JASON JOSE P

20EUCS053

EX NO:6

DATE:28.09.2021

UNIT TESTING

AIM:

To write unit tests aligned to xUnit framework for TDD.

THEORY:

1. Unit Testing

Unit testing is a software testing method by which individual units of source code—sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures—are tested to determine whether they are fit for use.

Procedure:

1. Open a Java IDE, and create a new package.
2. Create a class and write a code for the given problem.
3. Create a new JUnit Test Case and add it to the build path.
4. Create appropriate objects and call the required functions.
5. Pass the test case as a parameter in the built in assertEquals function .
6. Save the program and run it. By the Left hand side you can see the test case results (thecount of Runs or Errors or Failures).

PROGRAMS

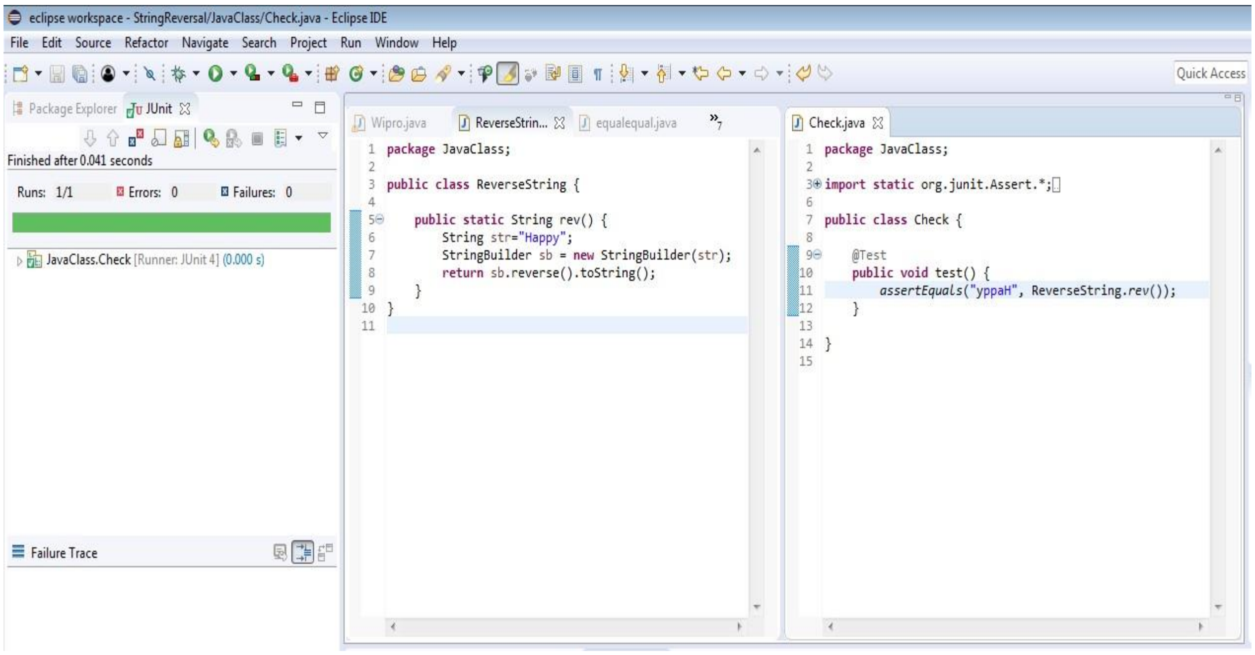
Reverse String :

```
Package javaclass;

Public class Reversestring{
      String str=”Happy”;

      StringBuilder sb=new StringBuilder(str);
      return  sb.reverse().toString();

}}
```

TEST CASE PROGRAM:

```
  Package javaclass;

  Import static org.Junit.Assert.* ;
Public class check{

Public void test(){
assertEquals(“yppaH”,reversestring.rev());}}
```

RUNS:



eclipse workspace - StringReversal/JavaClass/Check.java - Eclipse IDE

Finished after 0.041 seconds

Runs: 1/1 Errors: 0 Failures: 0

JavaClass.Check [Runner: JUnit 4] (0.000 s)

