

STQA Mini Project No.1

Roll-61,62,63,64

Title

Create a small application by selecting relevant system environment/ platform and programming languages. Narrate concise Test Plan consisting features to be tested and bug taxonomy. Prepare Test Cases inclusive of Test Procedures for identified Test Scenarios. Perform selective Black-box and White-box testing covering Unit and Integration test by using suitable Testing tools. Prepare Test Reports based on Test Pass/Fail Criteria and judge the acceptance of application developed.

Problem Definition:

Perform Desktop Application testing using Automation Tool like JUnit generate Test Report by Using tool like Apache Maven.

Prerequisite:

Knowledge of Core Java, Basic Concepts of Unit Testing, Test Cases Writing using Junit etc tool .

Software Requirements:

JDK 1.8, Eclipse java photon-R version, TestNG .

Hardware Requirement:

PIV, 2GB RAM, 500 GB HDD, Lenovo A13-4089Model.

Learning Objectives:

We are going to learn how to Prepare Test Cases inclusive of Test Procedures for identified Test Scenarios. Perform selective Black-box and White-box testing covering Unit and Integration test by using suitable Testing tools. also Prepare Test Reports based on Test Pass/Fail Criteria.

Outcomes:

You are able to understand Unit and Integration testing with Tool with Test Report.

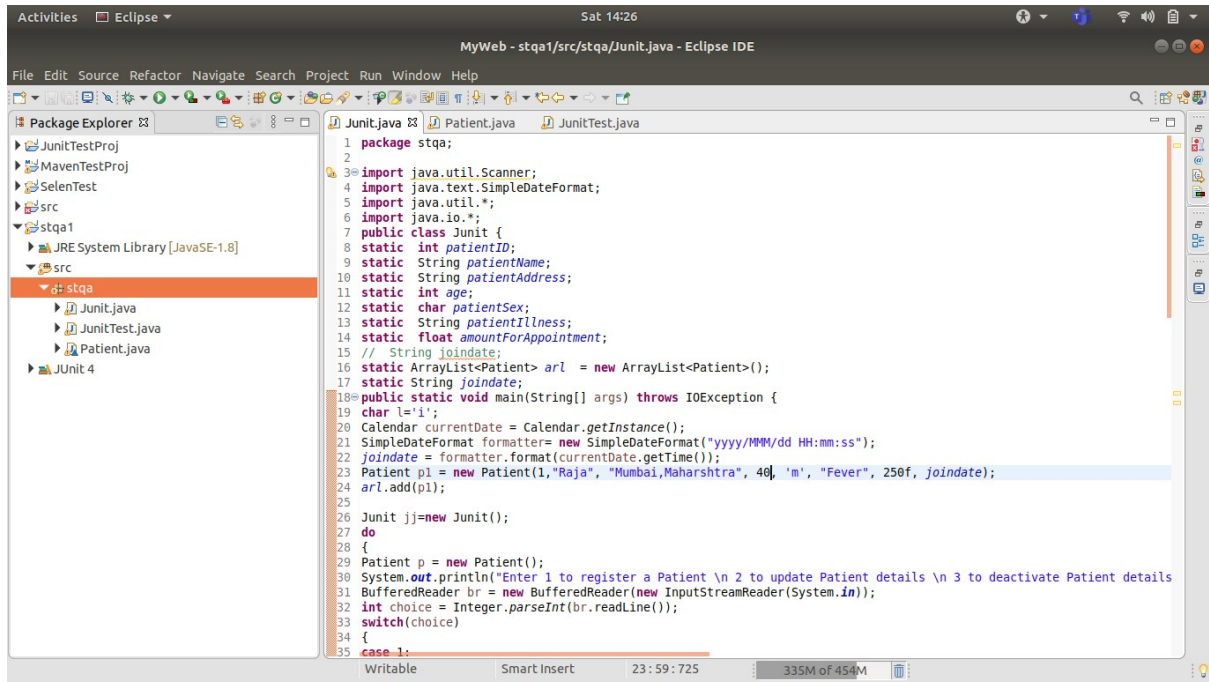
Theory Concepts:

What is Unit Testing?

Unit Testing of software applications is done during the development (coding) of an application.

The objective of Unit Testing is to isolate a section of code and verify its correctness. In procedural programming a unit may be an individual function or procedure

