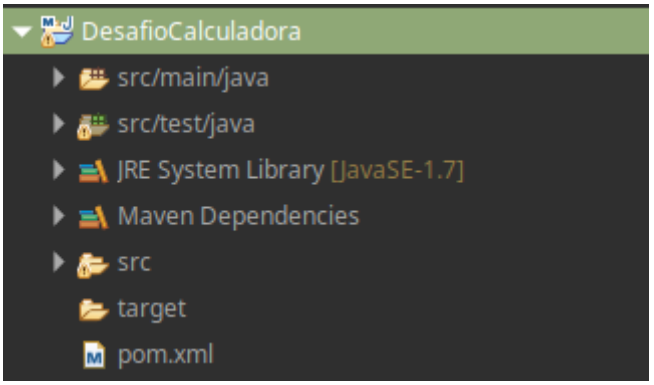
Solución Desafío - Calculadora (Parte I)

Paso 1: Se crea un nuevo proyecto con Maven, el cual contiene la siguiente estructura de carpetas.



Se pueden borrar las clases Java generadas en main y en test (App.java, AppTest.java) para crear algunas propias.

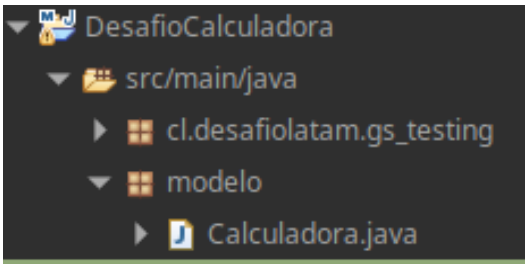
Paso 2: Se usa la versión más reciente de JUnit, la cual consiste en una librería adicional de Maven que se debe agregar al proyecto. Esta corresponde a la siguiente dependencia:

```
<dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <version>4.11</version>
  <scope>test</scope>
</dependency>
```

Se debe ir al archivo `pom.xml` en la raíz del proyecto, se borra la dependencia de JUnit que viene por defecto y se agrega la nueva dentro del tag dependencies. El archivo pom.xml sería el siguiente:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
 <modelVersion>4.0.0</modelVersion>
 <groupId>cl.desafiolatam</groupId>
 <artifactId>calculadora-test</artifactId>
 <version>1.0-SNAPSHOT</version>
 <name>calculadora-test</name>
<!-- FIXME change it to the project's website -->
<url>http://www.example.com</url>
<properties>
<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
<maven.compiler.source>1.7</maven.compiler.source>
<maven.compiler.target>1.7</maven.compiler.target>
</properties>
 <dependencies>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.11</version>
    <scope>test</scope>
   </dependency>
 </dependencies>
  <!--resto del archivo-->
</project>
```

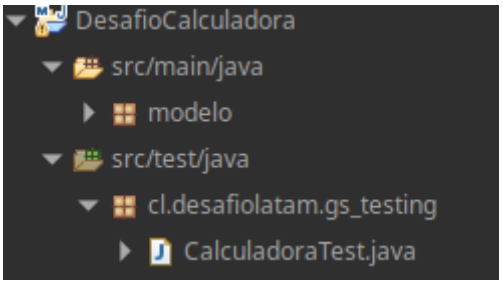
Paso 3: Una vez creado el proyecto, debemos crear la clase Calculadora dentro de la carpeta src/main del proyecto, en la siguiente ruta:



Paso 4: Creamos los métodos que realizarán operaciones en la clase Calculadora: sumar, restar, multiplicar y dividir.

```
public class Calculadora {
 public Integer sumar(Integer a, Integer b) {
  if (a != null && b != null ) {
  return a+b;
 } else {
  return 0;
 }
}
public Integer restar(Integer a, Integer b) {
  if (a != null && b != null ) {
   return a-b;
  } else {
  return 0;
 }
}
public Integer multiplicar(Integer a, Integer b) {
  if (a != null && b != null ) {
  return a*b;
 } else {
  return 0;
 }
}
public Integer dividir(Integer a, Integer b) {
  if (a != null && b != null ) {
  return a/b;
 } else {
  return 0;
  }
 }
}
```

Paso 5: Creamos la clase de prueba CalculadoraTest al interior de src/test/java para saber si los métodos de la clase Calculadora funcionan como deberían. Estos podemos ejecutarlos nosotros mismos o estar seguros de que el código no tiene errores, pero escribiendo sus pruebas unitarias para que la verificación sea segura y evitar así tener efectos secundarios.



Esta será la clase que contendrá las pruebas, inicialmente se instancia la clase Calculadora.