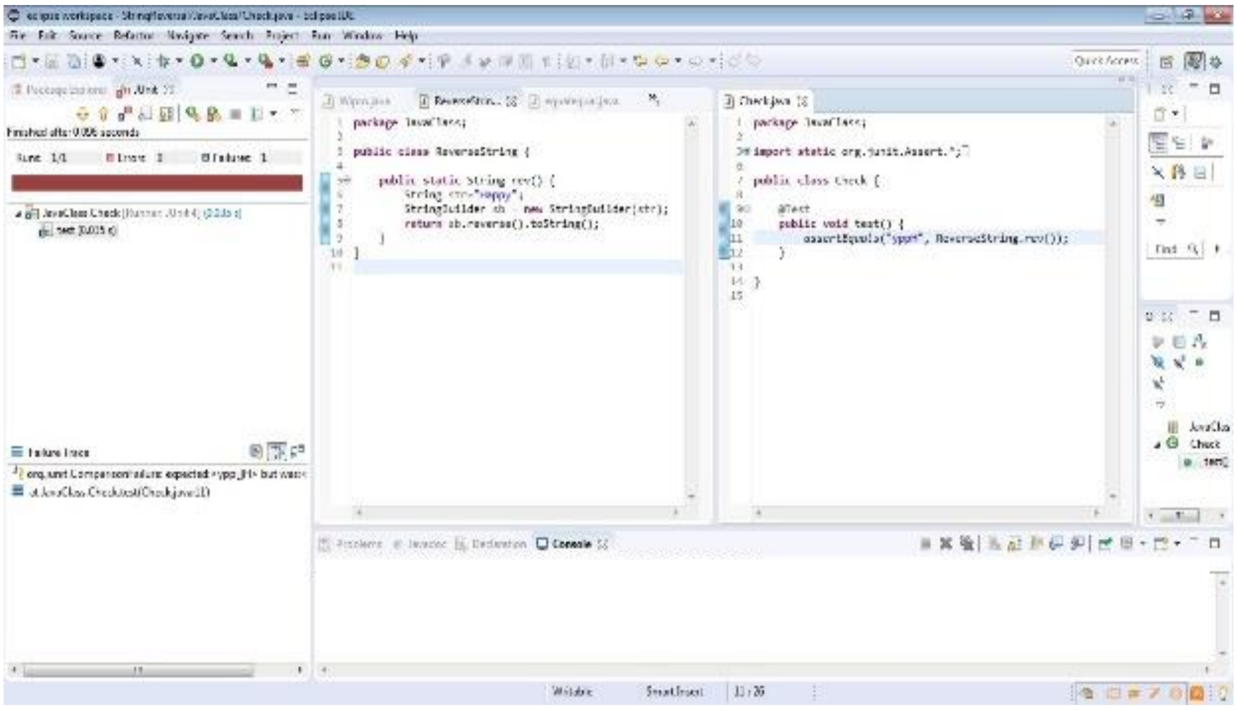
ReverseString.java:

```java
package JavaClass;

public class ReverseString {

    public static String rev() {
        String str="Happy";
        StringBuilder sb = new StringBuilder(str);
        return sb.reverse().toString();
    }
}
```

Check.java:

```java
package JavaClass;

import static org.junit.Assert.*;

public class Check {

    @Test
    public void test() {
        assertEquals("yppaH", ReverseString.rev());
    }

}
```

FAILS:



eclipse workspace - StringReversal/JavaClass/Check.java - Eclipse IDE

Finished after 0.006 seconds

Runs: 1/1 Errors: 0 Failures: 1

JavaClass.Check [Runner: JUnit 4] (0.015 s)
test (0.015 s)

Failure Trace

```
org.junit.ComparisonFailure: expected:<ypp[H]> but was:<
at JavaClass.Check.test(Check.java:11)
```

ReverseString.java:

```java
package JavaClass;

public class ReverseString {

    public static String rev() {
        String str="Happy";
        StringBuilder sb = new StringBuilder(str);
        return sb.reverse().toString();
    }
}
```

Check.java:

```java
package JavaClass;

import static org.junit.Assert.*;

public class Check {

    @Test
    public void test() {
        assertEquals("yppH", ReverseString.rev());
    }

}
```

RESULT:

Unit tests aligned to xUnit framework for TDD were written and tested successfully.

JASON JOSE P

20EUCS053

EX NO:7

DATE:6.10.2021

REFACTORING

AIM:

To Refactor a given design for next sprint requirements..

THEORY:

1. **Refactoring:**

Refactoring is the process of clarifying and simplifying the design of existing code, without changing its behavior. Agile teams are maintaining andextending their code a lot from iteration to iteration, and without
continuous refactoring, this is hard to do.

Procedure:

1. Open a Java IDE, and create a new package.
2. Create a class and write a code for the given problem.
3. Create a new JUnit Test Case and add it to the build path.
4. Create appropriate objects and call the required functions.
5. Pass the test case as a parameter in the built in assertEquals function .
6. Save the program and run it. By the Left hand side you can see the test case results (thecount of Runs or Errors or Failures).