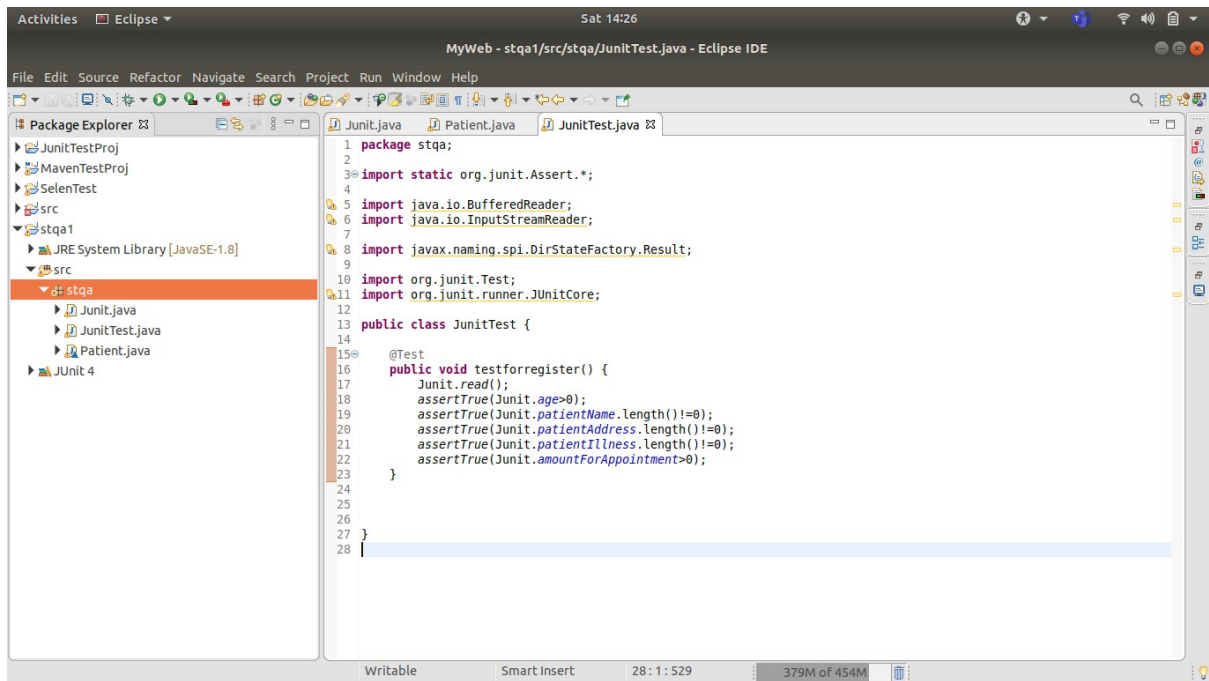
The goal of Unit Testing is to isolate each part of the program and show that the individual parts are correct. Unit Testing is usually performed by the developer.



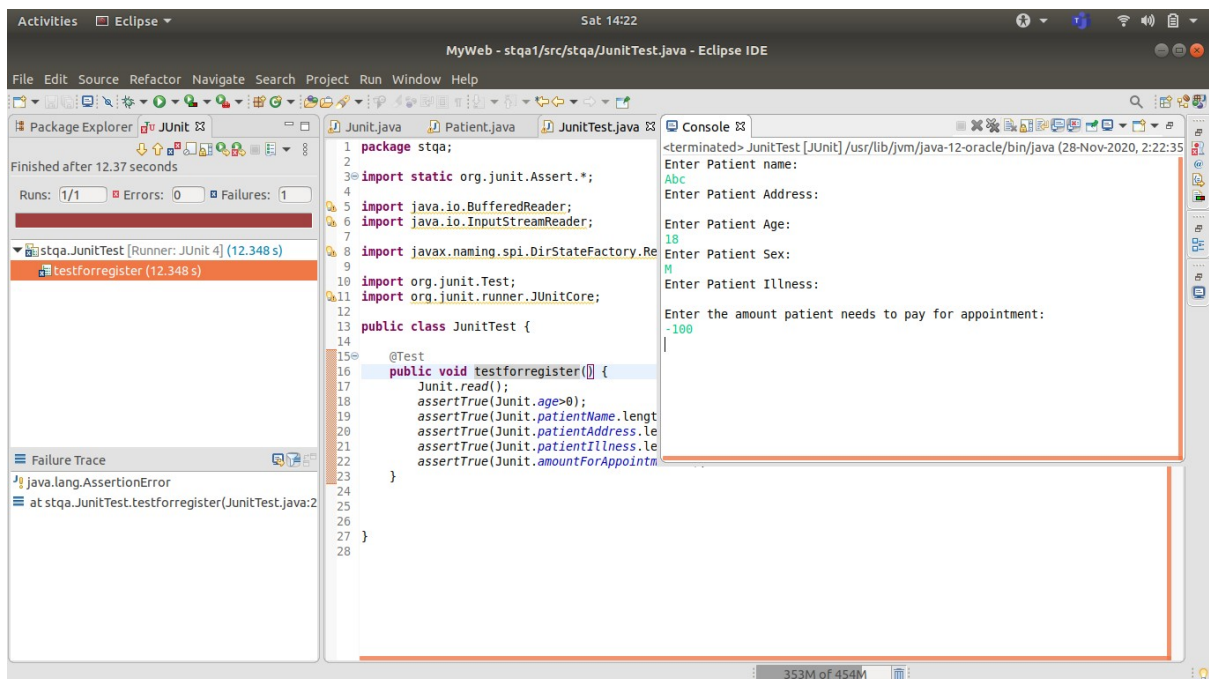
The screenshot shows the Eclipse IDE interface. The Package Explorer on the left displays a project named 'JunittestProj' with a source folder 'src' containing files 'Junittest.java', 'Patient.java', and 'JUnit4'. The main editor window shows the 'Junittest.java' file with the following code:

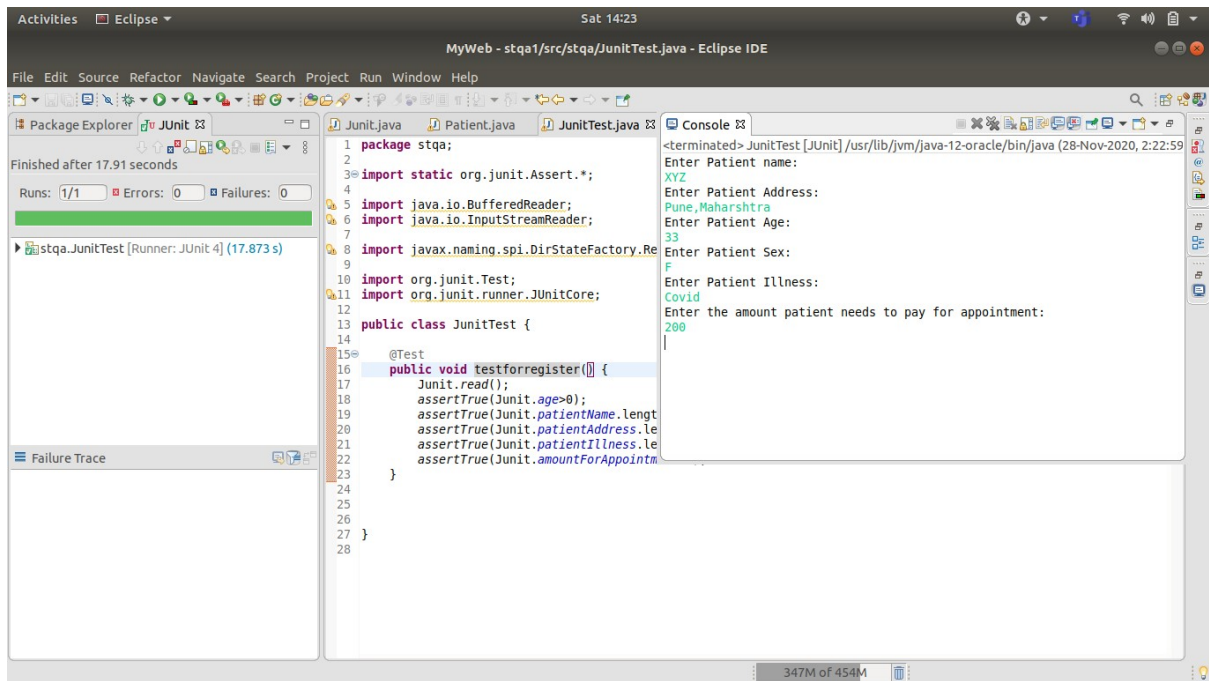
```
1 package stqa;
2
3 import java.util.Scanner;
4 import java.text.SimpleDateFormat;
5 import java.util.*;
6 import java.io.*;
7 public class Junit {
8     static int patientID;
9     static String patientName;
10    static String patientAddress;
11    static int age;
12    static char patientSex;
13    static String patientIllness;
14    static float amountForAppointment;
15    // String joindate;
16    static ArrayList<Patient> arl = new ArrayList<Patient>();
17    static String joindate;
18    public static void main(String[] args) throws IOException {
19        char l='i';
20        Calendar currentDate = Calendar.getInstance();
21        SimpleDateFormat formatter= new SimpleDateFormat("yyyy/MM/dd HH:mm:ss");
22        joindate = formatter.format(currentDate.getTime());
23        Patient p1 = new Patient(1,"Raja", "Mumbai,Maharashtra", 40, 'm', "Fever", 250f, joindate);
24        arl.add(p1);
25
26        Junit jj=new Junit();
27        do
28        {
29            Patient p = new Patient();
30            System.out.println("Enter 1 to register a Patient \n 2 to update Patient details \n 3 to deactivate Patient details");
31            BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
32            int choice = Integer.parseInt(br.readLine());
33            switch(choice)
34            {
35                case 1:
```

