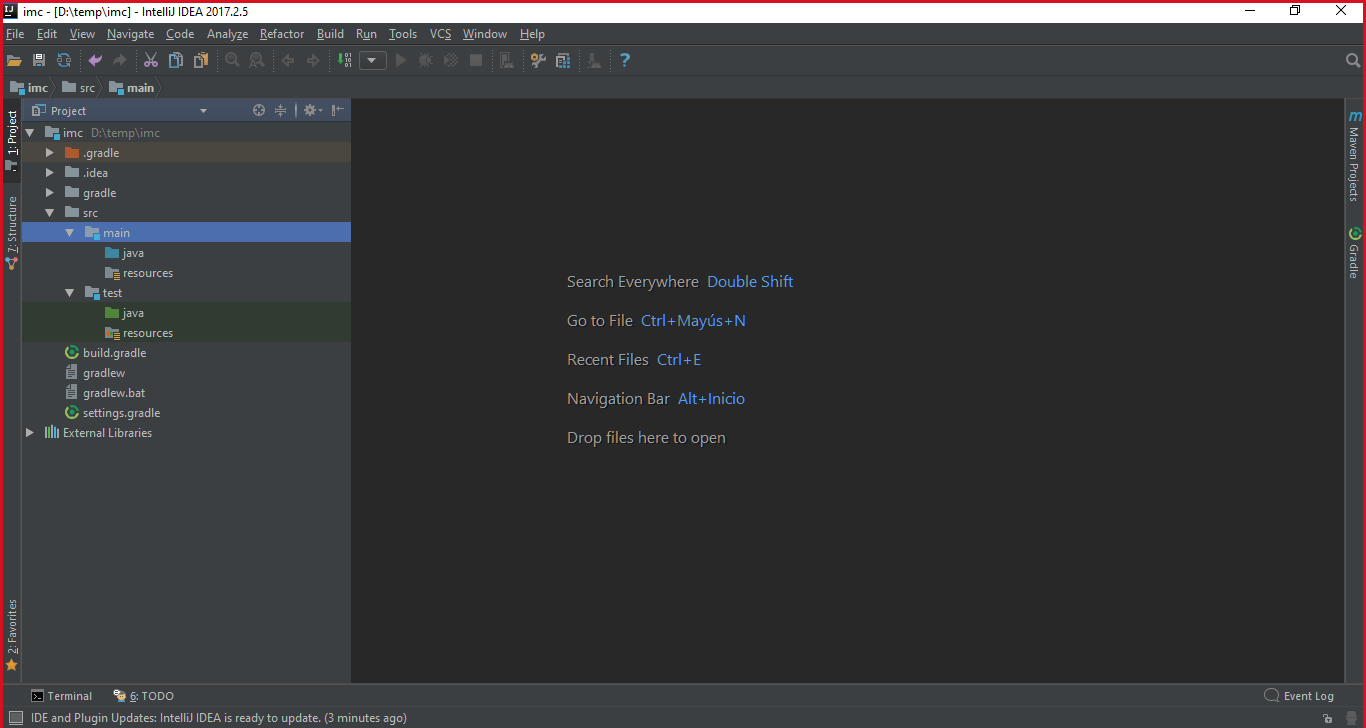
1. Si sale la siguiente ventana, por favor seleccione la opción, con lo cual, esta indicando que la carpeta donde se va a almacenar el proyecto no esta creada y se le da autorización a IntelliJ para crearla.



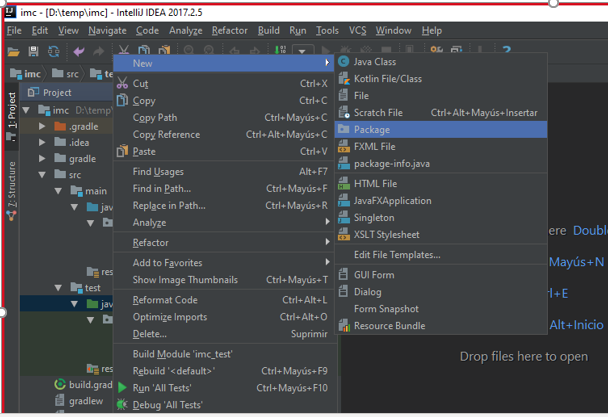
1. Se creará el proyecto y gradle comenzará a compilarlo.



