

EXP NO: 9**DATE:** _____**AIM:**

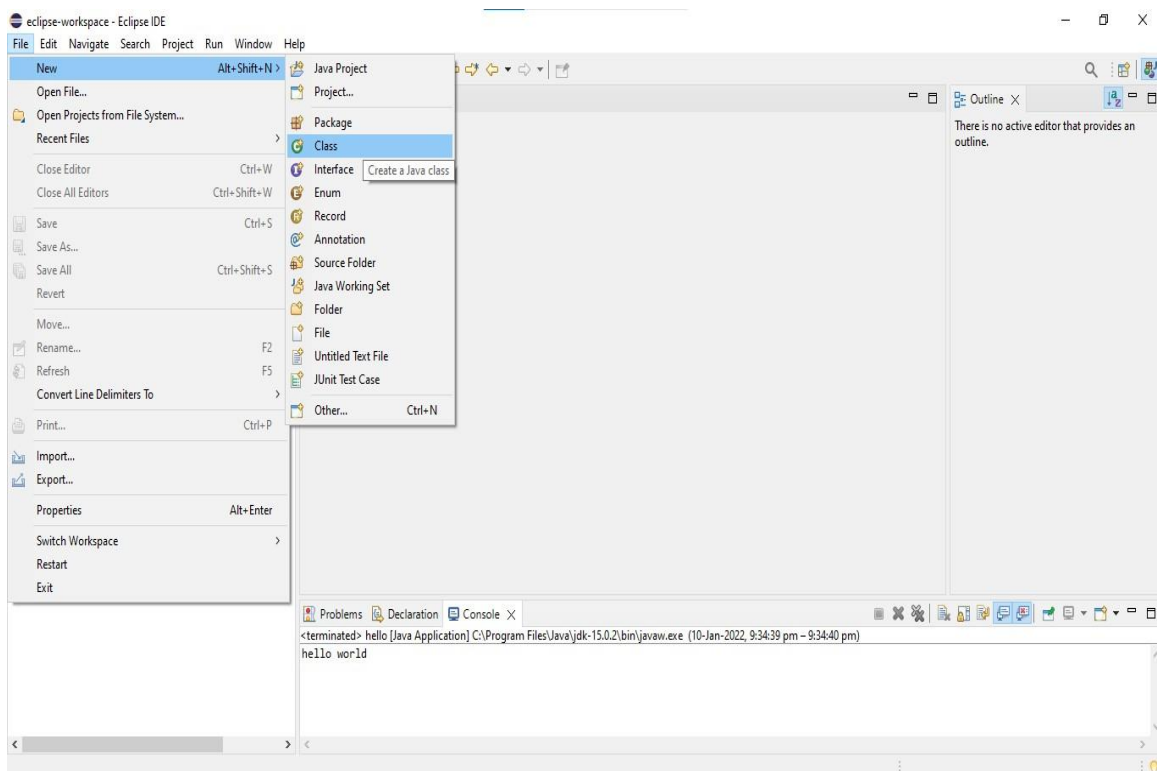
To Perform class level testing and measure coverage using tools such as Cobertura

DESCRIPTION:

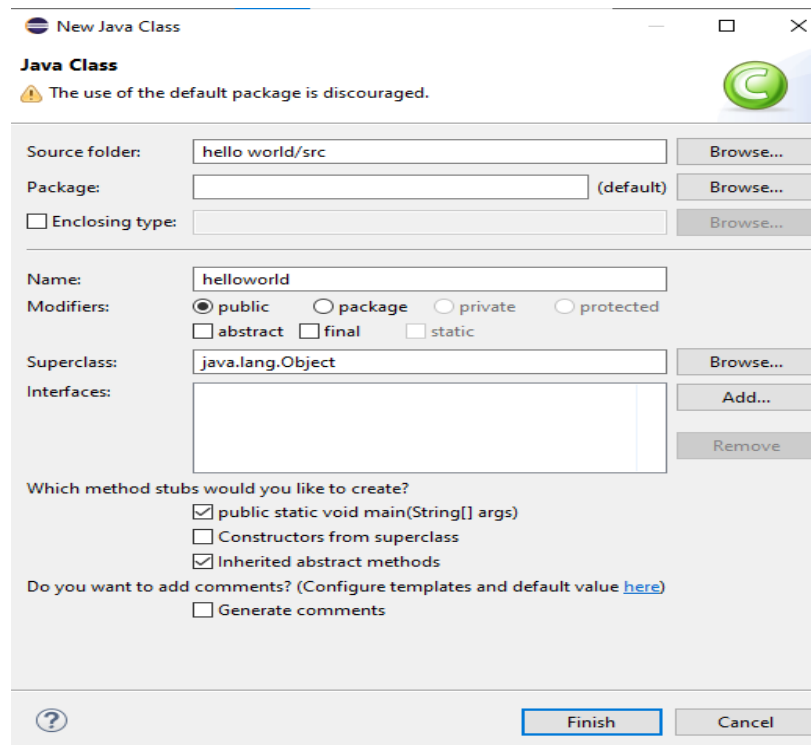
- Download the eclipse installer.



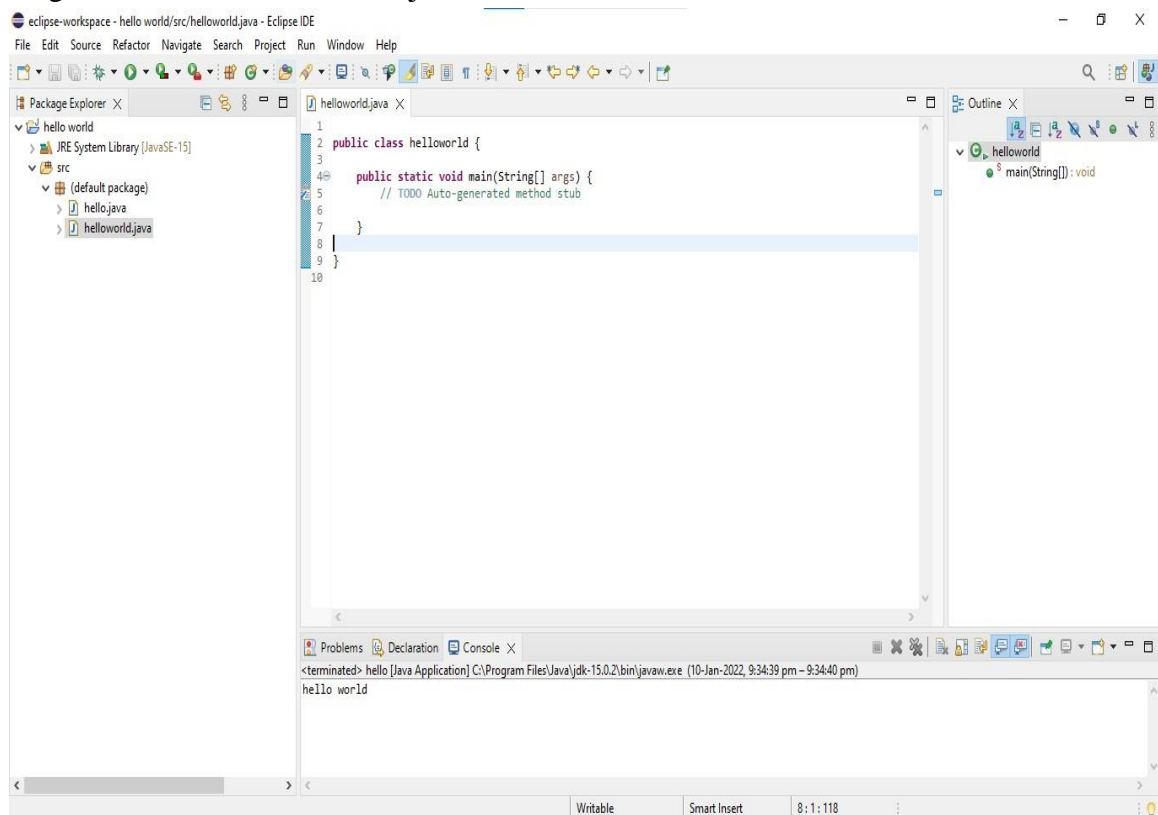
- Start the eclipse installer executable.
- Select the java developers package to install.
- Select your installation folder.
- Launch eclipse.



