Regression Testing

To implement unit testing in Java,
and accelerate programming speed
and increase the quality of code.

Features of JUnit Test Framework

- Fixtures
- Test suites
- Test runners
- JUnit classes

Fixtures

A fixed state of a set of objects used as a baseline for running tests to ensure that there is a well-known and fixed environment in which tests will execute.

- **setUp() method**, which runs before every test invocation.
- **tearDown() method**, which runs after every test method.

Fixture Example

```
import junit.framework.*;

public class JavaTest extends TestCase {
   protected int value1, value2;

   // assigning the values
   protected void setUp(){
      value1 = 3;
      value2 = 3;
   }

   // test method to add two values
   public void testAdd(){
      double result = value1 + value2;
      assertTrue(result == 6);
   }
}
```

Test Suites

A bundles of a few unit test cases and runs them together.

@RunWith and **@Suite** annotation are used to run the suite test.

Test Suites

```
import org.junit.runner.RunWith;
import org.junit.runners.Suite;

//JUnit Suite Test
@RunWith(Suite.class)

@Suite.SuiteClasses({
   TestJunit1.class ,TestJunit2.class
})

public class JunitTestSuite {
}
```

```
import org.junit.*;
import static org.junit.Assert.assertEquals;

public class TestJunit1 {

   String message = "Robert";

   @Test
   public void testPrintMessage() {
      System.out.println("Inside
testPrintMessage");
      assertEquals(message, “Robert”);
   }
}
```

```
import org.junit.*;
import static org.junit.Assert.assertEquals;

public class TestJunit2 {

   String message = "Robert";

   @Test
   public void testSalutationMessage() {
      System.out.println("Inside
testSalutationMessage ");
      message = "Hi!" + "Robert";
      assertEquals(message,”Hi! Robert”);
   }
}
```

Test Runners

used for executing the test cases.

```
import org.junit.runner.JUnitCore;
import org.junit.runner.Result;
import org.junit.runner.notification.Failure;

public class TestRunner {
   public static void main(String[] args) {
      Result result = JUnitCore.runClasses(TestJunit.class);

      for (Failure failure : result.getFailures()) {
         System.out.println(failure.toString());
      }

      System.out.println(result.wasSuccessful());
   }
}
```

JUnit Classes

used in writing and testing JUnits. Some of the important classes are −

Assert − Contains a set of assert methods.

TestCase − Contains a test case that defines the fixture to run multiple tests.

TestResult − Contains methods to collect the results of executing a test case.

Assert Class

public class Assert extends java.lang.Object

This class provides a set of assertion methods useful for writing tests. Only failed assertions are recorded.

TestJunit1.java

```
import org.junit.Test;
import static org.junit.Assert.*;
public class TestJunit1 {
   @Test
   public void testAdd() {
      //test data
      int num = 5;
      String temp = null;
      String str = "Junit is working fine";
      //check for equality
      assertEquals("Junit is working fine", str);
      //check for false condition
      assertFalse(num > 6);
      //check for not null value
      assertNotNull(temp);
   }
}
```

TestRunner1.java

```
import org.junit.runner.JUnitCore;
import org.junit.runner.Result;
import org.junit.runner.notification.Failure;
public class TestRunner1 {
   public static void main(String[] args) {
      Result result = JUnitCore.runClasses(TestJunit1.class);
      for (Failure failure : result.getFailures()) {
         System.out.println(failure.toString());
      }
      System.out.println(result.wasSuccessful());
   }
}
```

```
C:\JUNIT_WORKSPACE>javac TestJunit1.java TestRunner1.java
C:\JUNIT_WORKSPACE>java TestRunner1
```

TestCase Class

```
public abstract class TestCase extends Assert implements Test
A test case defines the fixture to run multiple tests
public class TestJunit2 extends TestCase  {
   protected double fValue1;
   protected double fValue2;
   @Before
   public void setUp() {
      fValue1 = 2.0;
      fValue2 = 3.0;
   }
   @Test
   public void testAdd() {
      //count the number of test cases
      System.out.println("No of Test Case = "+ this.countTestCases());
      //test getName
      String name = this.getName();
      System.out.println("Test Case Name = "+ name);
      //test setName
      this.setName("testNewAdd");
      String newName = this.getName();
      System.out.println("Updated Test Case Name = "+ newName);
   }
   //tearDown used to close the connection or clean up activities
@After
   public void tearDown(  ) {
fValue1 = 0;
      fValue2 = 0;
   }
}
```