Junittest file

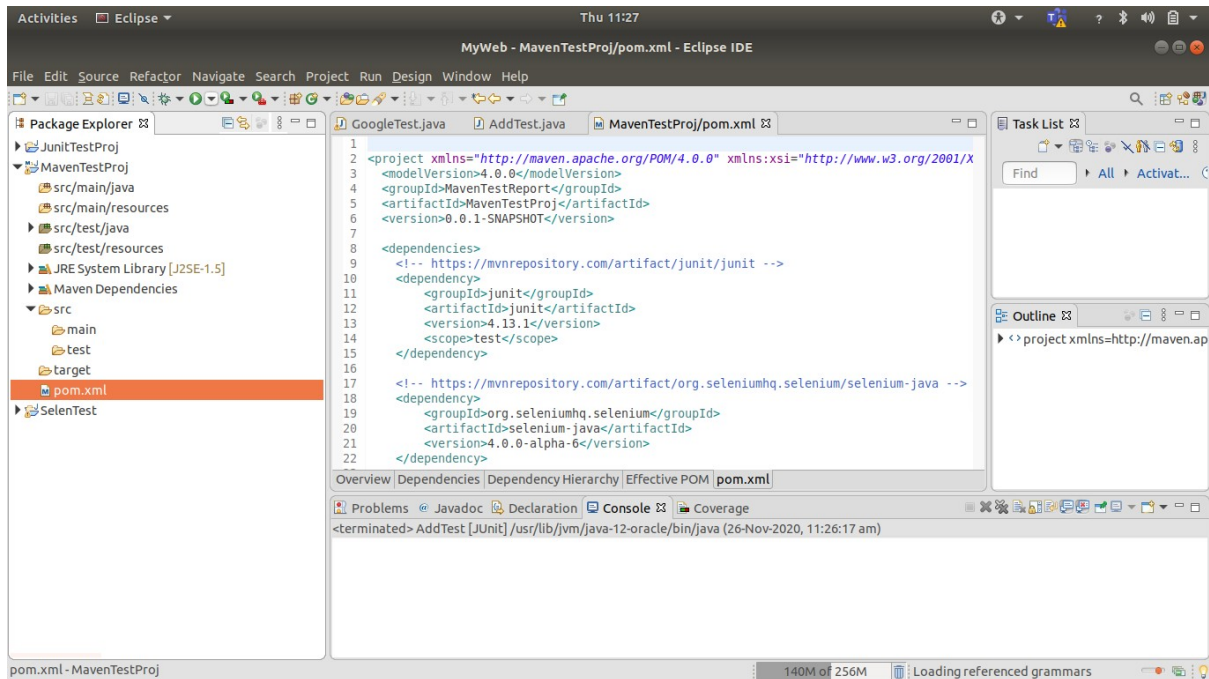


Lets run Junit_Test test case. Right click Junit_Test-> Debug As->JUnit Test





Create Test Report Using Apache Maven



31 . Now open terminal-> Go to our Mave Project Folder -> write mvn clean

```
Activities Terminal Sat 14:33
Terminal
File Edit View Search Terminal Help
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ MavenTestProj ---
[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] Copying 0 resource
[INFO]
[INFO] --- maven-compiler-plugin:3.1:testCompile (default-testCompile) @ MavenTestProj ---
[INFO] Changes detected - recompiling the module!
[WARNING] File encoding has not been set, using platform encoding UTF-8, i.e. build is platform dependent!
[INFO] Compiling 3 source files to /home/sky/Documents/MTL/MyWeb/MavenTestProj/target/test-classes
[INFO]
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ MavenTestProj ---
[INFO] Surefire report directory: /home/sky/Documents/MTL/MyWeb/MavenTestProj/target/surefire-reports

-----
T E S T S
-----
Running stqa.JunitTest
Enter Patient name:
Enter Patient Address:
Enter Patient Age:
Enter Patient Sex:
Enter Patient Illness:
Enter the amount patient needs to pay for appointment:
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.093 sec

Results :

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0

[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 3.288 s
[INFO] Finished at: 2020-11-28T14:33:14+05:30
[INFO] -----
(base) sky@om:~/Documents/MTL/MyWeb/MavenTestProj$
```

```
Activities Terminal Sat 14:33
Terminal
File Edit View Search Terminal Help
at org.apache.maven.surefire.junit4.JUnit4Provider.executeTestSet(JUnit4Provider.java:141)
at org.apache.maven.surefire.junit4.JUnit4Provider.invoke(JUnit4Provider.java:112)
at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
at java.lang.reflect.Method.invoke(Method.java:498)
at org.apache.maven.surefire.util.ReflectionUtils.invokeMethodWithArray(ReflectionUtils.java:189)
at org.apache.maven.surefire.booter.ProviderFactory$ProviderProxy.invoke(ProviderFactory.java:165)
at org.apache.maven.surefire.booter.ProviderFactory.invokeProvider(ProviderFactory.java:85)
at org.apache.maven.surefire.booter.ForkedBooter.runSuitesInProcess(ForkedBooter.java:115)
at org.apache.maven.surefire.booter.ForkedBooter.main(ForkedBooter.java:75)

Results :

Failed tests: testforregister(stqa.JunitTest)

Tests run: 1, Failures: 1, Errors: 0, Skipped: 0

[INFO] -----
[INFO] BUILD FAILURE
[INFO] -----
[INFO] Total time: 3.774 s
[INFO] Finished at: 2020-11-28T14:33:40+05:30
[INFO] -----
[ERROR] Failed to execute goal org.apache.maven.plugins:maven-surefire-plugin:2.12.4:test (default-test) on project MavenTestProj: There are test failures.
[ERROR] Please refer to /home/sky/Documents/MTL/MyWeb/MavenTestProj/target/surefire-reports for the individual test results.
[ERROR] -> [Help 1]
[ERROR] To see the full stack trace of the errors, re-run Maven with the -e switch.
[ERROR] Re-run Maven using the -X switch to enable full debug logging.
[ERROR] For more information about the errors and possible solutions, please read the following articles:
[ERROR] [Help 1] http://cwiki.apache.org/confluence/display/MAVEN/MojoFailureException
(base) sky@om:~/Documents/MTL/MyWeb/MavenTestProj$
```

Conclusion :

We Successfully completed the stqa miniproject 1 for performing the unit testing in java using selenium and junit.

Code:

JUnit.java file

```
package stqa;

import java.util.Scanner;
import java.text.SimpleDateFormat;
import java.util.*;
import java.io.*;

public class Junit {

    static      int patientID;
    static      String patientName;
    static      String patientAddress;
    static      int age;
    static      char patientSex;
    static      String patientIllness;
    static      float amountForAppointment;
    //      String joindate;

    static ArrayList<Patient> arl = new ArrayList<Patient>();
    static String joindate;

    public static void main(String[] args) throws IOException {
```

```
char l='i';

Calendar currentDate = Calendar.getInstance();

SimpleDateFormat formatter= new SimpleDateFormat("yyyy/MMM/dd HH:mm:ss");

joindate = formatter.format(currentDate.getTime());

Patient p1 = new Patient(1, "Raja", "Mumbai, Maharashtra", 40, 'm', "Fever",
250f, joindate);

arl.add(p1);
```

```
Junit jj=new Junit();
```

```
do
```

```
{
```

```
Patient p = new Patient();
```

```
System.out.println("Enter 1 to register a Patient \n 2 to update Patient
details \n 3 to deactivate Patient details \n 4 to display Patient
details");
```

```
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
```

```
int choice = Integer.parseInt(br.readLine());
```

```
switch(choice)
```

```
{
```

```
case 1:
```

```
p.registerPatient();
```

```
System.out.println("Patient Registered Successfully !!!");
```

```
p.showPatientDetails();
```

```
break;
```

```
case 2:
```

```
p.showPatientDetails();
```

```
p.updatePatientDetails();
```

```
break;
```

```
case 3:
```

```
p.removeInactivePatient();
```

```
break;
```