PROGRAMS

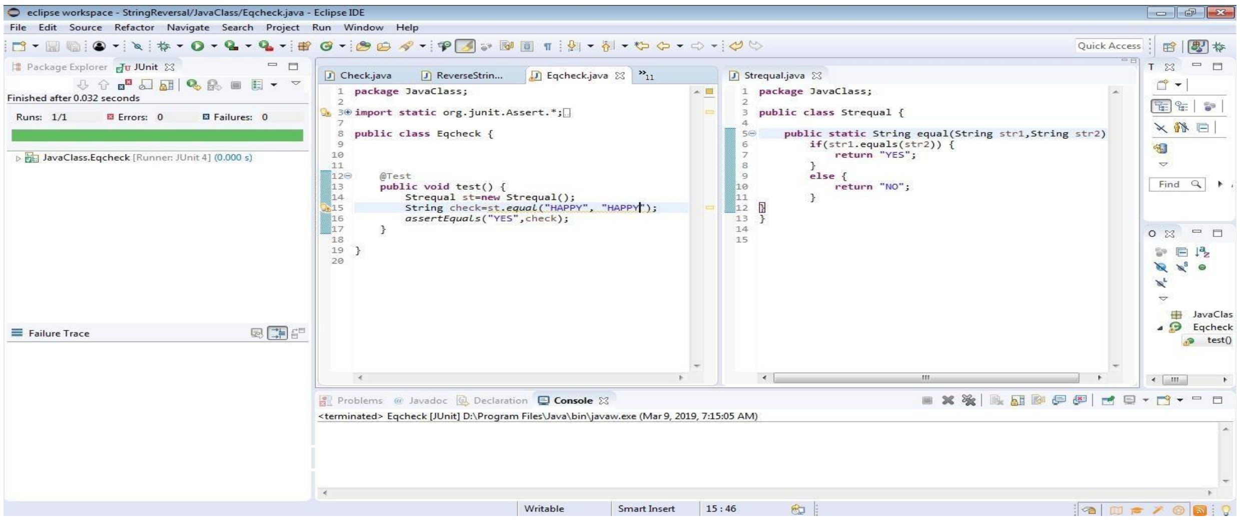
STRING EQUIVALENCE:

```
Package javaclass;

Public class strequals{

Public static String equal(String str1,String str2){
  if(str1.equals(str2)){

   return “yes”;

    }

  else{

   return “no”;

  }

  }
```

TEST CASE PROGRAM:

```
  Package javaclass;

  Import static org.Junit.Assert.* ;
Public class check{

Public void test(){

Stringequal s=new strequal();

String b=s.b.equal(“Happy”, “Malayalam”);

assertEquals(“yes”,b);}}
```

RUNS:



eclipse workspace - StringReversal/JavaClass/Eqcheck.java - Eclipse IDE

```
Finished after 0.032 seconds
Runs: 1/1    Errors: 0    Failures: 0
JavaClass.Eqcheck [Runner: JUnit 4] (0.000 s)
```

Eqcheck.java

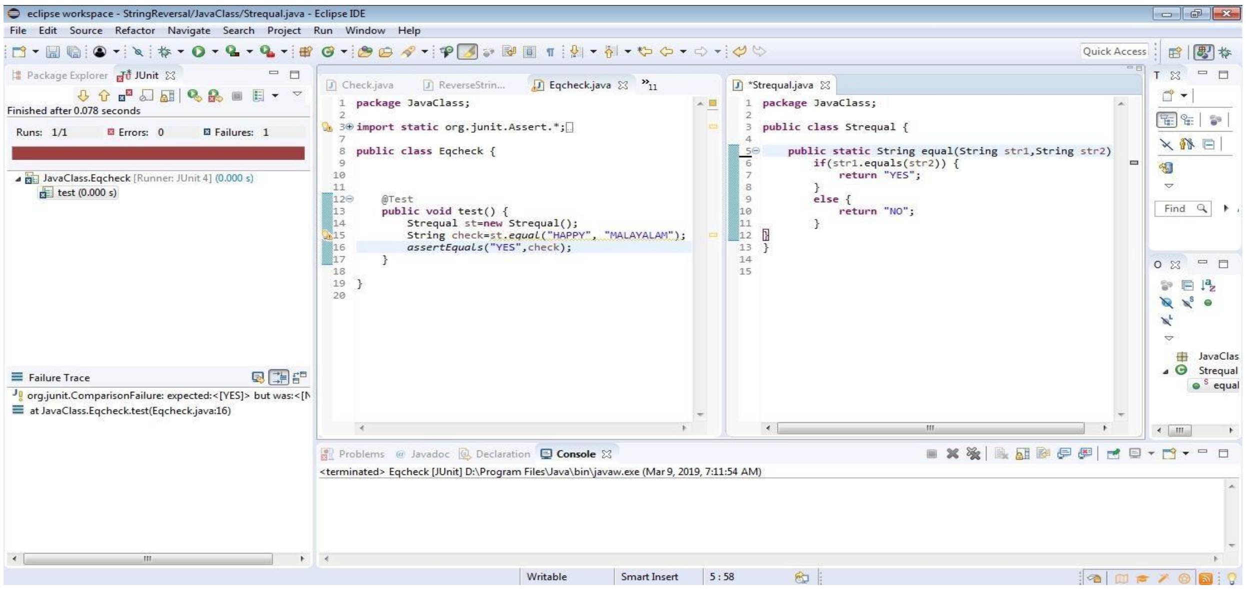
```
package JavaClass;

import static org.junit.Assert.*;

public class Eqcheck {



    @Test
    public void test() {
        Strequal st=new Strequal();
        String check=st.equal("HAPPY", "HAPPY");
        assertEquals("YES",check);
    }

}
```