TestRunner1.java

```
import org.junit.runner.JUnitCore;
import org.junit.runner.Result;
import org.junit.runner.notification.Failure;
public class TestRunner2 {
   public static void main(String[] args) {
      Result result = JUnitCore.runClasses(TestJunit2.class);
      for (Failure failure : result.getFailures()) {
         System.out.println(failure.toString());
      }
      System.out.println(result.wasSuccessful());
   }
}
```

```
C:\JUNIT_WORKSPACE>javac TestJunit2.java TestRunner2.java
C:\JUNIT_WORKSPACE>java TestRunner2
```

TestResult Class

public class TestResult extends Object

- A TestResult collects the results of executing a test case.
- It is an instance of the Collecting Parameter pattern.
- The test framework distinguishes between failures and errors.
- A failure is anticipated and checked for with assertions.
- Errors are unanticipated problems like an ArrayIndexOutOfBoundsException

TestResult Class Example

```
public class TestJunit3 extends TestResult {
   // add the error
   public synchronized void addError(Test test, Throwable t) {
      super.addError((junit.framework.Test) test, t);
   }
   // add the failure
   public synchronized void addFailure(Test test, AssertionFailedError t) {
      super.addFailure((junit.framework.Test) test, t);
   }
   @Test
   public void testAdd() {
      // add any test
   }
   // Marks that the test run should stop.
   public synchronized void stop() {
      //stop the test here
   }
}
```

TestResult Class Runner

```
import org.junit.runner.JUnitCore;
import org.junit.runner.Result;
import org.junit.runner.notification.Failure;
public class TestRunner3 {
   public static void main(String[] args) {
      Result result = JUnitCore.runClasses(TestJunit3.class);
      for (Failure failure : result.getFailures()) {
         System.out.println(failure.toString());
      }
      System.out.println(result.wasSuccessful());
   }
}
```

```
C:\JUNIT_WORKSPACE>javac TestJunit3.java TestRunner3.java
C:\JUNIT_WORKSPACE>java TestRunner3
```

TestSuite Class

public class TestSuite extends Object implements Test

A TestSuite is a Composite of tests. It runs a collection of test cases.

```
import junit.framework.*;
public class JunitTestSuite {
   public static void main(String[] a) {
      // add the test's in the suite
      TestSuite suite = new TestSuite(TestJunit1.class, TestJunit2.class,
TestJunit3.class );
      TestResult result = new TestResult();
      suite.run(result);
      System.out.println("Number of test cases = " + result.runCount());
   }
}
```

```
C:\JUNIT_WORKSPACE>javac JunitTestSuite.java
C:\JUNIT_WORKSPACE>java JunitTestSuite
```

Create a Class

```
public class MessageUtil {
   private String message;
   //Constructor
   //@param message to be printed
   public MessageUtil(String message){
      this.message = message;
   }
   // prints the message
   public String printMessage(){
      System.out.println(message);
      return message;
   }
}
```

Create Test Case Class

```
import org.junit.Test;
import static org.junit.Assert.assertEquals;
public class TestJunit {
   String message = "Hello World";
   MessageUtil messageUtil = new MessageUtil(message);
   @Test
   public void testPrintMessage() {
      assertEquals(message,messageUtil.printMessage());
   }
}
```

```
C:\JUNIT_WORKSPACE>javac MessageUtil.java TestJunit.java TestRunner.java
C:\JUNIT_WORKSPACE>java TestRunner
```

TestJunit.java

```
import org.junit.Test;
import static org.junit.Assert.assertEquals;
public class TestJunit {
   String message = "Hello World";
   MessageUtil messageUtil = new MessageUtil(message);
   @Test
   public void testPrintMessage() {
      message = "New Word";
      assertEquals(message,messageUtil.printMessage());
   }
}
```

TestRunner.java

```
import org.junit.runner.JUnitCore;
import org.junit.runner.Result;
import org.junit.runner.notification.Failure;
public class TestRunner {
   public static void main(String[] args) {
      Result result = JUnitCore.runClasses(TestJunit.class);
      for (Failure failure : result.getFailures()) {
         System.out.println(failure.toString());
      }
      System.out.println(result.wasSuccessful());
   }
}
```