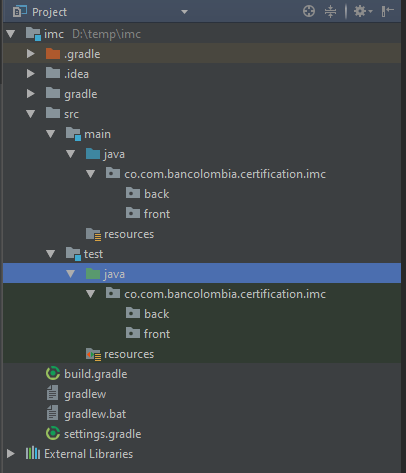
Una vez compilado el proyecto se muestra en el IDE así:



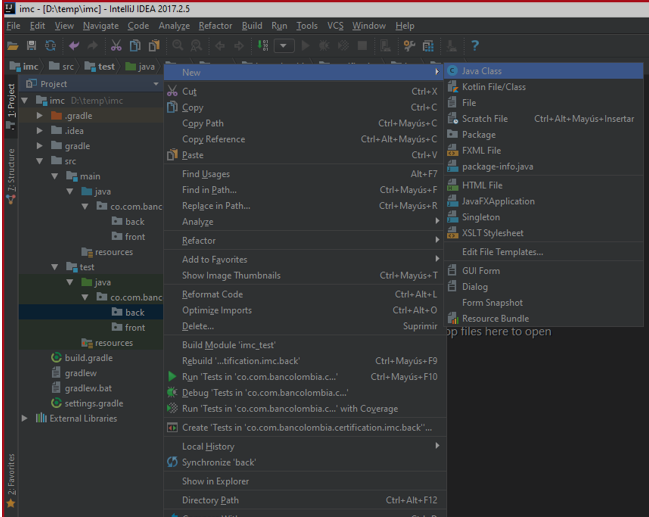
1. Creamos los paquetes para nuestro proyecto, tanto en el main como en el test. De acuerdo con el estándar Java, debemos nombrar los paquetes raíz con el inverso al dominio de la empresa, le adjuntamos el área y el nombre del aplicativo. En nuestro caso sería co.com.bancolombia.certification.imc



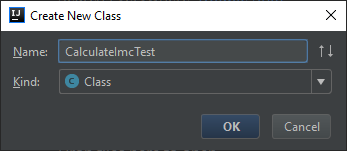
Teniendo como resultado



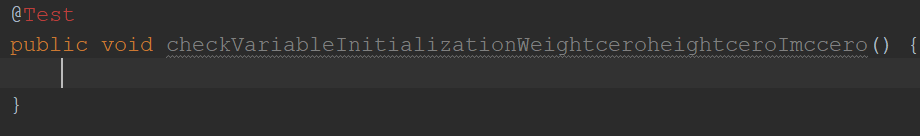
1. Creamos nuestras clases para codificar las pruebas unitarias. Estas nos permitirán crear nuestro software usando T.D.D.



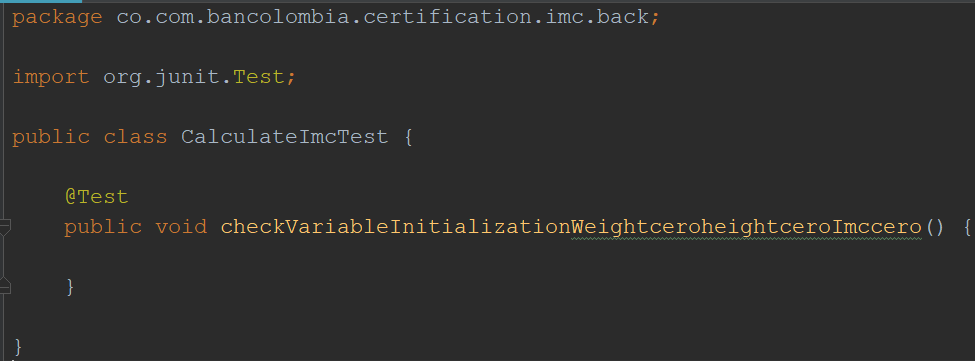
Nombramos la clase tal como lo haríamos cuando construimos el sistema de información, pero agregamos el sufijo Test. Presionamos ok



1. Construimos nuestra primera prueba. En esta vamos a verificar que el peso y la estatura, de la clase CalculateImc, sean inicializadas en cero.