Strequal.java

```
package JavaClass;

public class Strequal {

    public static String equal(String str1,String str2)
        if(str1.equals(str2)) {
            return "YES";
        }
        else {
            return "NO";
        }

}
```

```
<terminated> Eqcheck [JUnit] D:\Program Files\Java\bin\javaw.exe (Mar 9, 2019, 7:15:05 AM)
```

FAILS:



eclipse workspace - StringReversal/JavaClass/Strequal.java - Eclipse IDE

```
Finished after 0.078 seconds
Runs: 1/1    Errors: 0    Failures: 1
JavaClass.Eqcheck [Runner: JUnit 4] (0.000 s)
    test (0.000 s)
```

Failure Trace

```
org.junit.ComparisonFailure: expected:<[YES]> but was:<[N
at JavaClass.Eqcheck.test(Eqcheck.java:16)
```

Eqcheck.java

```
package JavaClass;

import static org.junit.Assert.*;

public class Eqcheck {



    @Test
    public void test() {
        Strequal st=new Strequal();
        String check=st.equal("HAPPY", "MALAYALAM");
        assertEquals("YES",check);
    }

}
```

*Strequal.java

```
package JavaClass;

public class Strequal {

    public static String equal(String str1,String str2)
        if(str1.equals(str2)) {
            return "YES";
        }
        else {
            return "NO";
        }

}
```

```
<terminated> Eqcheck [JUnit] D:\Program Files\Java\bin\javaw.exe (Mar 9, 2019, 7:11:54 AM)
```

RESULT: A given design for next sprint requirements was refactoring.

JASON JOSE P

20EUCS053

EX NO:8 (i)

DATE:27.10.2021

CONTINOUS INTEGRATION- MANUAL TOOLS

AIM:

To execute continuous integration using a Manual tool such as Sonarqube, Cobertura, Maven, Artifactory and Tomcat.

THEORY:

Continuous Integration:

Continuous integration (CI) is the practice of automating the integration of code changes from multiple contributors into a single software project. The CI process is comprised of automatic tools that assert the new code’s correctness before integration. A source code version control system is the crux of the CI process.

Procedure:

1. Open a Java IDE, and create a new package.
2. Create a class and write a code for the given problem.
3. Create a new JUnit Test Case and add it to the build path.
4. Create appropriate objects and call the required functions.
5. Pass the test case as a parameter in the built in assertEquals function .
6. Save the program and run it. By the Left hand side you can see the test case results (the count of Runs or Errors or Failures).

PROGRAMS:

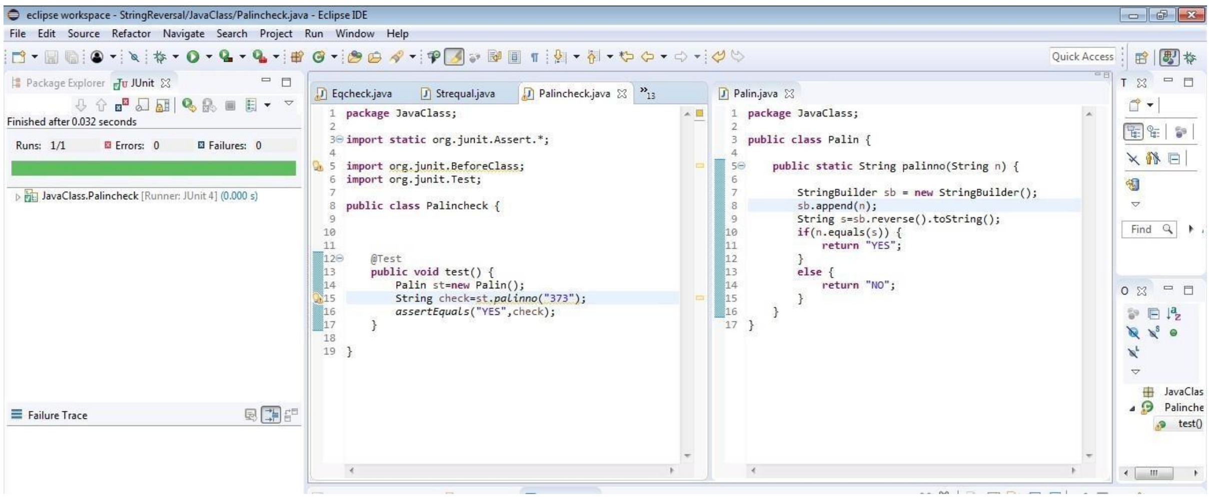
PALINDROME NUMBER :

```
Package javaclass;
Public class palin{

Public static String palindrome (String s){
StringBuilder sb=new StringBuilder();
Sb.append(s);

String s=sb.reverse().toString();
 if(n.equals(s)){

 return “yes”;

}

 else{ return
 “no;

}
```

TEST CASE PROGRAM:

```
Package javaclass;

import static org.Junit.Assert.* ;
Public class check{

     Public void test(){

     palin s=new palin();

    String b=s.b.palinno(“555”);
    assertEquals(“yes”,b);}}
```