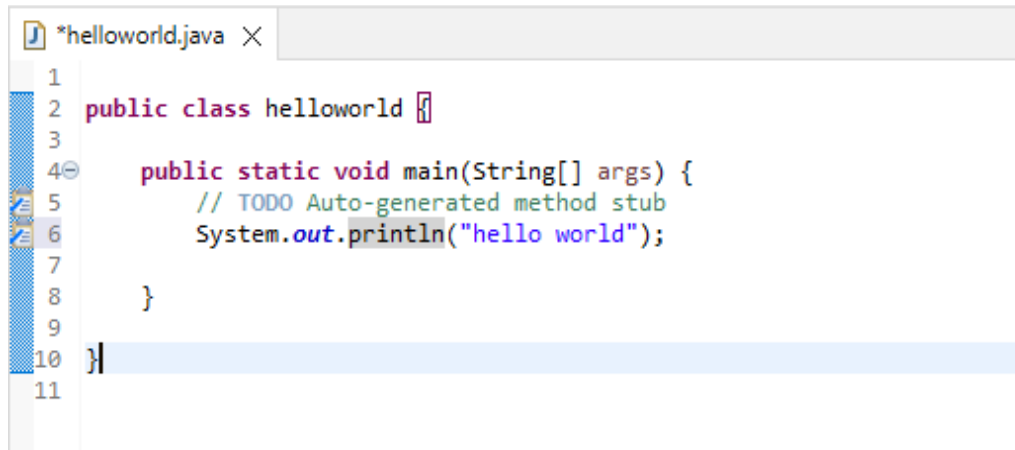
- Click on to the class and type here.
- The modifier is by default selected.
- Click the method public static void main(strings[],args)
- Click on finish.



- Right click on the 'helloworld.java'.

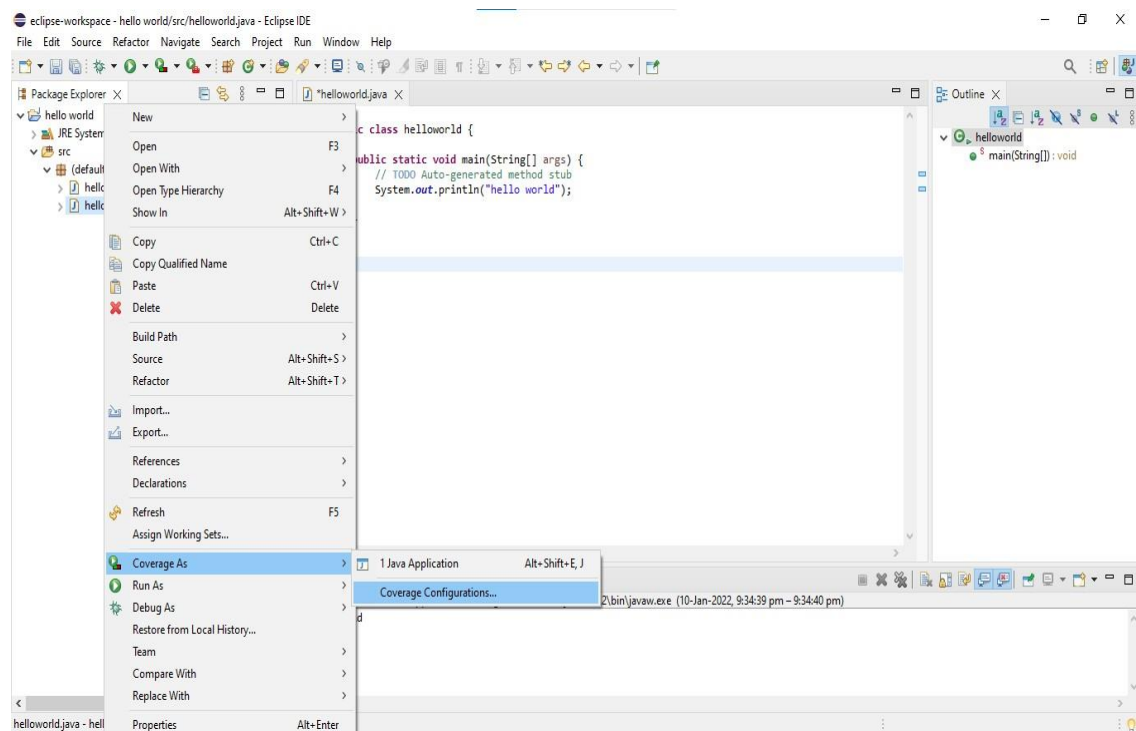


- For example enter the code.
- to print the hello world ,System.out.println("hello world")

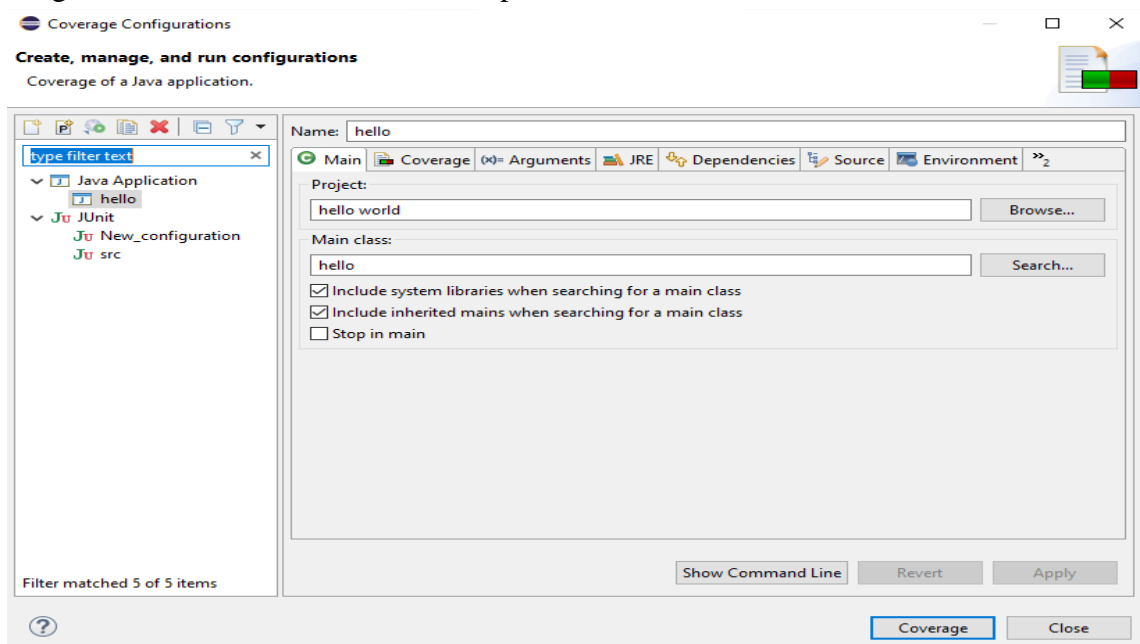


```
1 public class helloworld {
2
3
4 public static void main(String[] args) {
5 // TODO Auto-generated method stub
6 System.out.println('hello world');
7
8 }
9
10 }
11
```