Vemos que tenemos un error con la notación @Test, esto se debe a que necesitamos importar la librería **org.junit.Test.** Una vez importada la librería tendremos la clase CalculateImcTest de la siguiente manera

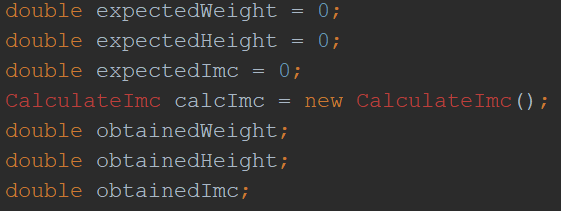


Codificamos nuestra prueba, aplicando el patrón AAA.

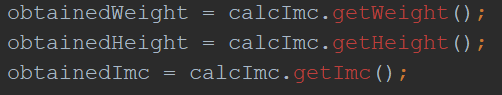
Recuerden que, de acuerdo con T.D.D. primero debe fallar la prueba.

Miremos las partes del código.

**Arrange**

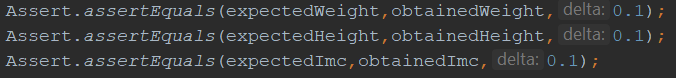


**Act**

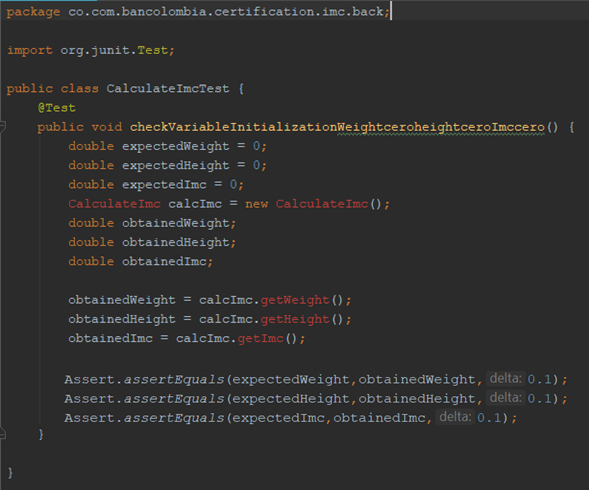


**Assert**

Aquí debemos importar la librería **org.junit.Assert**

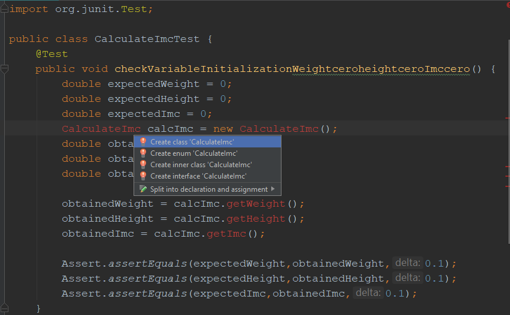


La clase completa se ve así

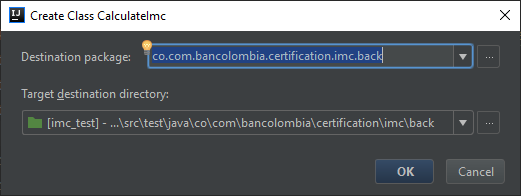


1. Creamos el código necesario para que nuestras pruebas no fallen.

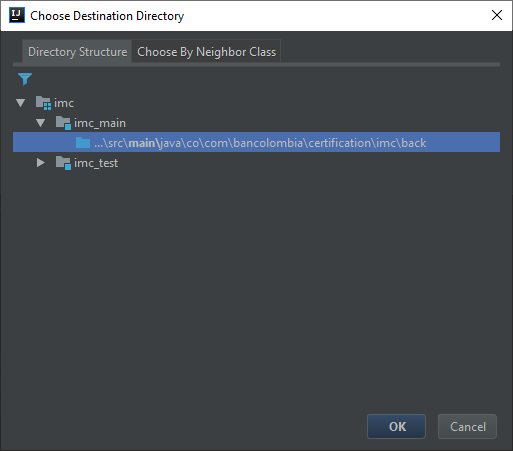
Presionamos el foco rojo que aparece y el IDE nos ofrece la posibilidad de crear esta clase.