case 4:

```
System.out.println("patient-ID \t Patient-Name \t Address \t\t Age \t Sex \t Illness \t Fees \t Join-date");
```

```
for(int i=0; i<jj.arl.size(); i++)
```

```
{
```

```
System.out.println(i+1 + " \t\t " +jj.arl.get(i).patientName+" \t\t "+jj.arl.get(i).patientAddress+" \t " +jj.arl.get(i).age+" \t "+jj.arl.get(i).patientSex+" \t " +jj.arl.get(i).patientIllness+" \t\t "+jj.arl.get(i).amountForAppointment+" \t " +jj.arl.get(i).joindate);
```

```
}
```

```
break;
```

default:

```
System.out.println("Patient does not exist with the entered ID");
```

```
System.out.println("Try again");
```

```
break;
```

```
}
```

```
System.out.println("Do you want to continue selecting options (y/n):");
```

```
l=(char)br.read();
```

```
}while(l=='y');
```

```
/*End of main() method */
```

```
public static void read() {
```

```
    BufferedReader br = new BufferedReader(new  
    InputStreamReader(System.in));
```

```
    Patient pr=new Patient();
```

```
    pr.patientID = arl.size()+1;
```

```
    try {
```

```
        System.out.println("Enter Patient name:");
```

```
        patientName = br.readLine();
```



```

pr.patientName =patientName;
System.out.println("Enter Patient Address:");
patientAddress = br.readLine();
pr.patientAddress=patientAddress;
System.out.println("Enter Patient Age:");
age = Integer.parseInt(br.readLine());
pr.age =age;
System.out.println("Enter Patient Sex:");
String temp = br.readLine();
patientSex = temp.charAt(0);
pr.patientSex =temp.charAt(0);
System.out.println("Enter Patient Illness:");
patientIllness = br.readLine();
pr.patientIllness=patientIllness;
System.out.println("Enter the amount patient needs to pay for
appointment:");
amountForAppointment = Float.parseFloat(br.readLine());
pr.amountForAppointment =amountForAppointment;
pr.joindate = joindate;
}catch(Exception e)
{
    e.printStackTrace();
}
arl.add(pr);
}
}

```

Patient.java file

```
package stqa;
```

```
import java.io.BufferedReader;
```

```
import java.io.IOException;
```

```
import java.io.InputStreamReader;
```

```
import java.text.SimpleDateFormat;
```

```
import java.util.Calendar;
```

```
class Patient implements Comparable<Patient>
```

```
{
```

```
int patientID;
```

```
String patientName;
```

```
String patientAddress;
```

```
int age;
```

```
char patientSex;
```

```
String patientIllness;
```

```
float amountForAppointment;
```

```
String joindate;
```

```
Junit hm = new Junit();
```

```
public int compareTo(Patient p)
```

```
{
```

```
return this.patientID - p.patientID;
```

```
}
```

```
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
```

```
Patient(){
```

```
    this.patientID=0;
```

```
}
```

```
Patient(int patientID,String patientName,String patientAddress,int age,char  
patientSex,String patientIllness,float amountForAppointment, String  
joindate)
```

```

{
this.patientID=patientID;
this.patientName=patientName;
this.patientAddress=patientAddress;
this.age=age;
this.patientSex=patientSex;
this.patientIllness=patientIllness;
this.amountForAppointment=amountForAppointment;
this.joindate = joindate;
}

void registerPatient()throws IOException //function to insert new patient
records
{
Patient pr = new Patient();
pr.patientID = hm.arl.size()+1;
System.out.println("Enter Patient name:");
pr.patientName = br.readLine();
System.out.println("Enter Patient Address:");
pr.patientAddress = br.readLine();
System.out.println("Enter Patient Age:");
pr.age = Integer.parseInt(br.readLine());
System.out.println("Enter Patient Sex:");
String temp = br.readLine();
pr.patientSex =temp.charAt(0);
System.out.println("Enter Patient Illness:");
pr.patientIllness = br.readLine();
System.out.println("Enter the amount patient needs to pay for
appointment:");
pr.amountForAppointment = Float.parseFloat(br.readLine());
pr.joindate = hm.joindate;
hm.arl.add(pr);

```

```

}

@SuppressWarnings("static-access")

void removeInactivePatient() throws IOException //function to remove
patient records

{
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
    System.out.println("Enter Patient ID:");
    int id1 = Integer.parseInt(br.readLine());
    int flag=0;
    int flag2=0;
    String presentdate;
    String afteraddingdays;
    for(int i=0; i<hm.arl.size(); i++)
    {
        if(id1 != hm.arl.get(i).patientID)
        {
            flag=0;
        }
        else if(id1 == hm.arl.get(i).patientID)
        {
            presentdate = hm.arl.get(i).joindate;
            Calendar cal = Calendar.getInstance();
            cal.add(Calendar.DAY_OF_MONTH, 15);
            SimpleDateFormat formatter = new SimpleDateFormat("yyyy/MMM/dd HH:mm:ss");
            afteraddingdays = formatter.format(cal.getTime());
            flag=1;
            if(presentdate == afteraddingdays)
            {
                hm.arl.remove(i);
                System.out.println("Patient deleted as his validity expired");
            }
        }
    }
}