Writing Test Cases

```
public class EmployeeDetails {
   private String name;
   private double monthlySalary;
   private int age;
//Getter Setter and Constructor
}
public class EmpBusinessLogic {
   // Calculate the yearly salary of employee
   public double calculateYearlySalary(EmployeeDetails employeeDetails) {
      double yearlySalary = 0;
      yearlySalary = employeeDetails.getMonthlySalary() * 12;
      return yearlySalary;
   }
   // Calculate the appraisal amount of employee
   public double calculateAppraisal(EmployeeDetails employeeDetails) {
      double appraisal = 0;
      if(employeeDetails.getMonthlySalary() < 10000){
         appraisal = 500;
      }else{
         appraisal = 1000;
      }
      return appraisal;
   }
}
```

TestEmployeeDetails.java

```
import org.junit.Test;
import static org.junit.Assert.assertEquals;
public class TestEmployeeDetails {
   EmpBusinessLogic empBusinessLogic = new EmpBusinessLogic();
   EmployeeDetails employee = new EmployeeDetails();
   //test to check appraisal
   @Test
   public void testCalculateAppriasal() {
      employee.setName("Rajeev");
      employee.setAge(25);
      employee.setMonthlySalary(8000);
      double appraisal = empBusinessLogic.calculateAppraisal(employee);
      assertEquals(500, appraisal, 0.0);
   }
   // test to check yearly salary
   @Test
   public void testCalculateYearlySalary() {
      employee.setName("Rajeev");
      employee.setAge(25);
      employee.setMonthlySalary(8000);
      double salary = empBusinessLogic.calculateYearlySalary(employee);
      assertEquals(96000, salary, 0.0);
   }
}
```

TestRunner.java

```
import org.junit.runner.JUnitCore;
import org.junit.runner.Result;
import org.junit.runner.notification.Failure;
public class TestRunner {
   public static void main(String[] args) {
      Result result = JUnitCore.runClasses(TestEmployeeDetails.class);
      for (Failure failure : result.getFailures()) {
         System.out.println(failure.toString());
      }
      System.out.println(result.wasSuccessful());
   }
}
```

```
C:\JUNIT_WORKSPACE>javac EmployeeDetails.java EmpBusinessLogic.java
TestEmployeeDetails.java TestRunner.java
```

```
C:\JUNIT_WORKSPACE>java TestRunner
```

Assertion

public class Assert extends java.lang.Object

- This class provides a set of assertion methods, useful for writing tests.
- Only failed assertions are recorded.

Assertion

```
public class TestAssertions {
   @Test
   public void testAssertions() {
      //test data
      String str1 = new String ("abc");
      String str2 = new String ("abc");
      String str3 = null;
      String str4 = "abc";
      String str5 = "abc";
      int val1 = 5;
      int val2 = 6;
      String[] expectedArray = {"one", "two", "three"};
      String[] resultArray =  {"one", "two", "three"};
      //Check that two objects are equal
      assertEquals(str1, str2);
      //Check that a condition is true
      assertTrue (val1 < val2);
      //Check that a condition is false
      assertFalse(val1 > val2);
      //Check that an object isn't null
      assertNotNull(str1);
      //Check that an object is null
      assertNull(str3);
      //Check if two object references point to the same object
      assertSame(str4,str5);
      //Check if two object references not point to the same object
      assertNotSame(str1,str3);
      //Check whether two arrays are equal to each other.
      assertArrayEquals(expectedArray, resultArray);
   }
}
```

Assertion Runner

```
import org.junit.runner.JUnitCore;
import org.junit.runner.Result;
import org.junit.runner.notification.Failure;
public class TestRunner2 {
   public static void main(String[] args) {
      Result result = JUnitCore.runClasses(TestAssertions.class);
      for (Failure failure : result.getFailures()) {
         System.out.println(failure.toString());
      }
      System.out.println(result.wasSuccessful());
   }
}
```

```
C:\JUNIT_WORKSPACE>javac TestAssertions.java TestRunner.java
C:\JUNIT_WORKSPACE>java TestRunner
```

JUnitCore class

- JUnitCore is a facade for running tests.
- supports running JUnit 4 tests, JUnit 3.8.x tests, and mixtures.
- To run tests from the command line, run java org.junit.runner.JUnitCore <TestClass>.
- For one-shot test runs, use the static method runClasses(Class[]).

