Karrasankar158@gmail.com

Java Tools

JUNIT

Example 1

Person.java

```
package com;

import java.util.logging.Logger;


public class Person {

	int a,b;
	Logger log = Logger.getAnonymousLogger();
	public int add(int a, int b){
		this.a=a;
		this.b=b;
		log.info("entered value for a ="+a);
		log.info("entered value for b ="+b);

		return a+b;
	}


	public static void main(String[] args) {
		Person p=new Person();

		p.add(2,4);

	}

}
```

PersonTest.java

```
package com;
```

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

import org.junit.After;
import org.junit.AfterClass;
import org.junit.Assert;
import org.junit.Before;
import org.junit.BeforeClass;
import org.junit.Test;

public class PersonTest {

     Person p=null;

     @BeforeClass
     public static void setUpBeforeClass() throws
Exception {
      System.out.println("setUpBeforeClass");
     }

     @AfterClass
     public static void tearDownAfterClass()
throws Exception {
      System.out.println("tearDownAfterClass");
     }

     @Before
     public void setUp() throws Exception {
      System.out.println("setUp");
      p=new Person();
     }

     @After
     public void tearDown() throws Exception {
      System.out.println("tearDown");
          p=null;
     }

     @Test
     public void testAdd() {
          Assert.assertEquals(10,p.add(2,8));
     }

     @Test
     public void testMain() {
          System.out.println("main success");
     }

}
```

output :

```
setUpBeforeClass
setUp
Dec 13, 2014 4:00:43 PM com.Person add
INFO: entered value for a =2
tearDown
Dec 13, 2014 4:00:43 PM com.Person add
INFO: entered value for b =8
setUp
main success
tearDown
tearDownAfterClass
```

Example 2

TestJunit.java

```
package com;

import static org.junit.Assert.assertEquals;

import org.junit.Test;

public class TestJunit {

@Test
public void testAdd(){
 String str="Junit is working fine";
 assertEquals("Junit is working fine", str);
}

}
```

TestRunner.java

```
package com;

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());
}//main

}

output :

true
```

JUnit write Tests :

EmployeeDetails.java

```
package com;

public class EmployeeDetails {

private String name;
private int age;
private double monthlySalary;

public String getName() {
     return name;
}
public void setName(String name) {
     this.name = name;
}
public int getAge() {
      return age;
}
public void setAge(int age) {
      this.age = age;
}
public double getMonthlySalary() {
      return monthlySalary;
}
public void setMonthlySalary(double monthlySalary)
{
      this.monthlySalary = monthlySalary;
}

}//class
```

EmpBusinessLogic.java

```
package com;

public class EmpBusinessLogic {

public double
calculateYearlySalary(EmployeeDetails empDetails){
 double yearlySalary=0;
 yearlySalary=empDetails.getMonthlySalary()*12;
 return yearlySalary;
}

public double calculateAppraisal(EmployeeDetails
empDetails){
 double appraisal=0;
 if(empDetails.getMonthlySalary()<10000){
   appraisal=500;
 }
 else{
   appraisal=1000;
 }
 return appraisal;

}
}
```

EmpTestCase.java

```
package com;

import static org.junit.Assert.assertEquals;

import org.junit.Test;

public class EmpTestCase {

EmpBusinessLogic empLogic=new EmpBusinessLogic();
EmployeeDetails empDetails=new EmployeeDetails();

@Test
public void testCalculateAppraisal(){
 empDetails.setName("Ashok");
 empDetails.setAge(27);
 empDetails.setMonthlySalary(8000d);
 double
appraisal=empLogic.calculateAppraisal(empDetails);
 assertEquals(500, appraisal,0.0);
}

@Test
public void testCalculateYearlySalary(){
 empDetails.setName("Ashok");
 empDetails.setAge(27);
 empDetails.setMonthlySalary(8000d);
 double
salary=empLogic.calculateYearlySalary(empDetails);
 assertEquals(96000,salary,0.0);
}

}//class
```

EmpTestRunner.java

```
package com;

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

public class EmpTestRunner {

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

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

 System.out.println(result.wasSuccessful());
}//main


output :

true
```

JUNIT using Assertion :

methods :

void assertEquals(boolean expected, boolean actual)

void assertTrue(boolean expected, boolean actual)

void assertFalse(boolean condition)

void assertNotNull(Object object)

void assertNull(Object object)

void assertSame(boolean condition)

void assertNotSame(boolean condition)

void assertArrayEquals(expectedArray, resultArray);

Ex :

AssertionTestCase.java