```

```

flag2=1;
}
else{
flag2=0;
}
}
}
}
if((flag) == 0)
{
System.out.println("Patient with the entered ID does not exist");
}
if(flag2==0)
{
System.out.println("Patient has still days left before his appointment
expires");
}
}
}
/*Function to update patient records*/
void updatePatientDetails() throws IOException
{
char l='n';

Patient p2 = new Patient(patientID, patientName, patientAddress, age,
patientSex, patientIllness, amountForAppointment, joindate);

do{
System.out.println("Enter patient ID you want to update:");
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
int idnum = Integer.parseInt(br.readLine());
for(int i=0; i<hm.arl.size(); i++)
{
if(idnum == hm.arl.get(i).patientID)
{

```

```
System.out.println("Enter 1 to change patient's name \n 2 to change
patient's address \n 3 to change patient's age \n 4 to change patient's
illness \n 5 to change registration fees along with the medical expenses");

int ch = Integer.parseInt(br.readLine());

switch(ch)
{
case 1:
System.out.println("Enter new patient's name:");
p2.patientName = br.readLine();
p2.patientID= hm.arl.get(i).patientID;
p2.patientAddress= hm.arl.get(i).patientAddress;
p2.age=hm.arl.get(i).age;
p2.patientSex=hm.arl.get(i).patientSex;
p2.patientIllness = hm.arl.get(i).patientIllness;
p2.amountForAppointment = hm.arl.get(i).amountForAppointment;
p2.joindate = hm.arl.get(i).joindate;
hm.arl.set(i,p2);
System.out.println("Patient updated !!!");
break;
case 2:
System.out.println("Enter new patient Address:");
p2.patientAddress = br.readLine();
p2.patientID = hm.arl.get(i).patientID;
p2.patientName = hm.arl.get(i).patientName;
p2.age = hm.arl.get(i).age;
p2.patientSex = hm.arl.get(i).patientSex;
p2.patientIllness = hm.arl.get(i).patientIllness;
p2.amountForAppointment = hm.arl.get(i).amountForAppointment;
p2.joindate = hm.arl.get(i).joindate;
hm.arl.set(i,p2);
System.out.println("Patient updated !!!");
```

```
break;
case 3:
System.out.println("Enter new Patient age:");
p2.age = Integer.parseInt(br.readLine());
p2.patientID = hm.arl.get(i).patientID;
p2.patientName = hm.arl.get(i).patientName;
p2.patientAddress = hm.arl.get(i).patientAddress;
p2.patientSex = hm.arl.get(i).patientSex;
p2.patientIllness = hm.arl.get(i).patientIllness;
p2.amountForAppointment = hm.arl.get(i).amountForAppointment;
p2.joindate= hm.arl.get(i).joindate;
hm.arl.set(i,p2);
System.out.println("Patient updated !!!");
break;
case 4:
System.out.println("Enter new Patient illness:");
p2.age = hm.arl.get(i).age;
p2.patientID= hm.arl.get(i).patientID;
p2.patientName = hm.arl.get(i).patientName;
p2.patientAddress = hm.arl.get(i).patientAddress;
p2.patientSex = hm.arl.get(i).patientSex;
p2.patientIllness = br.readLine();
p2.amountForAppointment = hm.arl.get(i).amountForAppointment;
p2.joindate = hm.arl.get(i).joindate;
hm.arl.set(i,p2);
System.out.println("Patient updated !!!");
break;
case 5:
System.out.println("Enter the new amount that patient needs to pay:");
p2.age = hm.arl.get(i).age;
```

```

p2.patientID = hm.arl.get(i).patientID;
p2.patientName = hm.arl.get(i).patientName;
p2.patientAddress = hm.arl.get(i).patientAddress;
p2.patientSex = hm.arl.get(i).patientSex;
p2.patientIllness = hm.arl.get(i).patientIllness;
p2.amountForAppointment = Float.parseFloat(br.readLine());
p2.joindate = hm.arl.get(i).joindate;
hm.arl.set(i,p2);
System.out.println("Patient details updated !!!");
break;
default:
System.out.println("Invalid choice.");
break;
} /*End of switch block */
}/*End of if block */
} /*End of for block */
System.out.println("Do you want to continue updating (y/n):");
l=(char)br.read();
}while(l=='y'); /*End of do-while block */
}

/* Function to display patients details*/
void showPatientDetails()
{
System.out.println("patient-ID \t Patient-Name \t Address \t\t Age \t Sex \t
\t Illness \t Fees \t Join-date");

for(int i=0; i<hm.arl.size(); i++)
{
System.out.println(hm.arl.get(i).patientID + " " + hm.arl.get(i).patientName + " " + hm.arl.get(i).patientAddress + " " + hm.arl.get(i).age + " " + hm.arl.get(i).patientSex + " " + hm.arl.get(i).patientIllness + " " + hm.arl.get(i).amountForAppointment + " " + hm.arl.get(i).joindate);

