**Grouped Assertions In JUnit 5**

JUnit 5 supports an additional feature called Grouped assertions. When you have to execute multiple assertions together and in turn, get one consolidated report, then grouped assertion comes to your rescue. The assertion method – assertAll () facilitates this feature.

**There are 6 versions of assertAll methods:**

| **Method variations** | **Description** |
| --- | --- |
| static void assertAll (String heading, Collection executables) | Verifies that no exception is thrown by all the executables of type Collection provided as input parameter |
| static void assertAll (String heading, Stream executables) | Verifies that no exception is thrown by all the assert functions of different Stream provided as input parameter |
| static void assertAll (String heading, Executable... executables) | Verifies that no exception is thrown by all the supplied executables provided as input parameter |
| static void assertAll (Collection executables) | Verifies that no exception is thrown by all the executables of type Collection provided as input parameter. This method doesn’t include header as the parameter |
| static void assertAll (Stream executables) | Verifies that no exception is thrown by all the supplied executables of different streams provided as input parameter. This method doesn’t include header as the parameter |
| static void assertAll (Executable... executables) | Verifies that no exception is thrown by all the supplied executable. This method doesn’t include header as the parameter |

**Grouped Assertions With Heading As Parameter**

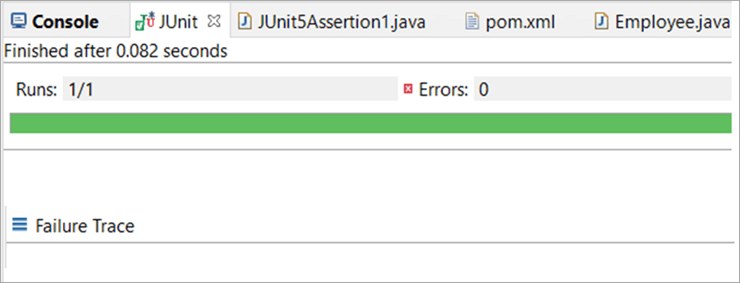
**Example 1:**

Here is an example where assertEquals () and assertIterableEquals () are grouped together using the method assertAll (). It consists of the heading parameter with the value “GroupedAssertionHeading”.

|  |
| --- |
| class JUnit5Assertion1 {       @Test       public void groupedAssertionTest () {                         float a=(float) 1.2;              float b=(float) 1.2;              float delta=(float) 1.0;              Iterable<String> expectedList = new ArrayList<>(Arrays.asList("First", "Two",   "Third"));              Iterable<String> actualList = new ArrayList<> (Arrays.asList("First", "Two",   "Third"));              assertAll (              "GroupedAssertionHeading",              () -> assertEquals (a, b, delta, "assert with delta"),              () -> assertIterableEquals (expectedList, actualList)       );         }  } |

**Result :**

As both asserts are passed, the final result passes.

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2021/03/Code-for-JUnit5Assertion1.java_.png)

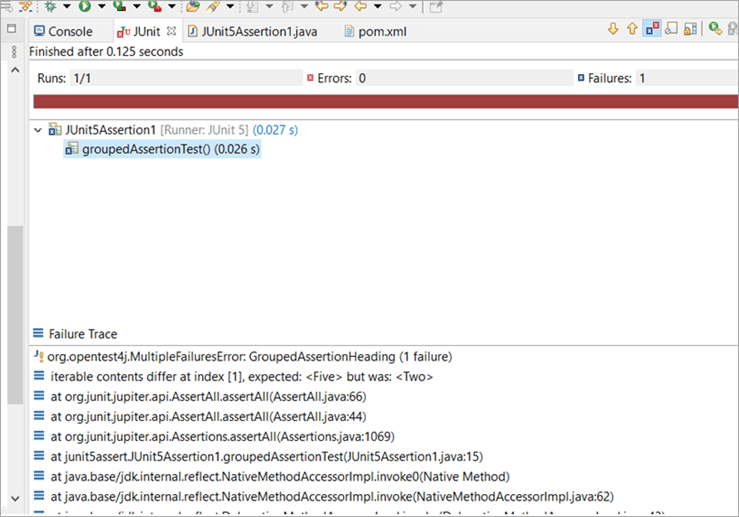
**Example 2:**

Here the same example is modified in such a way that assertIterableEquals () fails:

|  |
| --- |
| class JUnit5Assertion1 {       @Test       public void groupedAssertionTest () {                         float a=(float) 1.2;              float b=(float) 1.2;              float delta=(float) 1.0;              Iterable<String> expectedList = new ArrayList<> (Arrays.asList("First", "Five", "Third"));              Iterable<String> actualList = new ArrayList<> (Arrays.asList("First", "Two", "Third"));              assertAll (              "GroupedAssertionHeading",              () -> assertEquals (a, b, delta, "assert with delta"),              () -> assertIterableEquals (expectedList, actualList)       );         }  } |

**Result:**

As one of the assert in the group fails, instead of **AssertionFailureError** it results in **MultipleFailuresError**thereby displaying the heading of the grouped assertion passed as the input parameter i.e. GroupedAssertionHeading in this example.

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2021/03/MultipleFailuresError.png)

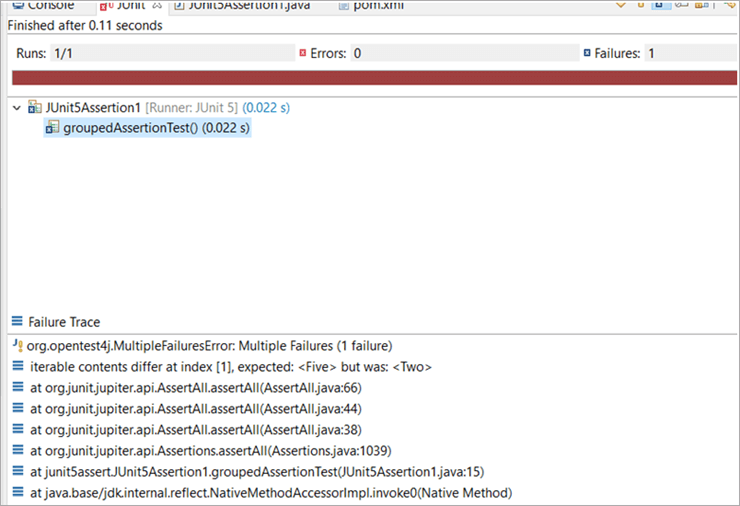
**Grouped Assertions Without Heading As Parameter**

The assertAll () can be implemented without using the heading parameter. The below example shows an implementation of grouped assertion using no heading parameter.

|  |
| --- |
| class JUnit5Assertion1 {       @Test       public void groupedAssertionTest () {                         float a=(float) 1.2;              float b=(float) 1.2;              float delta=(float) 1.0;              Iterable<String> expectedList = new ArrayList<> (Arrays.asList("In", "Five", "Lane"));              Iterable<String> actualList = new ArrayList<> (Arrays.asList("In", "Two", "Lane"));              assertAll (              () -> assertEquals (a, b, delta, "assert with delta"),              () -> assertIterableEquals (expectedList, actualList)       );         }  } |

**Result:**

The result displays without heading.

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2021/03/result-displays-without-heading.png)

**Nested Or Dependent Grouped Assertions**

In the above pointers 2 and 3, we saw the implementation of group assertions. We were able to use multiple assert methods under one assertAll () function.

Likewise, there is a variation to its implementation, too.

**Grouping of grouped assertions** is also a possible feature under JUnit 5 which is alternatively called “**Nested Grouped Assertions**” or “**Dependent Grouped Assertions**”.

* To clarify this in a layman language, one assertAll() is considered as one independent grouped assertion.
* When one assertAll() includes one or more assertAll() then these are referred to as a nested grouped assertions.

