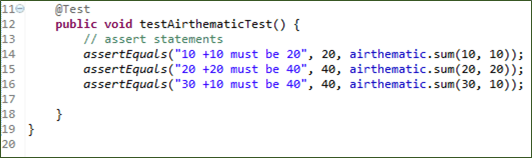
**What is Parameterized Test in Junit?**

Parameterized test is to execute the same test over and over again using different values. It helps developers to save time in executing the same test which differs only in their inputs and expected results.

Using a Parameterized test, one can set up a test method that retrieves data from some data source.

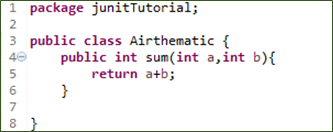
Consider a simple test to sum different numbers. The code may look like –



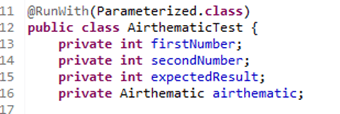
## Steps to create a Parameterized JUnit test

Following code shows an example for a parameterized test. It tests sum() method of the Arithmetic class :

**Step 1)**Create a class.In this example, we are going to input two numbers by using sum (int,int) method which will return the sum of given numbers



**Step 2)**Create a parameterized test class



Code Explanation

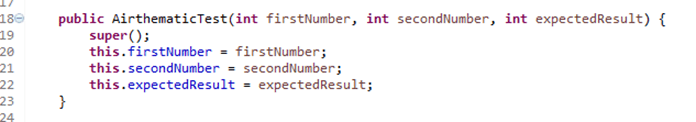
* **Code Line 11:**Annotate your test class using @RunWith(Parameterized.class).
* **Code Line 13:**Declaring the variable 'firstNumber' as private and type as int.
* **Code Line 14:**Declaring the variable 'secondNumber'as private and type as int.
* **Code Line 15:**Declaring the variable 'expectedResult'as private and type as int.
* **Code Line 16:**Declaring the variable 'airthematic'as private and type as Airthematic.

**@RunWith**(class\_name.class): **@RunWith** annotation is used to specify its runner class name. If we don't specify any type as a parameter, the runtime will choose **BlockJunit4ClassRunner** by default.

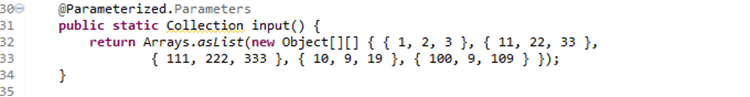
This class is responsible for tests to run with a new test instance. It is responsible for invoking JUnit lifecycle methods such as setup(associate resources) and teardown(release resources).

To parameterize you need to annotate using @RunWith and pass required .class to be tested

**Step 3)**Create a constructor that stores the test data. It stores 3 variables



**Step 4)** Create a static method that generates and returns test data.



**Code Line 32,33:**Creating a two-dimensional array (providing input parameters for addition). Using asList method we convert the data into a List type. Since, the return type of method input is collection.

**Code Line 30:**Using **@Parameters** annotation to create a set of input data to run our test.

The static method identified by @Parameters annotation returns a Collection where each entry in the Collection will be the input data for one iteration of the test.

Consider the elements

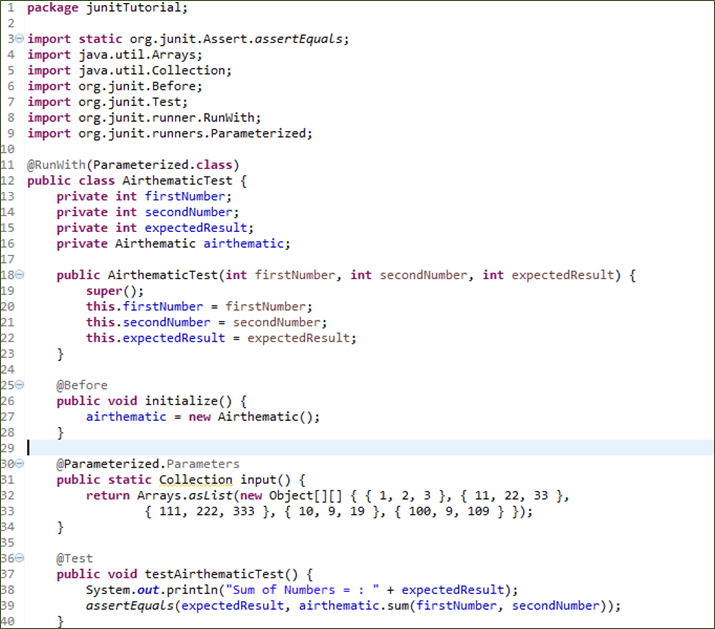
{1,2,3}

Here

firstNumber =1

secondNumber=2

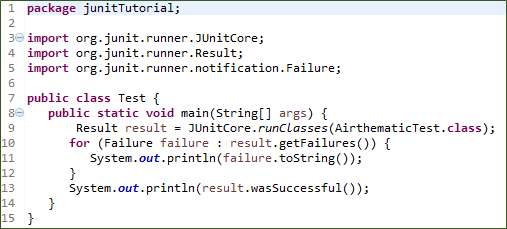
expectedResult=3



**Code Explanation:**

* **Code Line 25:**Using @Before annotation to setup the resources (Airthematic.class here). The @Before annotation is used here to run before each test case. It contains precondition of the test.
* **Code Line 36:**Using @Test annotation to create our test.
* **Code Line 39:**Creating an assert statement to check whether our sum is equivalent to what we expected.

**Step 6)**Create a test runner class to run parameterized test:



**Code Explanation:**

* **Code Line 8:**Declaring the main method of the class test which will run our JUnit test.
* **Code Line 9:**Executing test cases using JunitCore.runclasses, it will take the testclass name as a parameter (In our example we are using Airthematic.class).
* **Code Line 11:**Processing the result using for loop and printing out failed result.
* **Code Line 13:**Printing out the successful result.