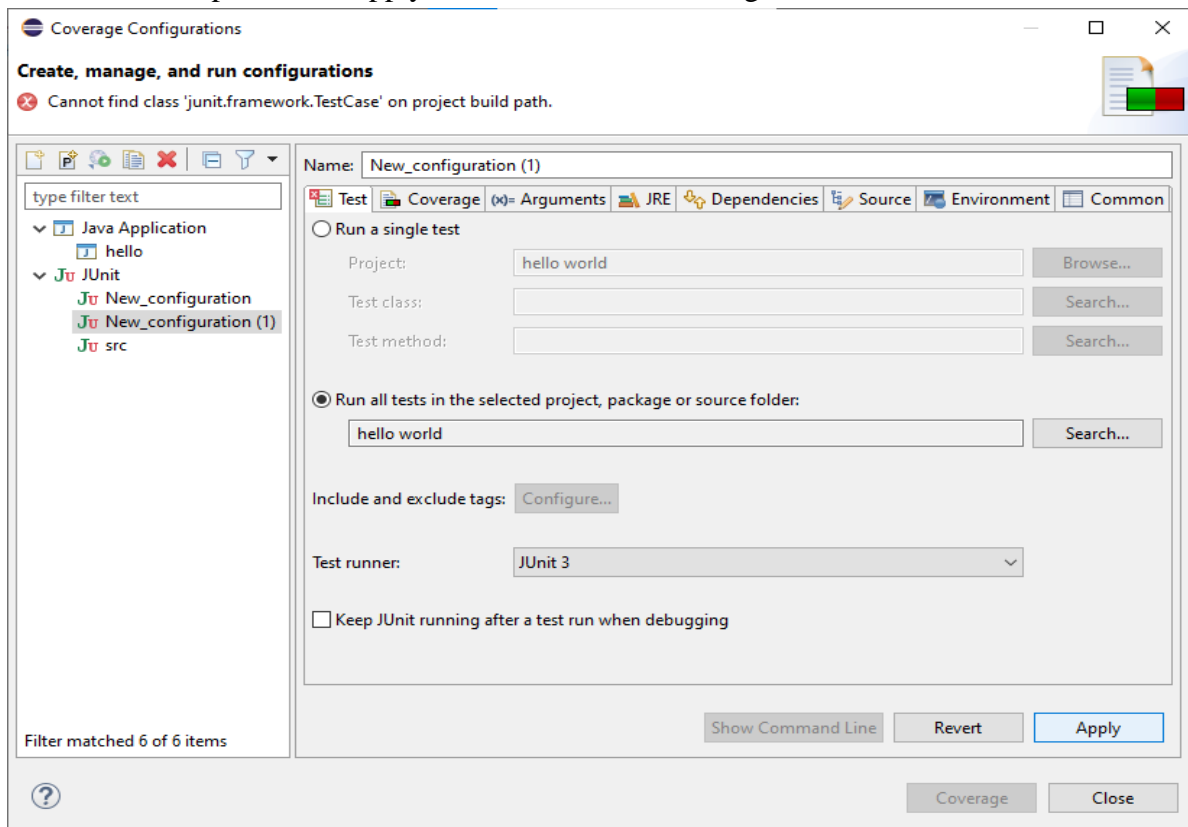
- Place the cursor on coverage as and click the coverage configurations.



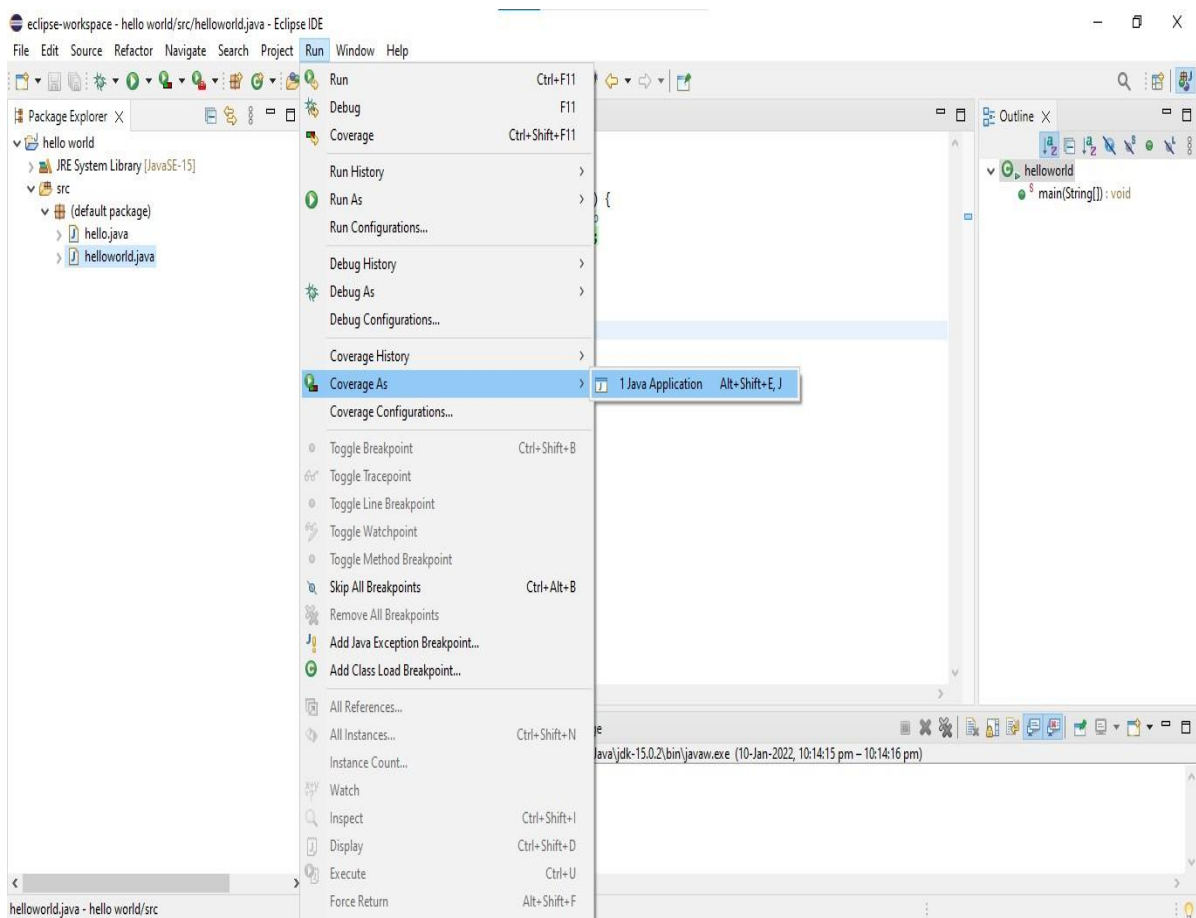
- Right click on the module to set the options.