public class JUnitCore extends java.lang.Object

MessageUtil.java

```
public class MessageUtil {
   private String message;
   //Constructor
   //@param message to be printed
   public MessageUtil(String message){
      this.message = message;
   }
   // prints the message
   public String printMessage(){
      System.out.println(message);
      return message;
   }
}
```

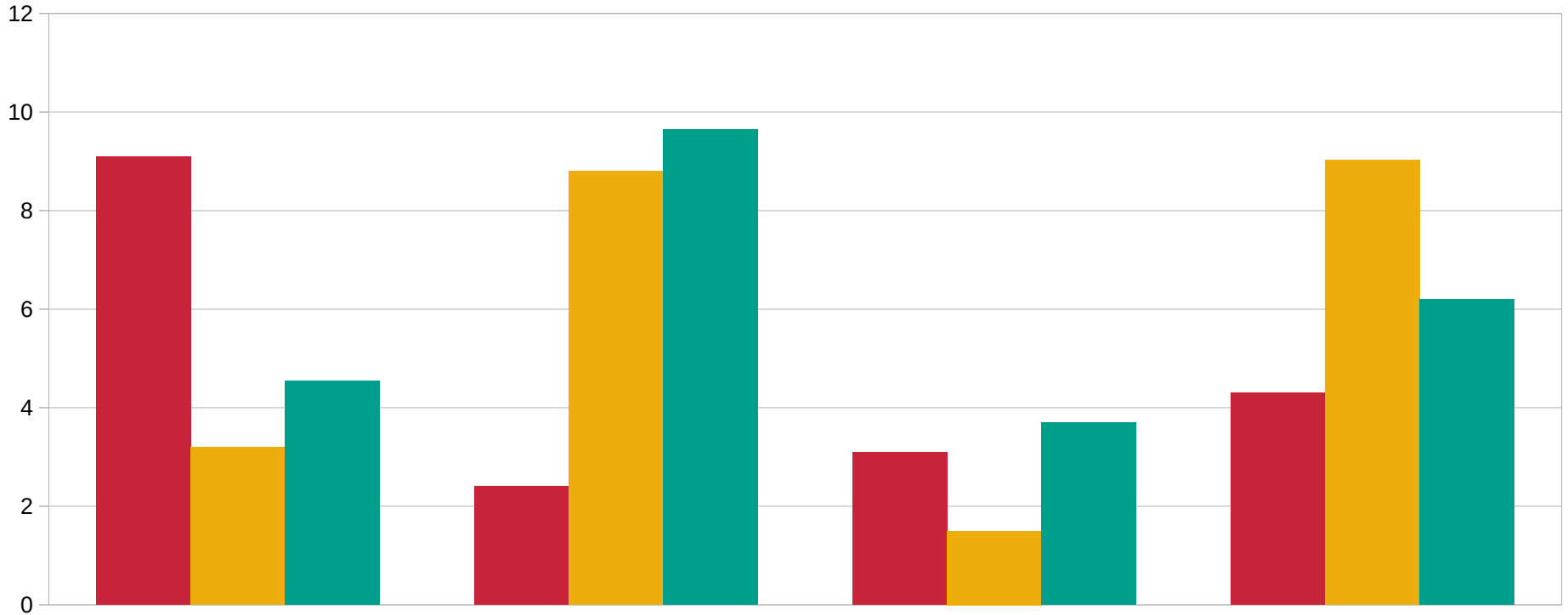
TestJunit.java

```
import org.junit.Test;
import static org.junit.Assert.assertEquals;
public class TestJunit {
   String message = "Hello World";
   MessageUtil messageUtil = new MessageUtil(message);
   @Test
   public void testPrintMessage() {
      assertEquals(message,messageUtil.printMessage());
   }
}
```

TestRunner.java

```
import org.junit.runner.JUnitCore;
import org.junit.runner.Result;
import org.junit.runner.notification.Failure;
public class TestRunner {
   public static void main(String[] args) {
      Result result = JUnitCore.runClasses(TestJunit.class);
      for (Failure failure : result.getFailures()) {
         System.out.println(failure.toString());
      }
      System.out.println(result.wasSuccessful());
   }
}
```

```
C:\JUNIT_WORKSPACE>javac MessageUtil.java TestJunit.java TestRunner.java
C:\JUNIT_WORKSPACE>java TestRunner
```



12
10
8
6
4
2
0