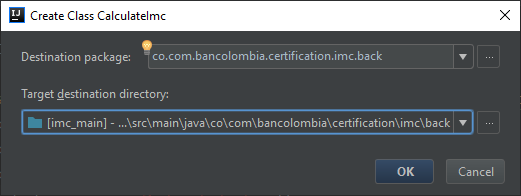
Demos click en **Crete class CalculateImc** y el IDE nos muestra esta ventana. Damos click en el botón de puntos



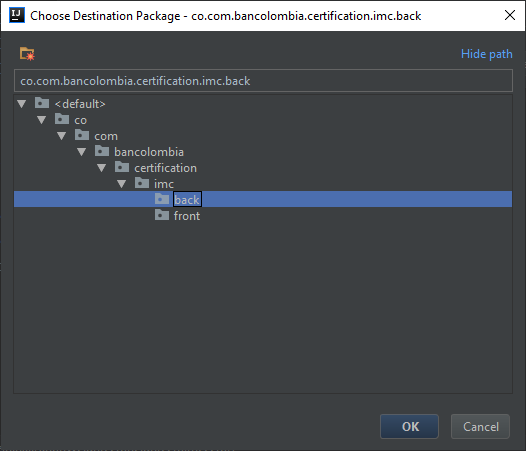
Nos aparece una nueva ventana. En esta vamos a elegir el sitio donde queremos crear nuestra clase. Vamos a elegir la siguiente ruta y presionamos ok



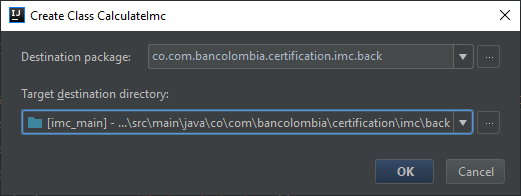
Ahora el IDE nos llevara de regreso a la ventana **Create Class.** Allí vamos a elegir el paquete donde crearemos la clase. Para ello hacemos click en en el botón de 3 puntos, como se muestra en la imagen



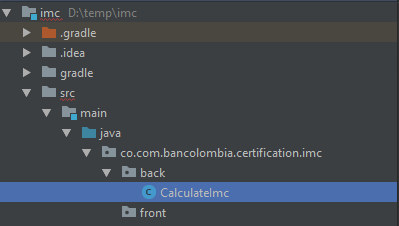
El IDE nos muestra la ventana Choose Destination Packge. Aquí elegimos el paquete back y presionamos ok



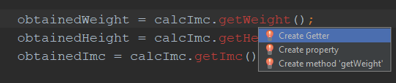
Regresamos a la ventana **Create Class,** presionamos OK

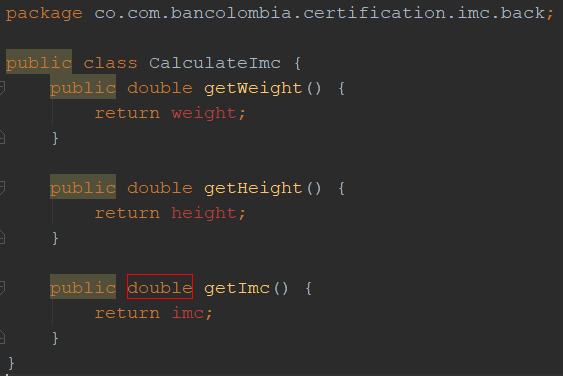


En este punto ya tenemos la clase CalculateImc creada.