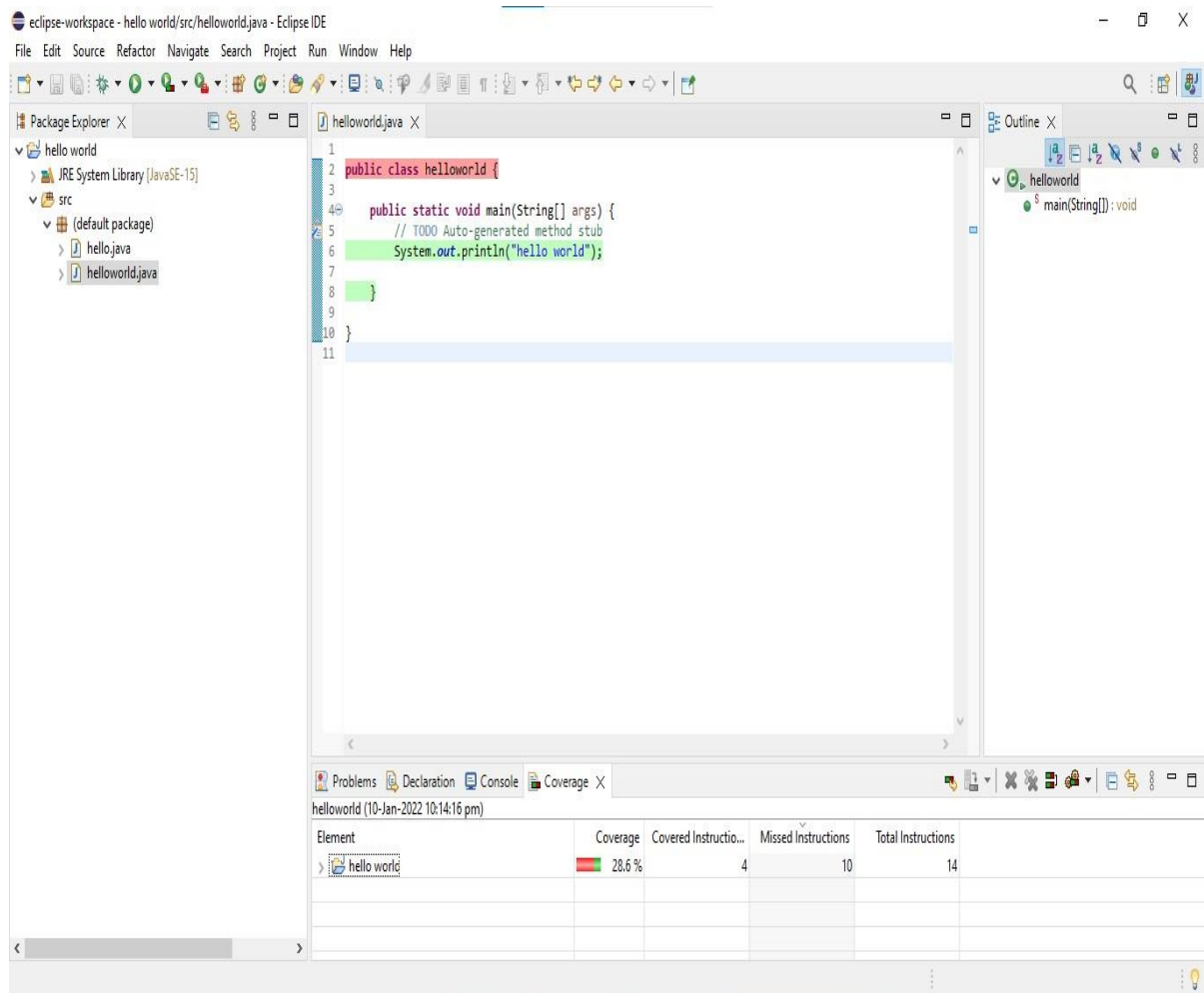
- Select all the options that apply and then click on coverage.



- Select coverage as option by right clicking on the file.



➤ Testing is done successfully .



RESULT:

Class level testing is done successfully and code coverage is as shown below sby using eclipse ide.

helloworld (10-Jan-2022 10:14:16 pm)				
Element	Coverage	Covered Instructio...	Missed Instructions	Total Instructions
> helloworld	28.6 %	4	10	14

EXP NO: 10**DATE:****AIM:**

To develop integration test cases from sequence diagram and perform integration testing using rational rose.

DESCRIPTION:**Integration test cases from sequence diagram:****Test Cases :**

Test can be defined as an action of exercising software with test cases, having as goals to find application faults or to demonstrate its correct execution . A test case is related with the software behavior and it has a set of inputs and a list of expect results. The structure of a test case can be defined as follows

Inputs:

- Precondition (initial condition) - initial state where the application must be, in order to start the test case execution;
- Input (step)- actions to be executed.

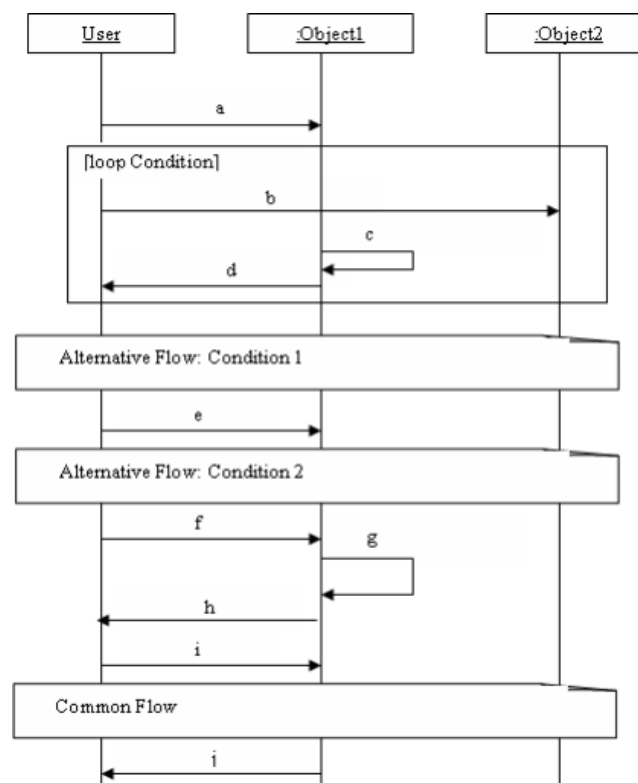
Expected results:

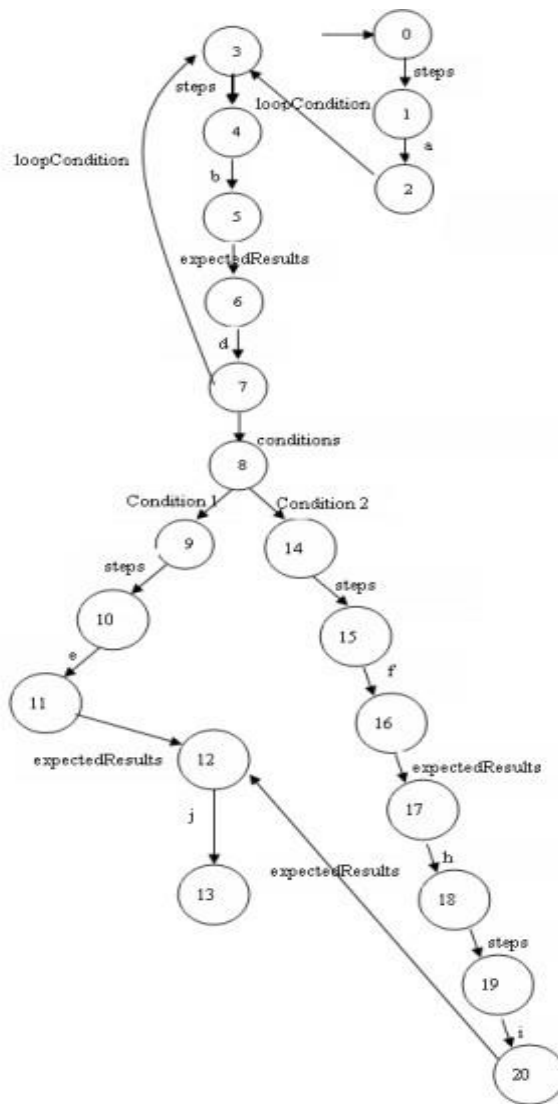
- Output - observed actions;
- Post condition - final state.

Sequence Diagram :

A sequence diagram describes a sequence of actions that occur in a system over time. It captures the invocation of methods from each object, and also the order in which it occurs. This makes sequence diagrams a very useful tool to easily represent the dynamic behavior of a system. In addition, sequence diagrams can be a good graphical representation of system scope requirements, since they can show the system behavior and the interfaces with other subsystems. A sequence diagram consists of:

Object: A sequence diagram consists of sequences of interaction among different objects.





Deriving LTS for testing from UML Sequence Diagram (PATH TABLE

Number	Path
1	steps, a, loopCondition, loop Condition, steps, b, expectedResults, d, loopCondition, conditions, Condition 1, steps, e, expectedResults, j
2	steps, a, loopCondition, loop Condition, steps, b, expectedResults, d, loopCondition, conditions, Condition 2, steps, f, expectedResults, h, steps, i, expectedResults, j

Initial Conditions	Condition 1
Steps	Expected Results
a	
Loop: loop Condition	
b	d
Loop	
e	j

(a)

Initial Conditions	Condition 2
Steps	Expected Results
a	
Loop: loop Condition	
b	d
Loop	
f	h
i	j

(b)

Test cases from path table

Integration testing:

The primary objective of integration testing is to test the module interfaces, i.e. there are no errors in the parameter passing, when one module invokes another module. There are four types of integration testing approaches. They are:

-Big bang approach:

It is the simplest integration testing approach, where all the modules making up a system are integrated in a single step.

-Bottom- up approach:

In bottom-up testing, each subsystem is tested separately and then the full system is tested.

-Top-down approach:

Top-down integration testing starts with the main routine and one or two subordinate routines in the system.

-Mixed-approach:

mixed (also called sandwiched) integration testing follows a combination of topdown and bottom-up testing approaches.

RESULT:

Integration test cases from sequence diagram have been successfully generated and performing integration testing by using rational rose.

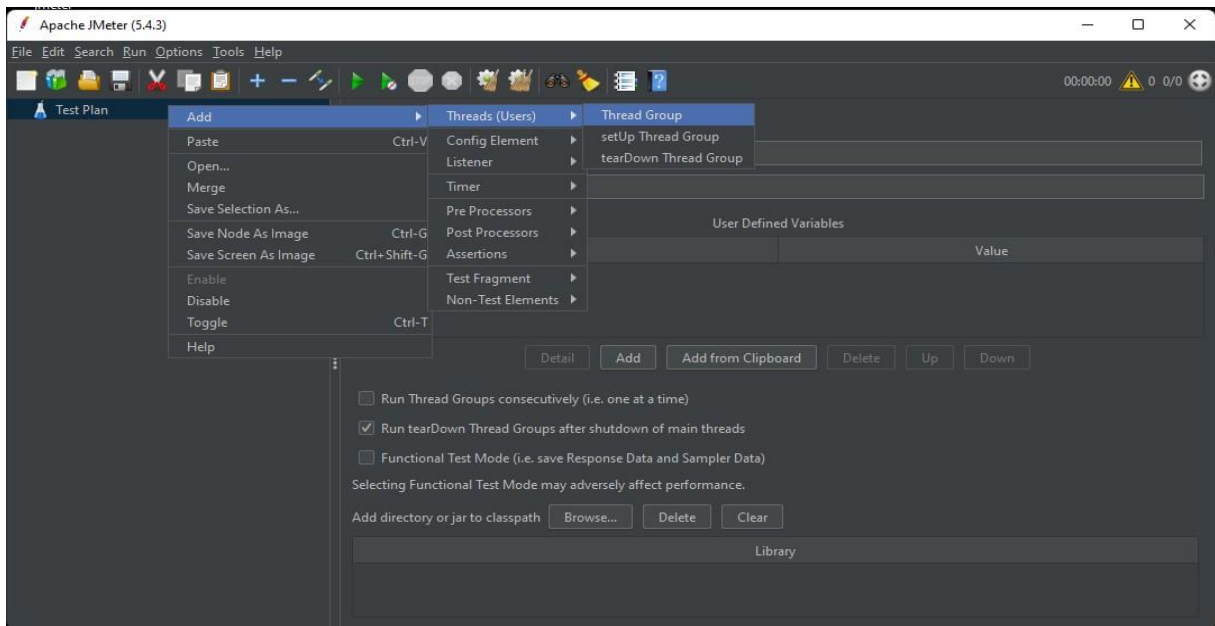
EXP NO: 11**DATE:****AIM:**

To perform performance testing using tool such as jmeter.