```
package com;

import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertFalse;
import static org.junit.Assert.assertNotNull;
import static org.junit.Assert.assertNotSame;
import static org.junit.Assert.assertNull;
import static org.junit.Assert.assertSame;
import static org.junit.Assert.assertTrue;
import static org.junit.Assert.*;

import org.junit.Test;

public class AssertionTestCase {

@Test
public void testAssertions(){
 String s1=new String("abc");
 String s2=new String("abc");
 String s3=null;
 String s4="abc";
 String s5="abc";
 int i1=234;
 int i2=456;
 String[] expectedArray={"one","two","three"};
 String[] resultArray={"one","two","three"};

 assertEquals(s1,s2);
 assertTrue(i1<i2);
 assertFalse(i1>i2);
 assertNotNull(s1);
 assertNull(s3);
 assertSame(s4,s5);
 assertNotSame(s1,s2);
 assertArrayEquals(expectedArray,resultArray);

}
}
```

AssertionTestRunner.java

```
package com;

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

public class AssertionTestRunner {

public static void main(String[] args) {
Result
result=JUnitCore.runClasses(AssertionTestCase.clas
s);

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

 System.out.println(result.wasSuccessful());
}//main

}
```

Annotation :

@Test

@Before

@After

@BeforeClass

@AfterClass

@Ignore

AnnotationTestCase.java

```
package com;

import org.junit.After;
import org.junit.AfterClass;
import org.junit.Before;
import org.junit.BeforeClass;
import org.junit.Ignore;
import org.junit.Test;

public class AnnotationTestCase {

@BeforeClass
public static void beforeClass(){
 System.out.println("in before class");
}

@AfterClass
public static void afterClass(){
 System.out.println("in after class");
}

@Before
public void before(){
 System.out.println("in before");
}

@After
public void after(){
 System.out.println("in after");
}

@Test
public void test(){
 System.out.println("in test");
}

@Ignore
public void ignoreTest(){
 System.out.println("in ignore test");
}
}//class
```

AnnotationTestRunner.java

```
package com;

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

public class AnnotationTestRunner {

public static void main(String[] args) {
 Result
result=JUnitCore.runClasses(AnnotationTestCase.cla
ss);
  for(Failure failure:result.getFailures()){
   System.out.println(failure.toString());
  }//for
 System.out.println(result.wasSuccessful());
}//main

}



output:
in before class
in before
in test
in after
in after class
true
```

JUNIT Executing Tests

MessageApp.java

```
package com;

public class MessageApp {
 private String message;

 public MessageApp(String message) {
     this.message=message;
}

 public String printMessage(){
  System.out.println(message);
  return message;
 }//printMessage

}
```

MessageTestCase.java

```
package com;

import static org.junit.Assert.assertEquals;

import org.junit.Test;

public class MessageTestCase {
String message="Hello";
MessageApp messageApp=new MessageApp(message);

@Test
public void testPrintMessage(){
     message="new word";  //--> 1
 assertEquals(message,messageApp.printMessage());
}

}//class
```

MessageTestRunner.java

```
package com;

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

public class MessageTestRunner {

public static void main(String[] args) {
 Result
result=JUnitCore.runClasses(MessageTestCase.class)
;
 for(Failure failure:result.getFailures()){
  System.out.println(failure.toString());
 }//for
 System.out.println(result.wasSuccessful());
}//main

}


output :
Hello
testPrintMessage(com.MessageTestCase):
expected:<[new word]> but was:<[Hello]>
false
```

in the above program, if we comments line (1) the output is : true

JUNIT Test Suit

MessageUtil.java

```
package com;

public class MessageUtil {
 private String message;

 public MessageUtil(String message) {
     this.message=message;
}

 public String printMessage(){
  System.out.println(message);
  return message;
 }

 public String salutationMessage(){
  message="Hi "+message;
```

```
   System.out.println(message);
   return message;
  }

}
```

MessageTestCaseOne.java

```
package com;

import static org.junit.Assert.assertEquals;
import org.junit.Test;

public class MessageTestCaseOne {
String message="Ashok";
MessageUtil messageUtil=new MessageUtil(message);

@Test
public void testPrintMessage(){
 System.out.println("inside testPrintMessage()");
 assertEquals(message,messageUtil.printMessage());
}

}//class
```

MessageTestCaseTwo.java

```
package com;

import static org.junit.Assert.assertEquals;

import org.junit.Test;

public class MessageTestCaseTwo {
String message="Ashok";
MessageUtil messageUtil=new MessageUtil(message);

@Test
public void testSalutationMessage(){
 System.out.println("in testSalutationTest()");
 message="Hi "+"Ashok";

assertEquals(message,messageUtil.salutationMessage
());
}

}//class
```

MessageTestSuit.java

```
package com;

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

@RunWith(Suite.class)
@Suite.SuiteClasses({
     MessageTestCaseOne.class,
     MessageTestCaseTwo.class
 })

public class MessageTestSuit {

}
```

MessageTestRunner.java

```
package com;