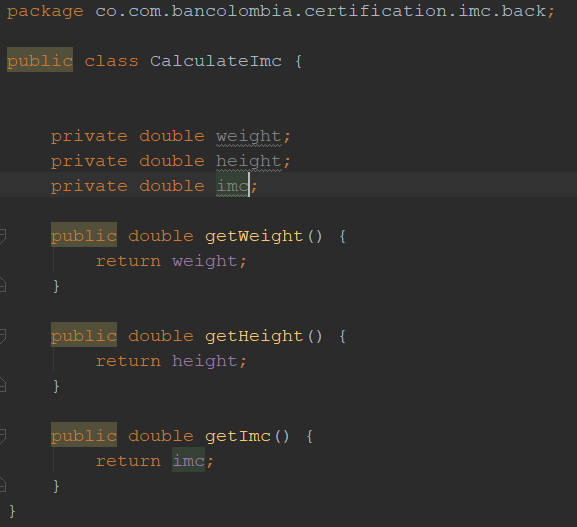


1. Vamos a la clase CalculateImcTest y continuamos reparando errores. Debemos crear, en la clase CalculateImc, cada una de las propiedades. Para ello posicionamos el cursor en cada propiedad, presionamos Ctrl + Enter y elegimos la opción Create Getter. Esto creará las propiedades.



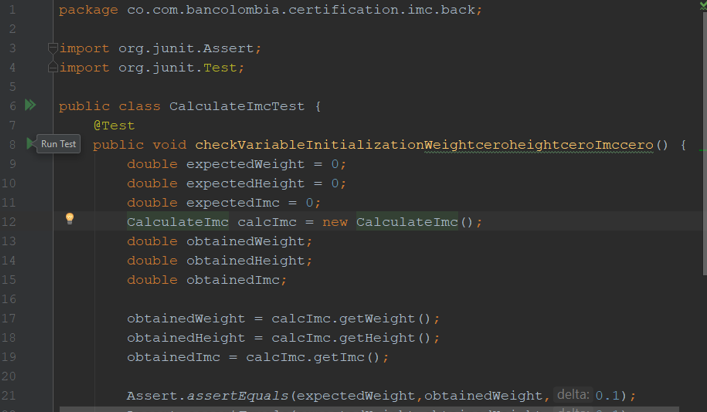


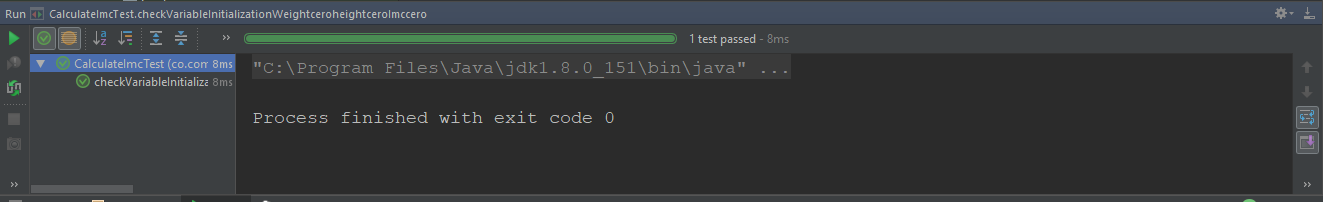
Cree las globales weight, height e imc en la clase CalculateImc. La clase se verá así



Ya hemos construido el código necesario para que nuestra prueba no falle. Aquí ya estamos en el segundo paso de T.D.D