Examples For Implementation Of Nested Assertions

Let us practically understand, how this is implemented:

**Example 1:**

**Here, we are using two program files:**

1. Employee.java file – A java class file with getter and setter functions.
2. JUnit5Assertion1.java – A JUnit 5 class file that calls the constructor for the Employee class file and also implements the nested grouped assertion.
3. The first assert to validate that the employee name fetched is not null and the internal or nested assert is to validate that the employee code is as expected code. In this example, both the asserts result in a pass hence the final test status passes.

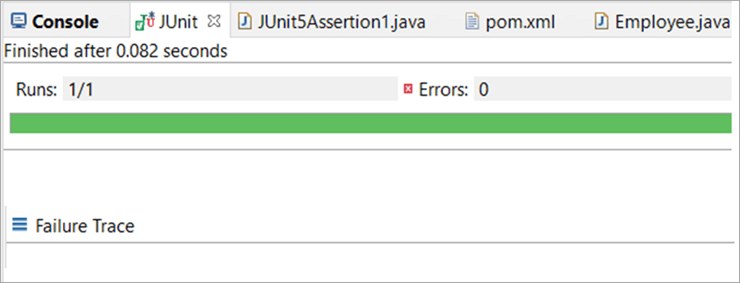
**Code For Employee.java**

|  |
| --- |
| package junit5assert;  public class Employee {      public String ename;       public int empcode;         public Employee (String ename, int empcode) {        this.ename = ename;        this.empcode = empcode;       }         public String getEname() {        return ename;       }         public void setEname(String ename) {        this.ename = ename;       }         public int getEcode() {        return empcode;       }         public void setEcode(int empcode) {        this.empcode = empcode;       }  } |

**Code for JUnit5Assertion1.java**

|  |
| --- |
| package junit5assert;  import static org.junit.jupiter.api.Assertions.\*;  import org.junit.jupiter.api.Test;    class JUnit5Assertion1 {      String ename;      int ecode;      String name=null;       @Test       public void dependentGrpAssert() {            Employee emp=new Employee("Nidhi Singh",1028838);            assertAll("ValidateEmpNameNotNull",                  () -> {                      ename=emp.getEname();                      assertNotNull(ename);                        assertAll("ValidateEmpCode",                              () -> {                                  ecode=emp.getEcode();                                  assertEquals (1028838,1028838);                              }                        ); //end of inner assertAll ();                  }            ) ;//end of outer assertAll ();      }  } |

**Result:**

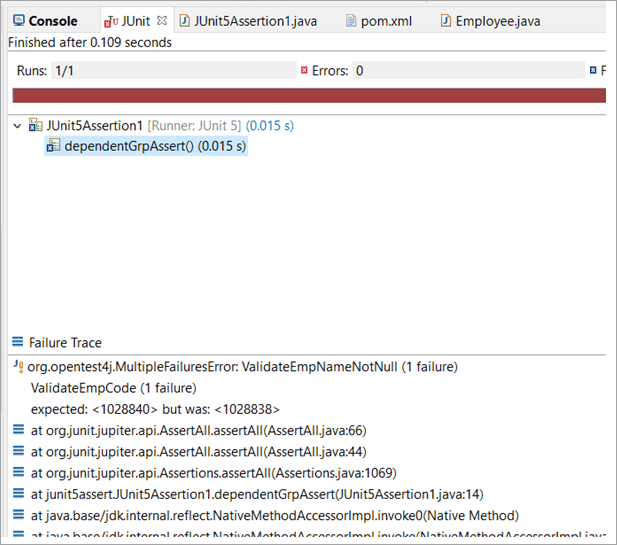
[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2021/03/Code-for-JUnit5Assertion1.java_-1.png)

**Example 2:**

In this example, the internal assertion for emp code fails. We see that the outer assert will execute and pass however the internal assert will fail and result in ‘MultipleFailuresError” by marking both the assertions as failed and keeping the number of failures as 1.