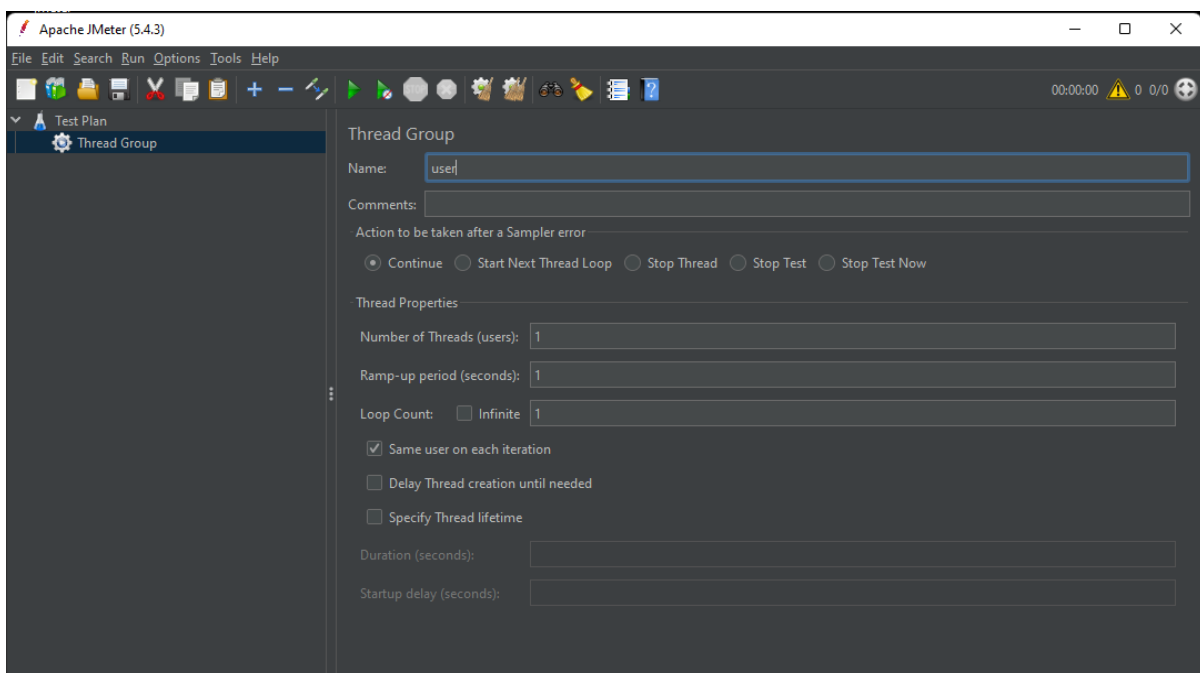
DESCRIPTION:

The **Apache JMeter™** application is open source software, a 100% pure Java application designed to load test functional behavior and measure performance. It was originally designed for testing Web Applications but has since expanded to other test functions. Follow the given below steps to set up and develop integration test cases from sequence diagram and perform integration testing:

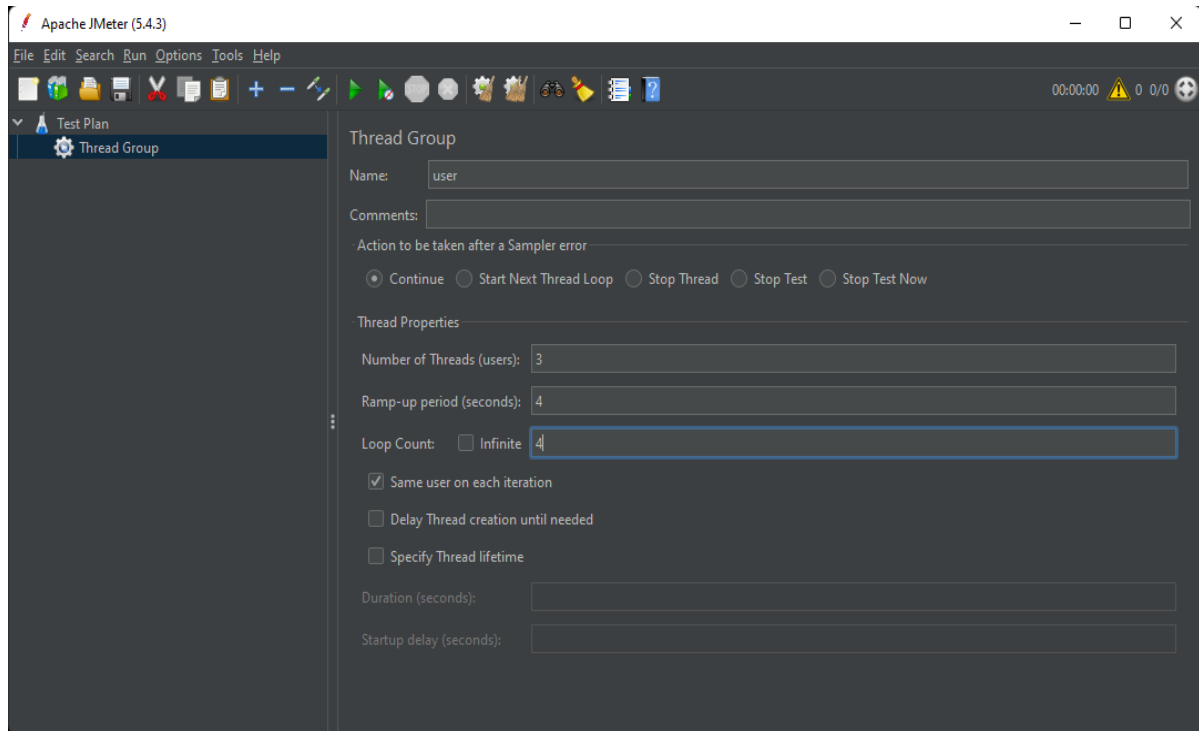
- Open Apache jmeter and click on test plan and then select thread group in thread(users).



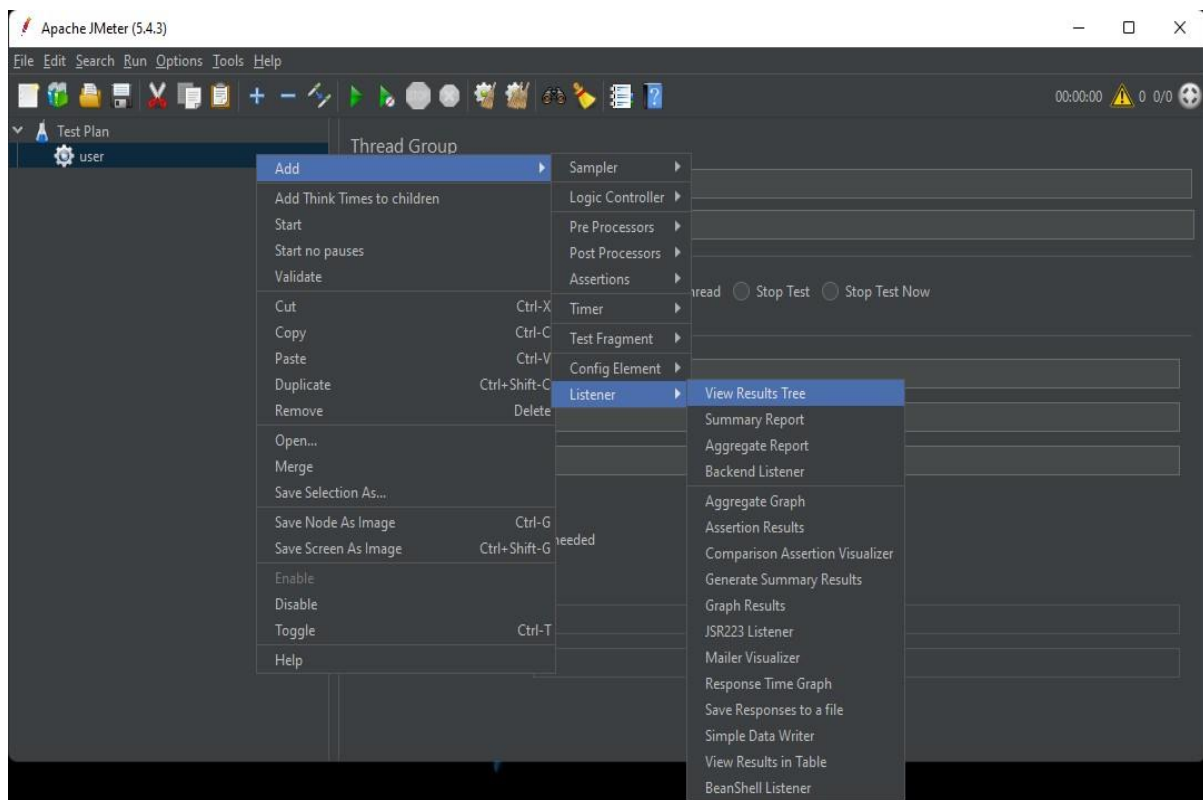
- Give it a name in thread group.