RUNS:



eclipse workspace - StringReversal/JavaClass/Palincheck.java - Eclipse IDE

Finished after 0.032 seconds

Runs: 1/1   Errors: 0   Failures: 0

JavaClass.Palincheck [Runner: JUnit 4] (0.000 s)

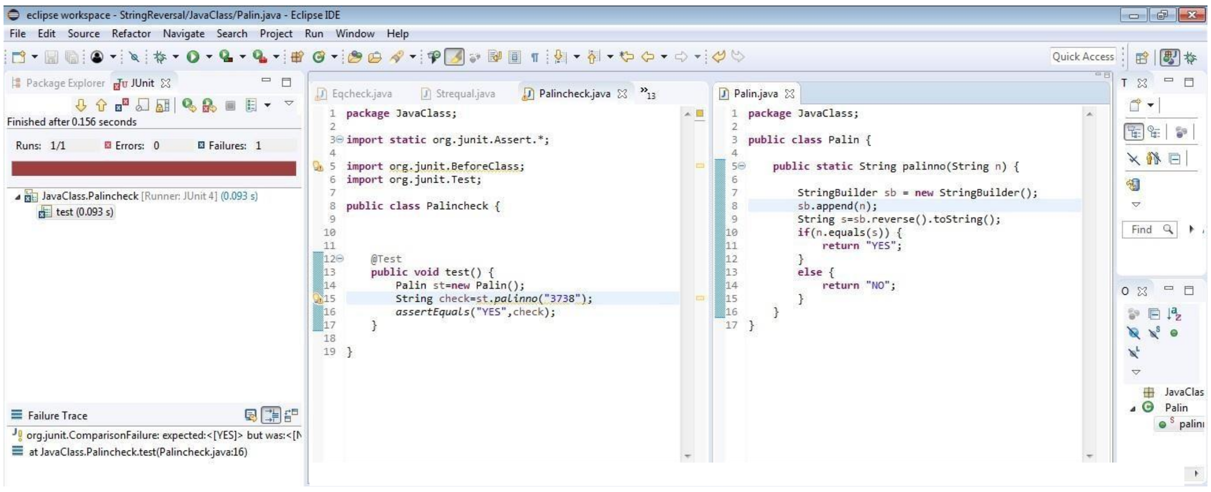
Palincheck.java

```java
package JavaClass;

import static org.junit.Assert.*;

import org.junit.BeforeClass;
import org.junit.Test;

public class Palincheck {


    @Test
    public void test() {
        Palin st=new Palin();
        String check=st.palinno("373");
        assertEquals("YES",check);
    }

}
```

Palin.java

```java
package JavaClass;

public class Palin {

    public static String palinno(String n) {

        StringBuilder sb = new StringBuilder();
        sb.append(n);
        String s=sb.reverse().toString();
        if(n.equals(s)) {
            return "YES";
        }
        else {
            return "NO";
        }
    }
}
```

FAILS:



eclipse workspace - StringReversal/JavaClass/Palin.java - Eclipse IDE

Finished after 0.156 seconds

Runs: 1/1   Errors: 0   Failures: 1

JavaClass.Palincheck [Runner: JUnit 4] (0.093 s)
test (0.093 s)

Failure Trace
```
org.junit.ComparisonFailure: expected:<[YES]> but was:<[N
at JavaClass.Palincheck.test(Palincheck.java:16)
```

Palincheck.java

```java
package JavaClass;

import static org.junit.Assert.*;

import org.junit.BeforeClass;
import org.junit.Test;

public class Palincheck {


    @Test
    public void test() {
        Palin st=new Palin();
        String check=st.palinno("3738");
        assertEquals("YES",check);
    }

}
```

Palin.java

```java
package JavaClass;

public class Palin {

    public static String palinno(String n) {

        StringBuilder sb = new StringBuilder();
        sb.append(n);
        String s=sb.reverse().toString();
        if(n.equals(s)) {
            return "YES";
        }
        else {
            return "NO";
        }
    }
}
```

RESULT:

continuous integration using a Manual tool such as SonarQube, Coverture, Maven, Artifactory and Tomcat were executed and tested successfully.

JASON JOSE P

20EUCS053

EX NO:8 (ii)

DATE:2.11.2021

CONTINOUS INTEGRATION- AUTOMATED TOOL

AIM:

To execute continuous integration using a automated tool such as Jenkins.

THEORY:

Continuous Integration:

Continuous integration (CI) is the practice of automating the integration of code changes from multiple contributors into a single software project. The CI process is comprised of automatic tools that assert the new code's correctness before integration. A source code version control system is the crux of the CI process.

Procedure:

1. Open a Java IDE, and create a new package.
2. Create a class and write a code for the given problem.
3. Create a new JUnit Test Case and add it to the build path.
4. Create appropriate objects and call the required functions.
5. Pass the test case as a parameter in the built in assertEquals function .
6. Save the program and run it. By the Left hand side you can see the test case results (the count of Runs or Errors or Failures).