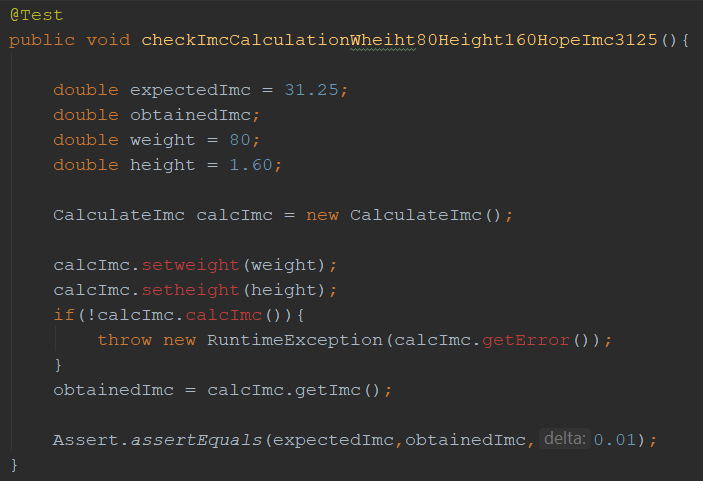
1. Ejecutamos nuestra prueba unitaria dando click en la flecha verde. Esta no debe fallar



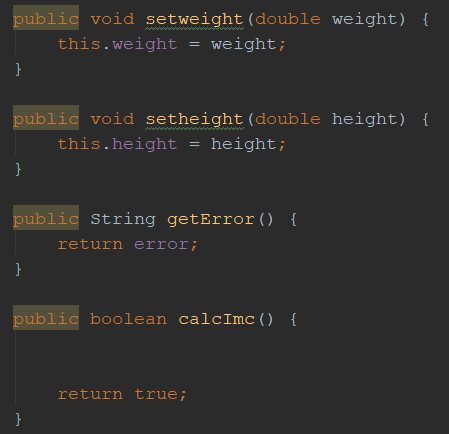


1. Vamos a crear otra prueba, usando T.D.D. La prueba validará que el sistema retorne IMC igual a 31.25 cuando ingreso el peso en 80 Kg y la estura en 1.60 mts.

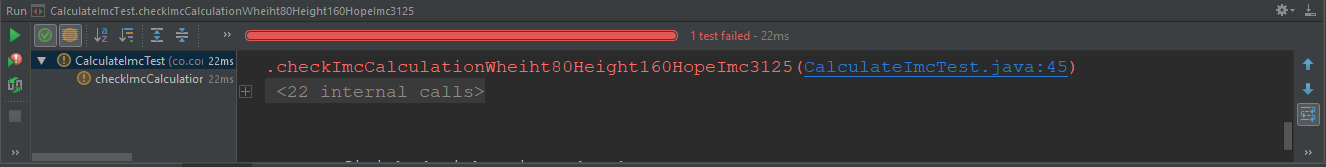
Escribimos la prueba



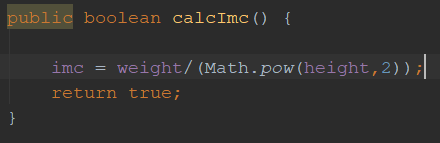
Esta nos muestra errores de compilación, los cuales debemos solucionar escribiendo el código necesario en la clase CalculateImc.