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

public class MessageTestRunner {

public static void main(String[] args) {
 Result
result=JUnitCore.runClasses(MessageTestSuit.class)
;
 for(Failure failure:result.getFailures()){
  System.out.println(failure.toString());
 }//for
 System.out.println(result.wasSuccessful());
}//main
}



output :

inside testPrintMessage()
Ashok
in testSalutationTest()
Hi Ashok
true
```

JUNIT Ignore Test

MessageUtil.java

```
package com;

public class MessageUtil {
 private String message;

 public MessageUtil(String message) {
     this.message=message;
}

 public String printMessage(){
  System.out.println(message);
  return message;
 }

 public String salutationMessage(){
  message="Hi!"+message;
  System.out.println(message);
  return message;
 }

}
```

MessageTestCase.java

```
package com;
```

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

import org.junit.Ignore;
import org.junit.Test;

public class MessageTestCase {
String message="Ashok";
MessageUtil messageUtil=new MessageUtil(message);

@Ignore
@Test
public void testPrintMessage(){
 System.out.println("inside testPrintMessage()");
 assertEquals(message,messageUtil.printMessage());
}

@Test
public void testSalutationMessage(){
 System.out.println("in testSalutationTest()");
 message="Hi!"+"Ashok";

assertEquals(message,messageUtil.salutationMessage
());
}

}//class
```

MessageTestRunner.java

```
package com;

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

public class MessageTestRunner {

public static void main(String[] args) {
 Result
result=JUnitCore.runClasses(MessageTestCase.class)
;
 for(Failure failure:result.getFailures()){
   System.out.println(failure.toString());
  }//for
  System.out.println(result.wasSuccessful());
 }//main

 }

output:

in testSalutationTest()
Hi!Ashok
true
```

```
package com;

import static org.junit.Assert.assertEquals;

import org.junit.Ignore;
import org.junit.Test;

@Ignore
public class MessageTestCase {
String message="Ashok";
MessageUtil messageUtil=new MessageUtil(message);

@Test
public void testPrintMessage(){
 System.out.println("inside testPrintMessage()");
 assertEquals(message,messageUtil.printMessage());
}

@Test
public void testSalutationMessage(){
 System.out.println("in testSalutationTest()");
 message="Hi!"+"Ashok";

assertEquals(message,messageUtil.salutationMessage
());
}
}//class
```

output: true

JUNIT Time Test

MessageUtil.java

```
package com;

public class MessageUtil {
 private String message;

 public MessageUtil(String message) {
     this.message=message;
}

 public void printMessage(){
  System.out.println(message);
  while(true);
 }

 public String salutationMessage(){
  message="Hi!"+message;
  System.out.println(message);
  return message;
 }

}
```

MessageTestCase.java

```
package com;

import static org.junit.Assert.assertEquals;
import org.junit.Test;

public class MessageTestCase {
String message="Ashok";
MessageUtil messageUtil=new MessageUtil(message);
@Test(timeout=1000)
public void testPrintMessage(){
 System.out.println("inside testPrintMessage()");
  messageUtil.printMessage();
}

@Test
public void testSalutationMessage(){
 System.out.println("in testSalutationTest()");
 message="Hi!"+"Ashok";

assertEquals(message,messageUtil.salutationMessage
());
}

}//class
```

MessageTestRunner.java

```
package com;

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

public class MessageTestRunner {

public static void main(String[] args) {
 Result
result=JUnitCore.runClasses(MessageTestCase.class)
;
 for(Failure failure:result.getFailures()){
  System.out.println(failure.toString());
 }//for
 System.out.println(result.wasSuccessful());
}//main

}

output :
```

```
in testSalutationTest()
Hi!Ashok
inside testPrintMessage()
Ashok
testPrintMessage(com.MessageTestCase):
    test timed out after 1000 milliseconds
false
```

JUNIT Exception Test

MessageUtil.java

```
package com;

public class MessageUtil {
 private String message;

 public MessageUtil(String message) {
     this.message=message;
}

 public void printMessage(){
  System.out.println(message);
   int a=0;
   int b=1/a;
 }

 public String salutationMessage(){
  message="Hi!"+message;
  System.out.println(message);
  return message;
 }

}
```

MessageTestCase.java

```
package com;

import static org.junit.Assert.assertEquals;
import org.junit.Test;

public class MessageTestCase {
```

```
String message="Ashok";
MessageUtil messageUtil=new MessageUtil(message);

@Test(expected=ArithmeticException.class)
public void testPrintMessage(){
 System.out.println("inside testPrintMessage()");
  messageUtil.printMessage();
}

@Test
public void testSalutationMessage(){
 System.out.println("in testSalutationTest()");
 message="Hi!"+"Ashok";

assertEquals(message,messageUtil.salutationMessage
());
}

}//class
```