PROGRAMS

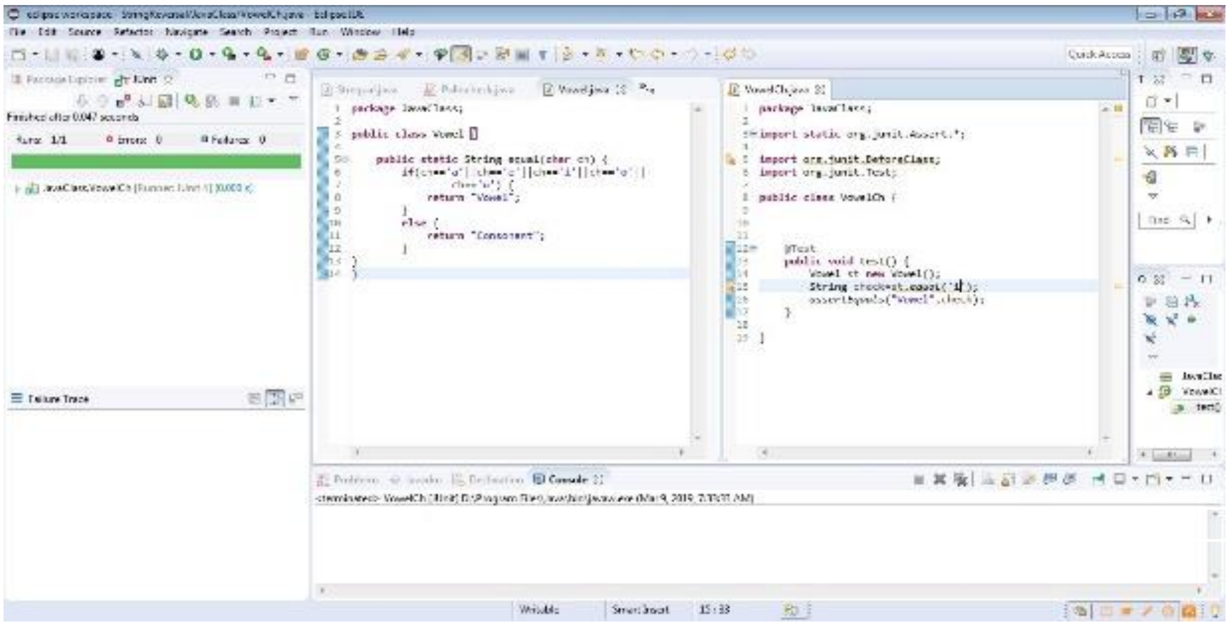
ALPHABET CHECK:

```
Package javaclass;
Public class vowel{

Public static String equal(char ch){

  if(ch==’a’ || ch==’e’ || ch==’i’ || ch==’o’ || ch==’u’ ){
   return “vowel”;}

  else{

   return “consonant”;

}}
```

TEST CASE PROGRAM:

```
Package javaclass;

import static org.Junit.Assert.* ;
Public class check

{

    Public void test() {
    vowel s=new vowel();
    String b=s.b.equal(“a”);
    assertEquals(“yes”,b);

}}
```

RUNS:



```
package JavaClass;

public class Vowel {

    public static String equal(char ch) {
        if(ch=='a'||ch=='e'||ch=='i'||ch=='o'||
                ch=='u') {
            return "Vowel";
        }
        else {
            return "Consonent";
        }
}
}
```

```
package JavaClass;

import static org.junit.Assert.*;

import org.junit.BeforeClass;
import org.junit.Test;

public class VowelCh {


    @Test
    public void test() {
        Vowel ct new Vowel();
        String check=ct.equal('i');
        assertEquals("Vowel",check);
    }

}
```