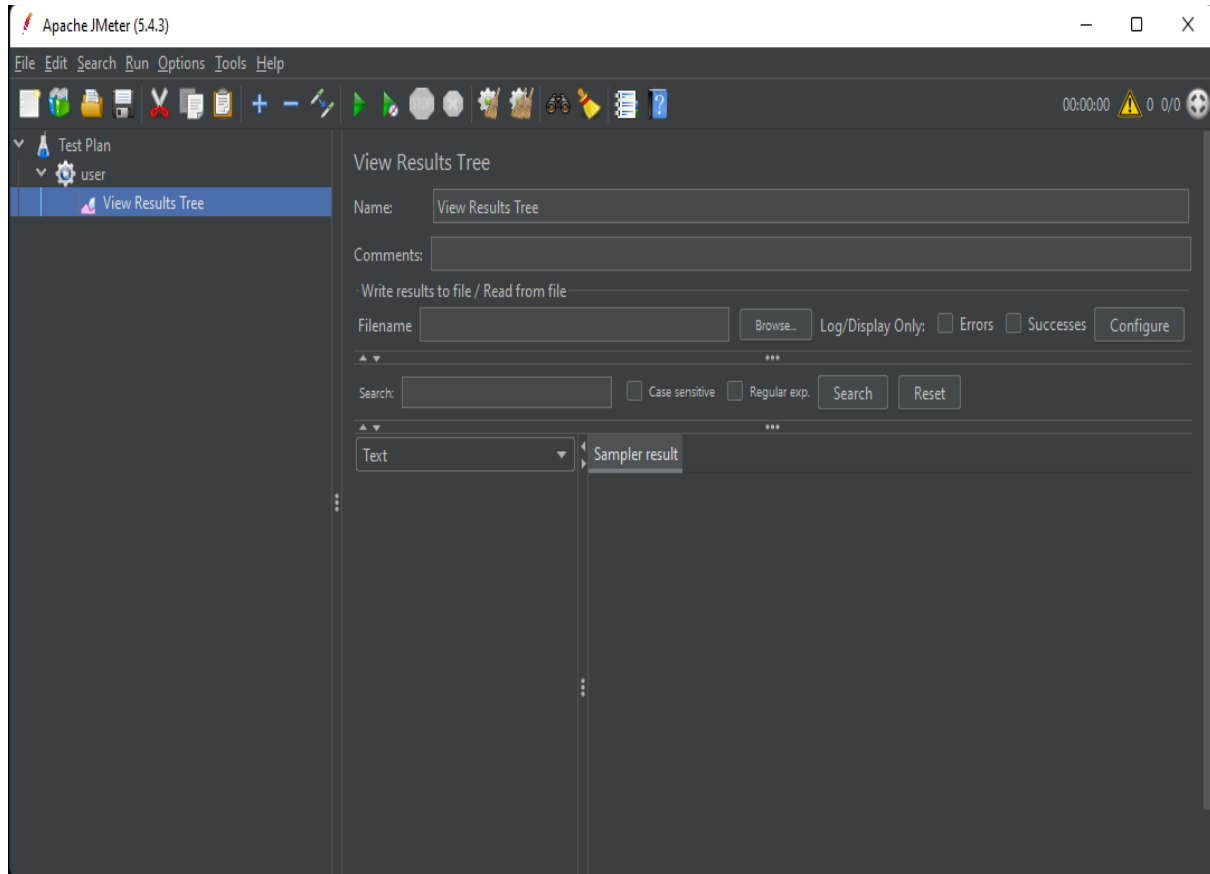
- Fill up number of threads, time period and count.



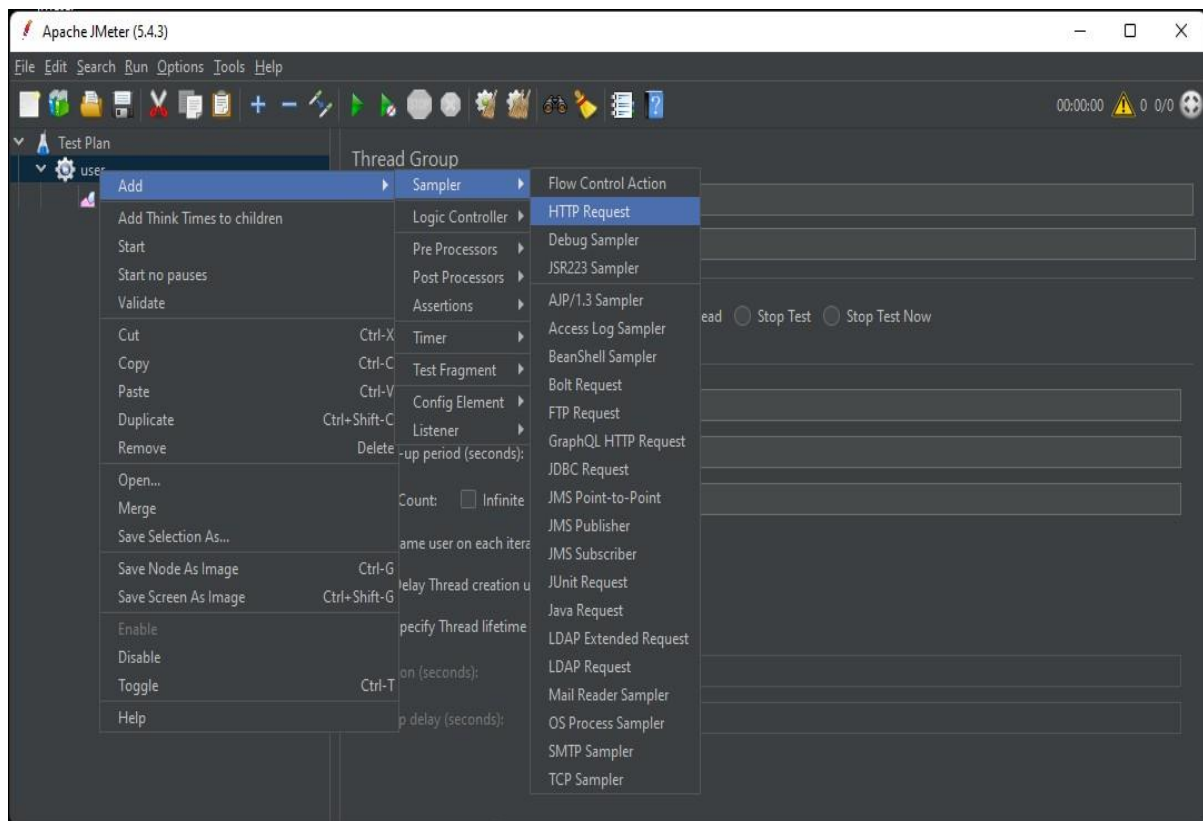
- Give a right click on the user you have created and then click on add then listener and then view results table.