```



```
}  
}  
}
```

JUnitTest.java

```
package stqa;
```

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

```
import java.io.BufferedReader;
```

```
import java.io.InputStreamReader;
```

```
import javax.naming.spi.DirStateFactory.Result;
```

```
import org.junit.Test;
```

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

```
public class JUnitTest {
```

```
    @Test
```

```
    public void testforregister() {
```

```
        Junit.read();
```

```
        assertTrue(Junit.age>0);
```

```
        assertTrue(Junit.patientName.length()!=0);
```

```
        assertTrue(Junit.patientAddress.length()!=0);
```

```
        assertTrue(Junit.patientIllness.length()!=0);
```

```
        assertTrue(Junit.amountForAppointment>0);
```

```
    }
```

}

Output:

Enter 1 to register a Patient

2 to update Patient details

3 to deactivate Patient details

4 to display Patient details

1

Enter Patient name:

abc

Enter Patient Address:

Pune,Maharashtra

Enter Patient Age:

45

Enter Patient Sex:

M

Enter Patient Illness:

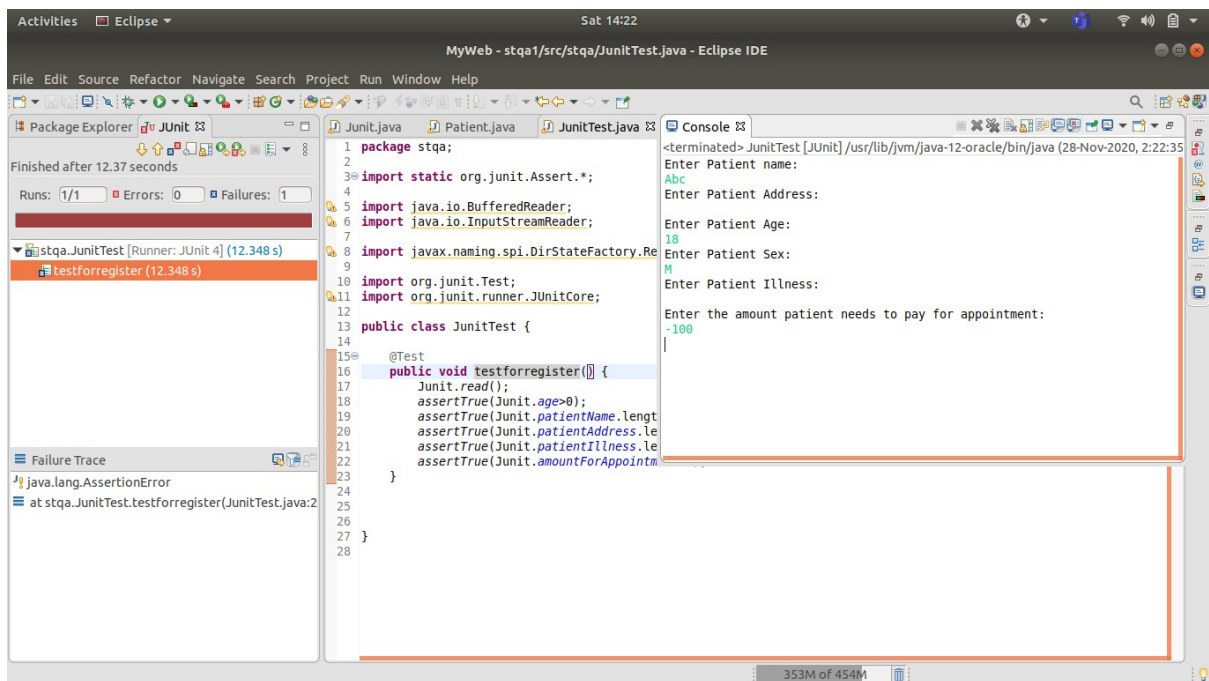
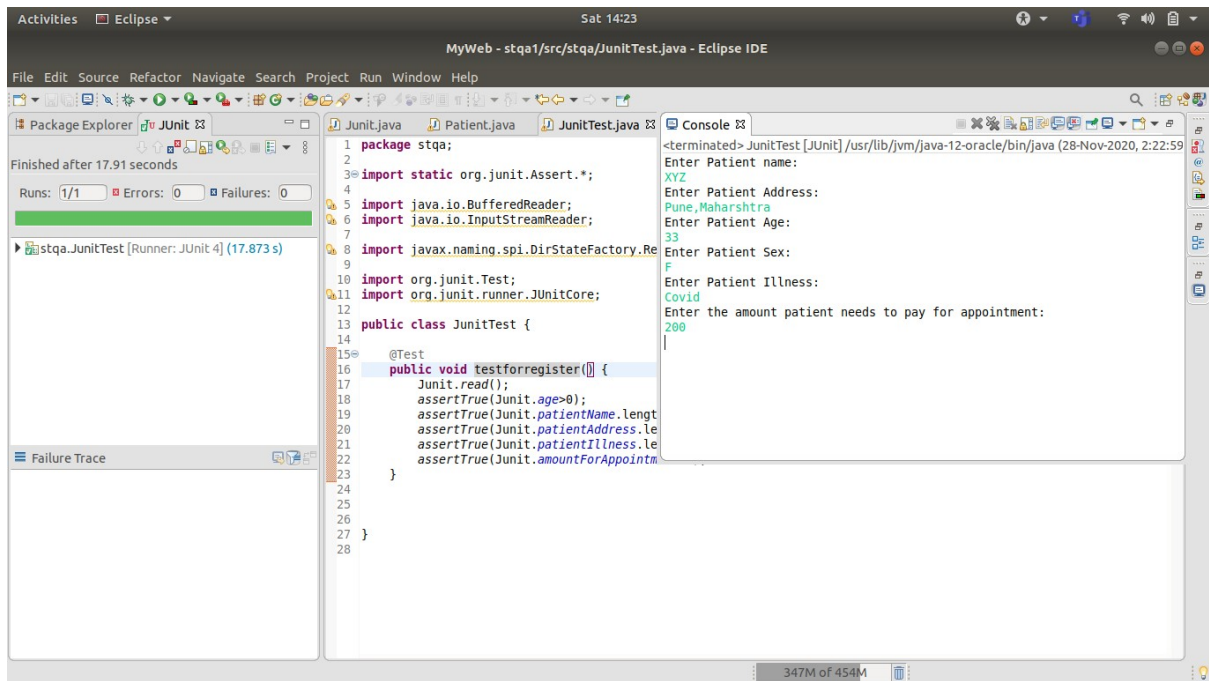
Covid