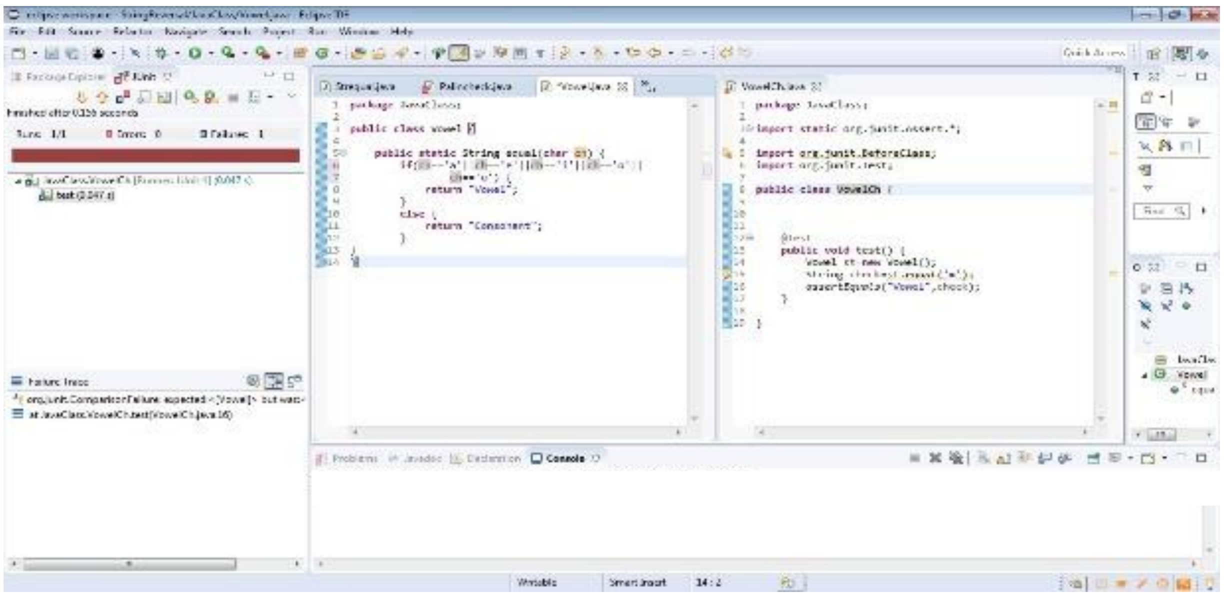
Finished after 0.047 seconds

Runs: 1/1   Errors: 0   Failures: 0

JavaClass.VowelCh [Runner: JUnit 4] (0.000 s)

<terminated> VowelCh [JUnit] D:\Program Files\Java\bin\javaw.exe (Mar 9, 2019, 2:37:31 AM)

FAILS:



```
package JavaClass;

public class Vowel {

    public static String equal(char ch) {
        if(ch=='a'||ch=='e'||ch=='i'||ch=='o'||
                ch=='u') {
            return "Vowel";
        }
        else {
            return "Consonent";
        }
}
}
```

```
package JavaClass;

import static org.junit.Assert.*;

import org.junit.BeforeClass;
import org.junit.Test;

public class VowelCh {


    @Test
    public void test() {
        Vowel ct new Vowel();
        String check=ct.equal('m');
        assertEquals("Vowel",check);
    }

}
```

Finished after 0.156 seconds

Runs: 1/1   Errors: 0   Failures: 1

JavaClass.VowelCh [Runner: JUnit 4] (0.047 s)
test (0.047 s)

Failure Trace
org.junit.ComparisonFailure: expected:<[Vowel]> but was:<
at JavaClass.VowelCh.test(VowelCh.java:16)

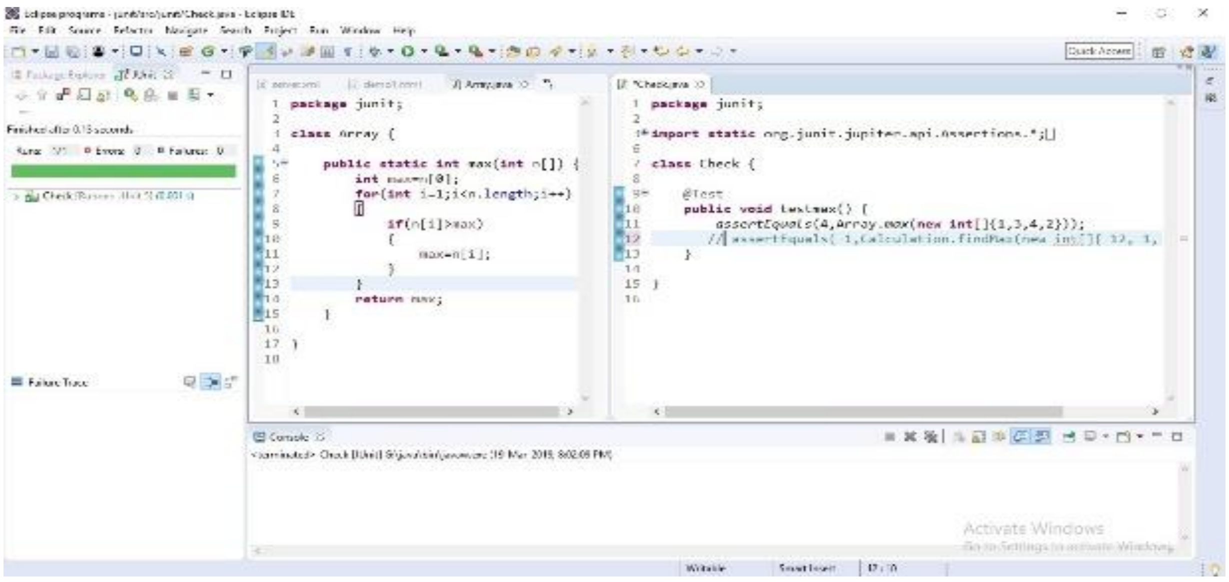
Largest number:

```
Package junit;
Class Array{
Public static int max(int n[])
{
int max=n[0];
for(int i=1;i<n.length;i++)
{
if(n[i]>max)
{
max=n[i];
}
}
 return max;
}
```