```
package cl.desafiolatam;
import org.junit.jupiter.api.Test;

public class CalculadoraTest {
  private Calculadora calculadora = new Calculadora();
}
```

Paso 6: Probamos el método sumar dentro de la nueva clase CalculadoraTest creada:

```
La clase CalculadoraTest, test para método sumar:
package cl.desafiolatam;
import org.junit.jupiter.api.DisplayName;
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.assertEquals;

  public class CalculadoraTest {
   private Calculadora calculadora = new Calculadora();
   @Test
   public void testSumar() {
   Integer resultado = calculadora.sumar(1,1);
   assertEquals(Integer.valueOf(2),resultado);
  }
}
```

Paso 7: Probamos el método restar dentro de la nueva clase CalculadoraTest creada:

```
package cl.desafiolatam;
import org.junit.jupiter.api.DisplayName;
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.assertEquals;

 public class CalculadoraTest {
   private Calculadora calculadora = new Calculadora(); //resto de la
clase
   @Test
   public void testRestar() {
   Integer resultado = calculadora.restar(111,11);
   assertEquals(Integer.valueOf(100),resultado);
  }
}
```

Paso 8: Probamos el método multiplicar dentro de la nueva clase CalculadoraTest creada:

```
package cl.desafiolatam;
import org.junit.jupiter.api.DisplayName;
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.assertEquals;

 public class CalculadoraTest {
   private Calculadora calculadora = new Calculadora(); //resto de la
clase
   @Test
   public void testMultiplicar() {
   Integer resultado = calculadora.multiplicar(4,4);
     assertEquals(Integer.valueOf(16),resultado);

 }
}
```

Paso 9: Probamos el método sumar dentro de la nueva clase CalculadoraTest creada:

```
package cl.desafiolatam;
import org.junit.jupiter.api.DisplayName;
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.assertEquals;

 public class CalculadoraTest {
   private Calculadora calculadora = new Calculadora(); //resto de la
clase
   @Test
   public void testDividir() {
   Integer resultado = calculadora.dividir(9,3);
   assertEquals(Integer.valueOf(3),resultado);
  }
}
```

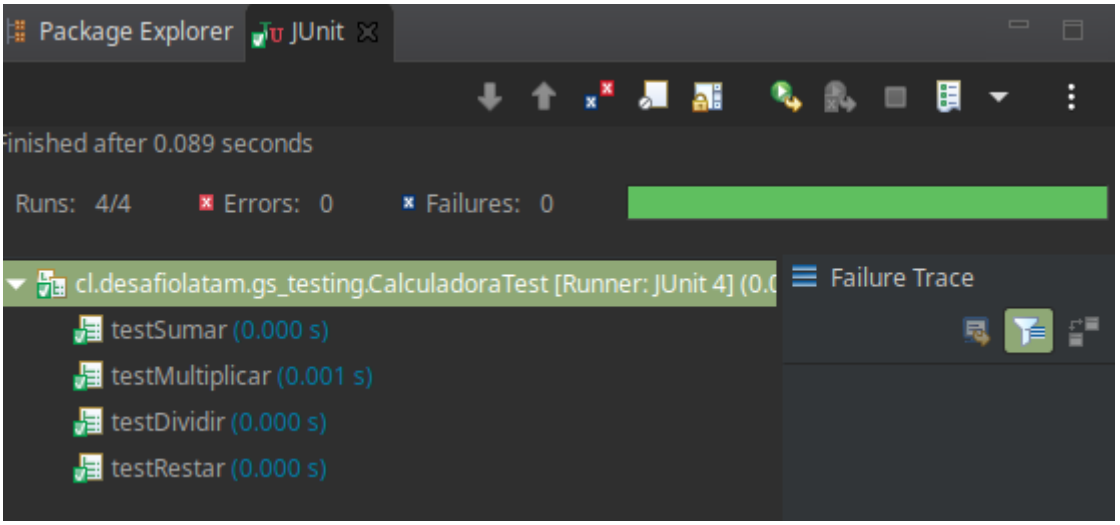
La clase CalculadoraTest completa debe lucir así:

```
package cl.desafiolatam;
import org.junit.jupiter.api.DisplayName;
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.assertEquals;

 public class CalculadoraTest {
  private Calculadora calculadora = new Calculadora();
   @Test
      public void testSumar() {
      Integer resultado = calculadora.sumar(1,1);
      assertEquals(Integer.valueOf(2),resultado);
    }
      @Test
      public void testRestar() {
      Integer resultado = calculadora.restar(111,11);

      assertEquals(Integer.valueOf(100),resultado);
    }
      @Test
      public void testMultiplicar() {
      Integer resultado = calculadora.multiplicar(4,4);
      assertEquals(Integer.valueOf(16),resultado);
    }
      @Test
      public void testDividir() {
      Integer resultado = calculadora.dividir(9,3);
      assertEquals(Integer.valueOf(3),resultado);
     }
}
```

Paso 10: Ejecutar el comando Maven test para que la salida sea:




```
[INFO] ------------------------------------------------------- [INFO] T
E S T S
[INFO] ------------------------------------------------------- [INFO]
Running cl.desafiolatam.CalculadoraTest
[INFO] Tests run: 4, Failures: 0, Errors: 0, Skipped: 0, Time elapsed:
0.025 s - in cl.desafiolatam.CalculadoraTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 4, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO]
------------------------------------------------------------------------
-
[INFO] BUILD SUCCESS
[INFO]
------------------------------------------------------------------------
-
[INFO] Total time: 2.960 s
[INFO] Finished at: 2019-07-09T12:22:28-04:00
[INFO]
------------------------------------------------------------------------
-
```