Enter the amount patient needs to pay for appointment:

200

Patient Registered Successfully !!!

patient-ID	Patient-Name	Address	Age	Sex	Illness
Fees	Join-date				
1	Raja	Mumbai,Maharashtra	40	m	Fever
	250.0	2020/Nov/28 14:35:41			
2	abc	Pune,Maharashtra	45	M	Covid
	200.0	2020/Nov/28 14:35:41			



TEST PLAN

Sr.No	Description of Tests
1	Check for test if user entered positive age
2	Check whether user has not put empty address
3	Check whether user has not put empty gender field
4	Check whether user has not put empty illness field
5	Check whether user has paid the fee

TESTING TECHNOLOGY USED

TYPE OF TESTING	PACKAGE,LIBRARY USED
UNIT TESTING	Junit,Eclipse J2EE

EXECUTION REPORT

For Failed TEST CASE

Test set: stqa.JunitTest

Tests run: 1, Failures: 1, Errors: 0, Skipped: 0, Time elapsed: 0.104 sec <<< FAILURE!

testforregister(stqa.JunitTest) Time elapsed: 0.018 sec <<< FAILURE!

java.lang.AssertionError

at org.junit.Assert.fail(Assert.java:87)

at org.junit.Assert.assertTrue(Assert.java:42)

at org.junit.Assert.assertTrue(Assert.java:53)

at stqa.JunitTest.testforregister(JunitTest.java:18)

at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)

at

sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)

at

sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)

at java.lang.reflect.Method.invoke(Method.java:498)

at

org.junit.runners.model.FrameworkMethod\$1.runReflectiveCall(FrameworkMethod.java:59)

at

org.junit.internal.runners.model.ReflectiveCallable.run(ReflectiveCallable.java:12)

at

org.junit.runners.model.FrameworkMethod.invokeExplosively(FrameworkMethod.java:56)

at

org.junit.internal.runners.statements.InvokeMethod.evaluate(InvokeMethod.java:17)

at org.junit.runners.ParentRunner\$3.evaluate(ParentRunner.java:306)

at

org.junit.runners.BlockJUnit4ClassRunner\$1.evaluate(BlockJUnit4ClassRunner.java:100)

at org.junit.runners.ParentRunner.runLeaf(ParentRunner.java:366)

```
        at
org.junit.runners.BlockJUnit4ClassRunner.runChild(BlockJUnit4ClassRunner.java:1
03)

        at
org.junit.runners.BlockJUnit4ClassRunner.runChild(BlockJUnit4ClassRunner.java:6
3)

    at org.junit.runners.ParentRunner$4.run(ParentRunner.java:331)
    at org.junit.runners.ParentRunner$1.schedule(ParentRunner.java:79)
    at org.junit.runners.ParentRunner.runChildren(ParentRunner.java:329)
    at org.junit.runners.ParentRunner.access$100(ParentRunner.java:66)
    at org.junit.runners.ParentRunner$2.evaluate(ParentRunner.java:293)
    at org.junit.runners.ParentRunner$3.evaluate(ParentRunner.java:306)
    at org.junit.runners.ParentRunner.run(ParentRunner.java:413)

        at
org.apache.maven.surefire.junit4.JUnit4Provider.execute(JUnit4Provider.java:252
)

        at
org.apache.maven.surefire.junit4.JUnit4Provider.executeTestSet(JUnit4Provider.ja
va:141)

        at
org.apache.maven.surefire.junit4.JUnit4Provider.invoke(JUnit4Provider.java:112)
            at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)

        at
sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62
)

        at
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorIm
pl.java:43)
            at java.lang.reflect.Method.invoke(Method.java:498)

        at
org.apache.maven.surefire.util.ReflectionUtils.invokeMethodWithArray(Reflection
Utils.java:189)

        at
org.apache.maven.surefire.booter.ProviderFactory$ProviderProxy.invoke(Provide
rFactory.java:165)
```

at
org.apache.maven.surefire.booter.ProviderFactory.invokeProvider(ProviderFactor
y.java:85)

at
org.apache.maven.surefire.booter.ForkedBooter.runSuitesInProcess(ForkedBoote
r.java:115)

at
org.apache.maven.surefire.booter.ForkedBooter.main(ForkedBooter.java:75)

FOR SUCCESSFUL RUNNING TESTCASE

Test set: stqa.JunitTest

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.089 sec