TEST CASE PROGRAM:

```
Package junit;
import static org,junit.Assertions.*;
public class check{
public void testmax()
{
assertEquals(4,Array.max(new int[] {1,3,4,2}));
}
}
```

RUNS:



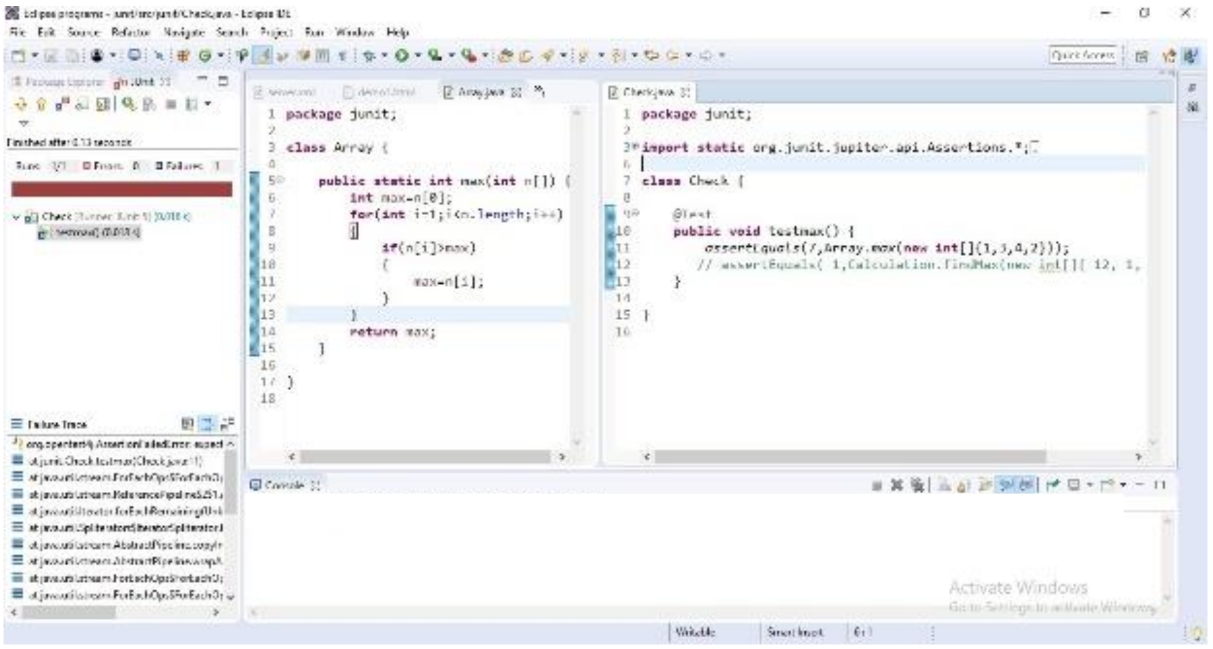
```java
package junit;

class Array {

    public static int max(int n[]) {
        int max=n[0];
        for(int i=1;i<n.length;i++)
        {
            if(n[i]>max)
            {
                max=n[i];
            }
        }
        return max;
    }

}
```

```java
package junit;

import static org.junit.jupiter.api.Assertions.*;

class Check {

    @Test
    public void testmax() {
        assertEquals(4,Array.max(new int[]{1,3,4,2}));
        // assertEquals( 1,Calculation.findMax(new int[]{ 12, 1,
    }

}
```

```
Finished after 0.15 seconds
Runs: 1/1    Errors: 0    Failures: 0
```

```
<terminated> Check [JUnit] 5\java\bin\javaw.exe (19 Mar 2019, 8:02:08 PM)
```

FAILS:



```java
package junit;

class Array {

    public static int max(int n[]) {
        int max=n[0];
        for(int i=1;i<n.length;i++)
        {
            if(n[i]>max)
            {
                max=n[i];
            }
        }
        return max;
    }

}
```

```java
package junit;

import static org.junit.jupiter.api.Assertions.*;

class Check {

    @Test
    public void testmax() {
        assertEquals(7,Array.max(new int[]{1,3,4,2}));
        // assertEquals( 1,Calculation.findMax(new int[]{ 12, 1,
    }

}
```

```
Finished after 0.13 seconds
Runs: 1/1    Errors: 0    Failures: 1
```

```
org.opentest4j.AssertionFailedError: expect
at junit.Check.testmax(Check.java:11)
at java.util.stream.ForEachOps$ForEachO
at java.util.stream.ReferencePipeline$2$1
at java.util.Iterator.forEachRemaining(Unk
at java.util.Spliterators$IteratorSpliterator
at java.util.stream.AbstractPipeline.copyIn
at java.util.stream.AbstractPipeline.wrapA
at java.util.stream.ForEachOps$ForEachO
at java.util.stream.ForEachOps$ForEachO
```

RESULT:

Continuous integration using a automated tool such as Jenkins were executed andtested successfully.