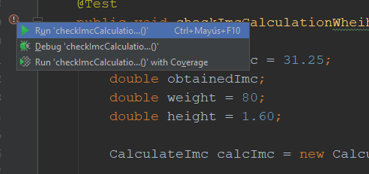
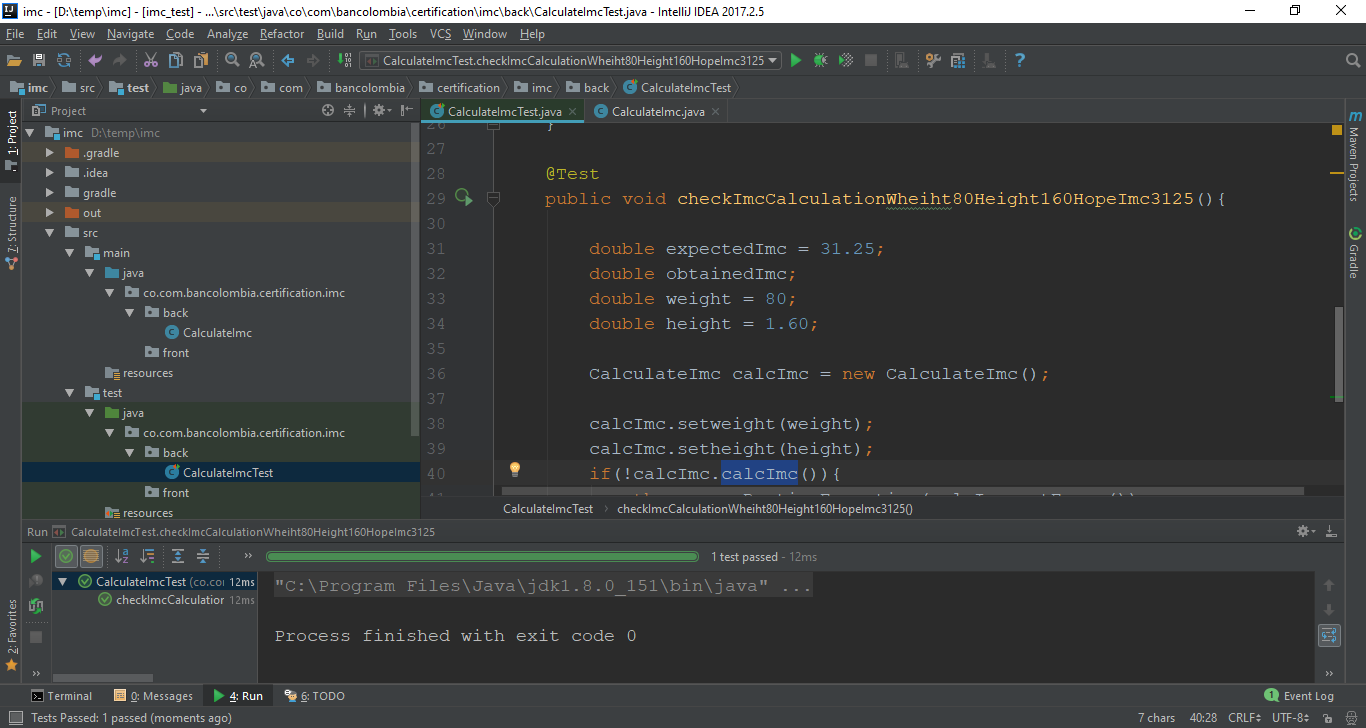
Una vez corregidos los errores de compilación, ejecutamos la prueba y esta debe fallar. Este es el primer paso del ciclo T.D.D.



Corregimos la lógica del método calcImc, para que funcione. Aquí debemos escribir solo el código necesario para que esta prueba pase. Este es el segundo paso del ciclo T.D.D

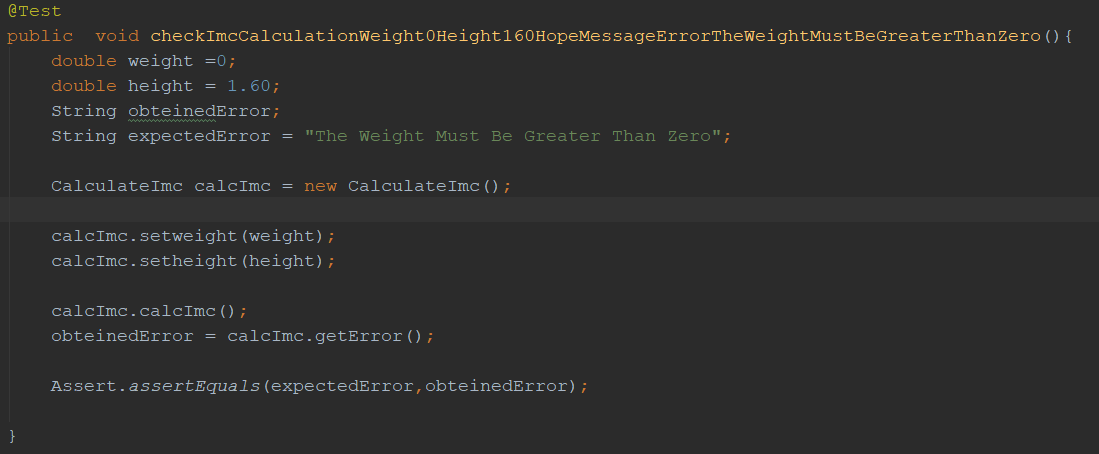


Ejecutamos de nuevo la prueba y esta debe ser exitosa. Este es el tercer paso del ciclo T.D.D