|  |
| --- |
| import static org.junit.jupiter.api.Assertions.\*;  import org.junit.jupiter.api.Test;    class JUnit5Assertion1 {      String ename;      int ecode;      String name=null;       @Test       public void dependentGrpAssert() {            Employee emp=new Employee("Nidhi Singh",1028838);            assertAll("ValidateEmpNameNotNull",                  () -> {                      ename=emp.getEname();                      assertNotNull(ename);                        assertAll("ValidateEmpCode",                              () -> {                                  ecode=emp.getEcode();                                  assertEquals(1028840,1028838);                              }                      );                  }            );      }  } |

**Result:**

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2021/03/internal-assertion-for-emp-code.png)

**Example 3:**

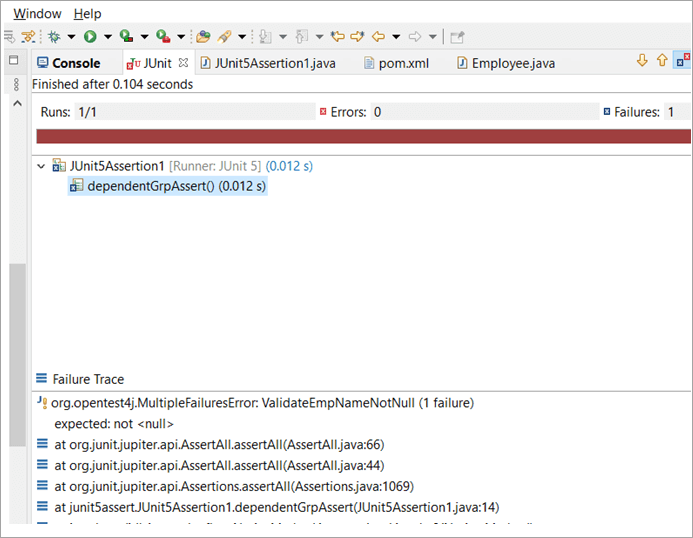
In this example, the external assertion for employee code should fail as the employee’s name returns null. Despite that, the inner assert should pass as the actual employee code matches the expected code, and the internal assert is never executed.

So**, the inference is that if the external assert fails, it skips all its dependent or internally grouped assertions during execution.**

|  |
| --- |
| import org.junit.jupiter.api.Test;    class JUnit5Assertion1 {      String ename;      int ecode;      String name=null;       @Test       public void dependentGrpAssert() {          Employee emp=new Employee("Nidhi Singh",1028838);          assertAll("ValidateEmpNameNotNull",                  () -> {                      assertNotNull(name);                        assertAll("ValidateEmpCode",                              () -> {                                  ecode=emp.getEcode();                                  assertEquals(1028838,1028838);                              }                        ); //end of inner assertAll();                  }            );//end of outer assertAll();      }  } |

**Result:**

As per the result, the outer grouped assertion with the heading ValidateEmpNameNotNull fails and skips the execution of the inner grouped assertion with the heading ValidateEmpCode skips for execution. This is why 1 failure is reported for external assertion.

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2021/03/ValidateEmpNameNotNull-1.png)

When Do We Use Nested Grouped Assertions?

For independent test cases, the usual independent assertion could be used.

**Having independent assertion in place will have a couple of the following benefits:**

1. It will make sure that each of the assert functions is tested separately and independently.
2. In this case, if one assert function fails, the system will execute the next assertion and so on.
3. The system shall run all the assertions one by one.
4. So, basically, this approach would succeed where unit testing would work.

We could now be curious about how do we decide when to use nested grouped assertions.

**We could use nested grouped assertions in the below scenarios:**

1. When one or more test cases are dependent on the other.
2. OR when the assertion result of one assert statement is the deciding factor for another assert statement in the code.

**With grouped assertions, given below is the code flow in brief:**

1. If the first assert method passes then the next assert function is executed until the last assert function in the external or parent group.
2. If any of the assets from the external group fails, it skips the run for internal or nested grouped assertions.
3. else If the external group collectively passes, then the internal group executes by picking each assert one by one.