MessageTestRunner.java

```
package com;

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

public class MessageTestRunner {

public static void main(String[] args) {
 Result
result=JUnitCore.runClasses(MessageTestCase.class)
;
 for(Failure failure:result.getFailures()){
  System.out.println(failure.toString());
 }//for
 System.out.println(result.wasSuccessful());
}//main

}
```

```
output:

in testSalutationTest()
Hi!Ashok
inside testPrintMessage()
Ashok
true
```

JUNIT Parameterized Test

PrimeNumberChecker.java

```
package com;

public class PrimeNumberChecker {

 public Boolean validate(final Integer
primeNumber){
      for(int i=2;i<(primeNumber/2);i++){
           if(primeNumber%i==0)
                return false;
      }//for
      return true;
 }//validate
}
```

PrimeCheckerTestCase.java

```
package com;

import java.util.Arrays;
import java.util.Collection;

import org.junit.Before;
import org.junit.Test;
import org.junit.runner.RunWith;
import org.junit.runners.Parameterized;
import static org.junit.Assert.assertEquals;

@RunWith(Parameterized.class)
public class PrimeCheckerTestCase {
 private Integer inputNumber;
 private boolean expectedResult;
 private PrimeNumberChecker primeNumberChecker;

 @Before
 public void initialize(){
  primeNumberChecker=new PrimeNumberChecker();
  System.out.println("in initialize");
 }

 public PrimeCheckerTestCase(Integer inputNumber,
                              boolean
expectedResult){
  this.inputNumber=inputNumber;
  this.expectedResult=expectedResult;
  System.out.println("constructor");
 }

 @Parameterized.Parameters
 public static Collection primeNumbers(){
      System.out.println("In primeNumbers()");
   return Arrays.asList(new Object[][]{
      {2,true},
      {6,false},
      {19,true},
      {22,false},
      {23,true}
   });

 }//primeNumbers

 @Test
public void testPrimeNumberChecker(){
 System.out.println("parameterized No is:
"+inputNumber);

assertEquals(expectedResult,primeNumberChecker.val
idate(inputNumber));
}

}//class
```

PrimeTestRunner.java

```
package com;

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

public class PrimeTestRunner {

public static void main(String[] args) {
 Result
result=JUnitCore.runClasses(PrimeCheckerTestCase.c
lass);
 for(Failure failure:result.getFailures()){
  System.out.println(failure.toString());
 }//for
 System.out.println(result.wasSuccessful());
}//main

}


output:

In primeNumbers()

constructor
in initialize
parameterized No is: 2

constructor
in initialize
parameterized No is: 6

constructor
in initialize
parameterized No is: 19

constructor
in initialize
```

```
parameterized No is: 22

constructor
in initialize
parameterized No is: 23

true
```

Agile Methodology

Introduction :

Agility in Software Development :

1. Agile Model
2. Agile Development and Principles

Agile Software Development Methodologies

Agile development methodologies are emerging in the software industry.

we provide an introduction to agile development methodologies and an overview of 4 specific methodologies:

1. Extreme Programming
2. Crystal Methods
3. Scrum
4. Feature Driven Development

Extreme Programming (XP)

1. communication
2. simplicity

3. feedback
4. courage
5. respect

Documents and Artifacts

- User story cards, paper index cards
- Task list
- CRC cards (optional)
- Customer acceptance tests
- Visible Wall Graphs

Roles

- Manager
- Coach
- Tracker
- Programmer
- Tester
- Customer

Process

primary technical practices of XP (13)

1. Sit together
2. Whole team
3. Informative workspace
4. Energized work
5. Pair programming
6. Stories
7. Weekly cycle

8. Quarterly cycle
9. Slack
10. Ten-minute build
11. Test-first programming
12. Continuous integration
13. Incremental design

corollary technical practices of XP (11)

1. Real customer involvement
2. Incremental deployment
3. Team continuity
4. Shrinking team
5. Root cause analysis
6. Shared code
7. Code and tests
8. Daily deployment
9. Negotiated scope contract
10. Pay-per-use

11. Stand-Up Meetings :

1. What he or she accomplished the prior day
2. What he or she plans to do today
3. Any obstacles or difficulties he or she is experiencing

Crystal Methods

Crystal Clear

1. Documents and artifacts
2. Roles

3. Process

Crystal Orange

1. Documents and artifacts
2. Roles
3. Process

Scrum

Overview

Documents and Artifacts

1. Product Backlog
2. Sprint Backlog
3. Sprint Burndown chart

Roles

- Product Owner
- Scrum Master
- Developer

Process

Feature Driven Development (FDD)

Documents and Artifacts

1. Feature lists
2. Design packages
3. Track by Feature
4. "Burn Up" Chart

Roles

- Project manager
- Chief architect
- Development manager
- Chief programmer
- Class owner
- Domain experts
- Feature teams

Process

1. Develop an overall model
2. Build a features list
3. Plan by feature
4. Design by feature
5. Build by feature