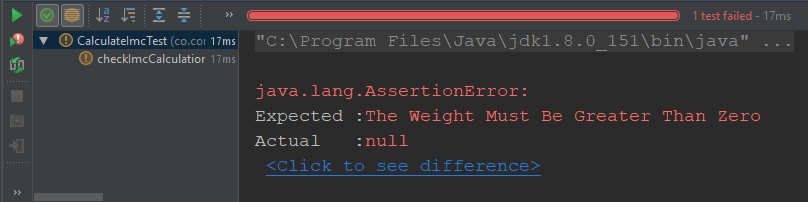


1. Creamos la siguiente prueba. La prueba va a validar que el sistema retorne un mensaje de error cuando le ingreso el peso en 0 kg y la altura en 1.60 mts.

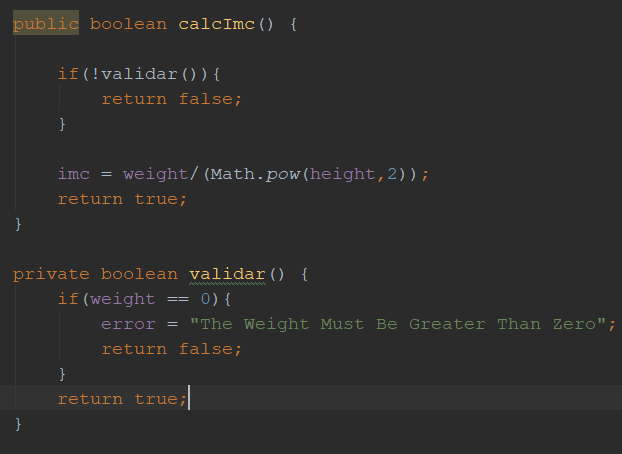
Escribimos la prueba



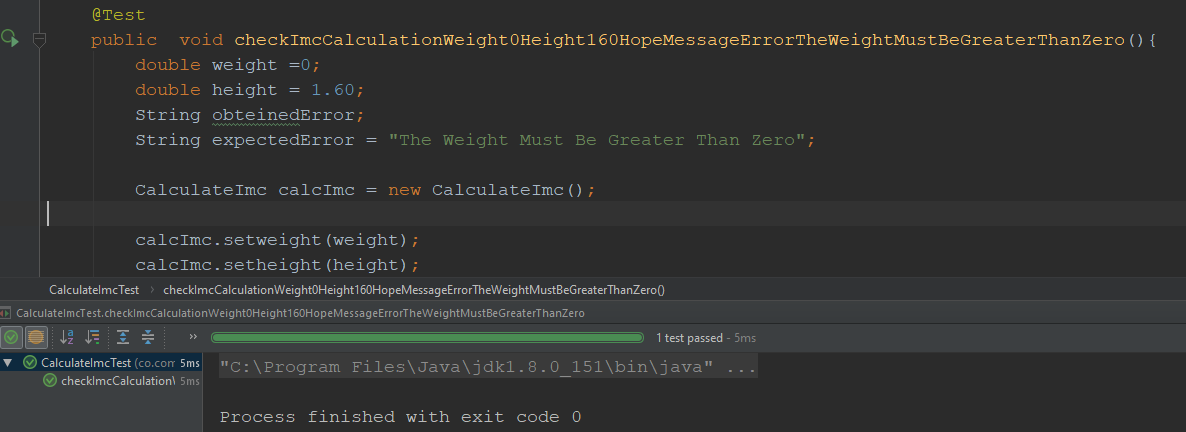
Ejecutamos la prueba obteniendo un error



Corregimos el código en CalculateImc. Recuerde escribir solo el código necesario para que la prueba pase



Ejecutamos la prueba y esta resulta exitosa



Finalmente optimizamos nuestro código: Las pruebas y el sistema de información.

En nuestro caso, podemos poner la instancia de la clase CalculateImc en un método etiquetado con el @Before y quitar, de cada test, esta instancia.