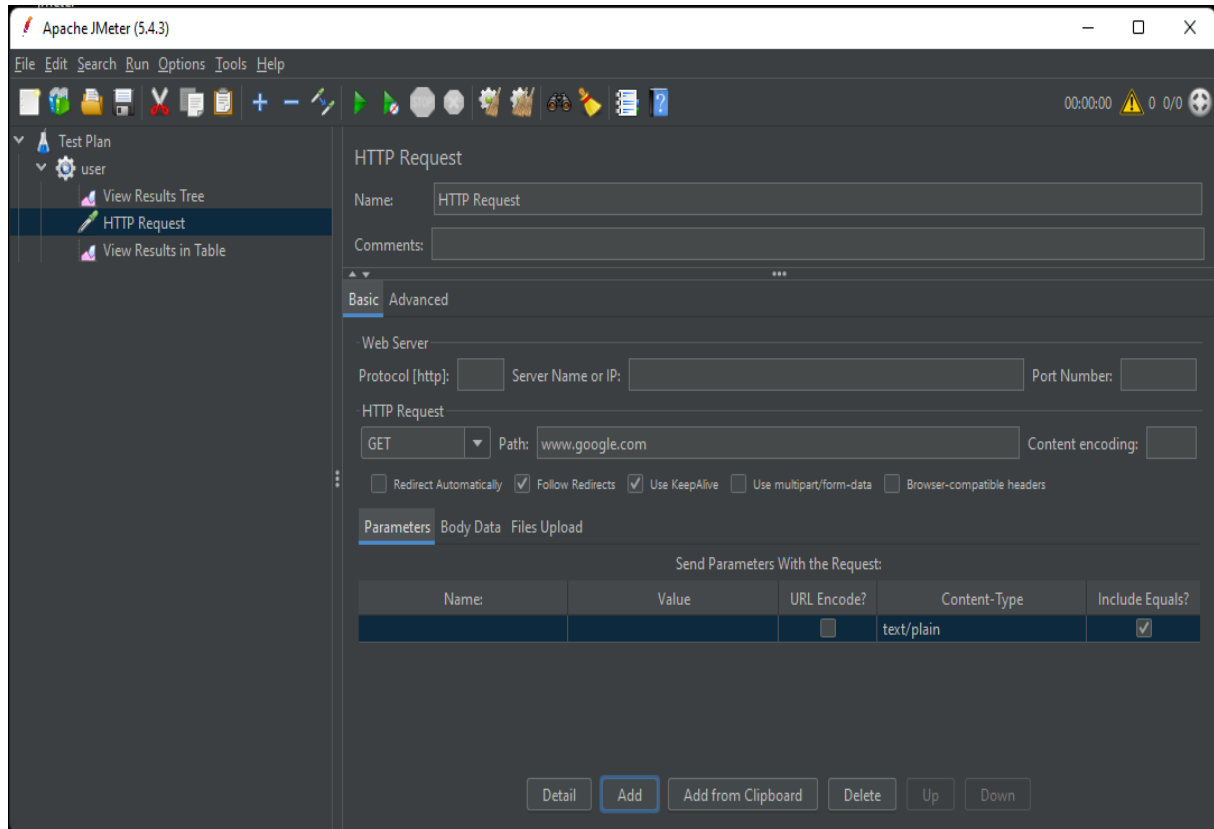
- The graphical view of results table is as shown below:



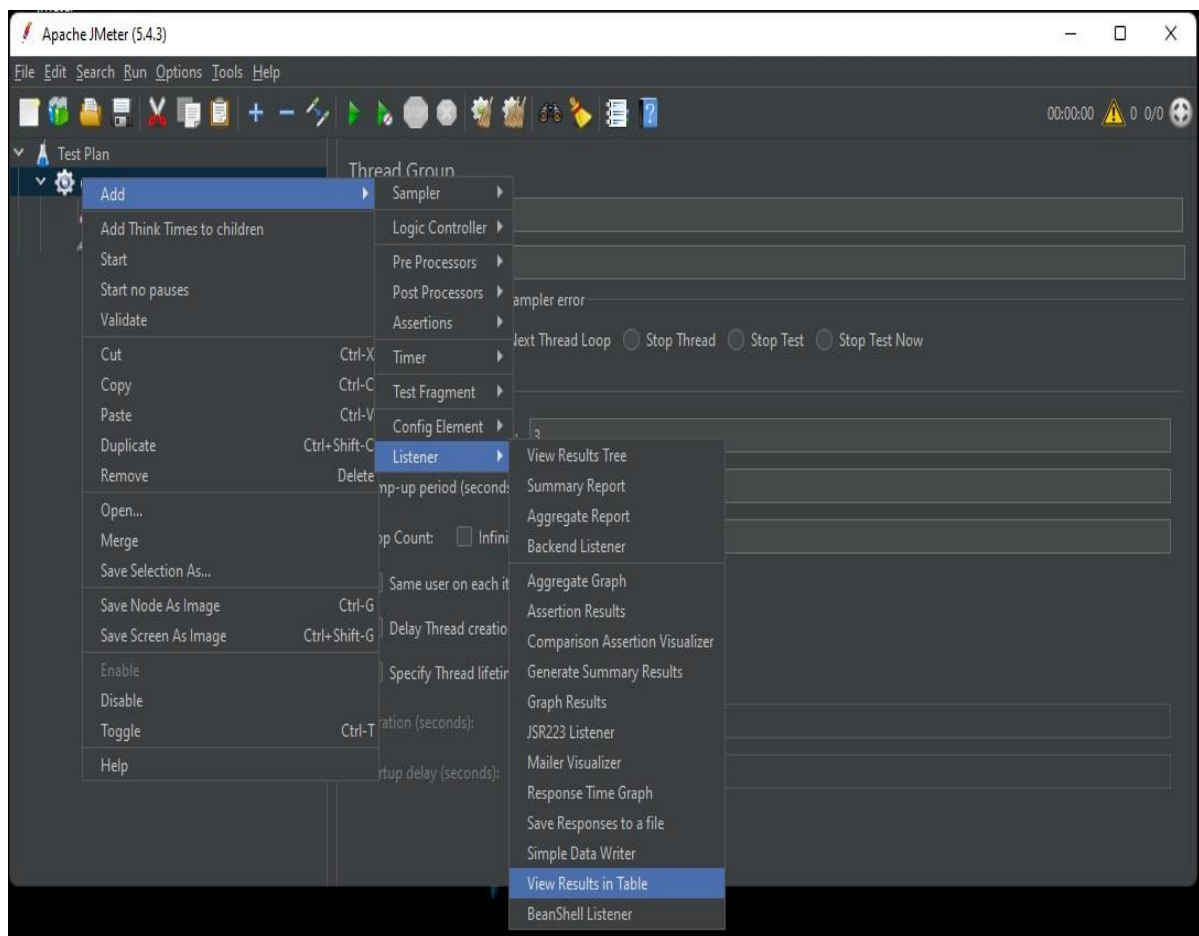
➤ Again give it a right click to add sampler and then click on HTTP Request.



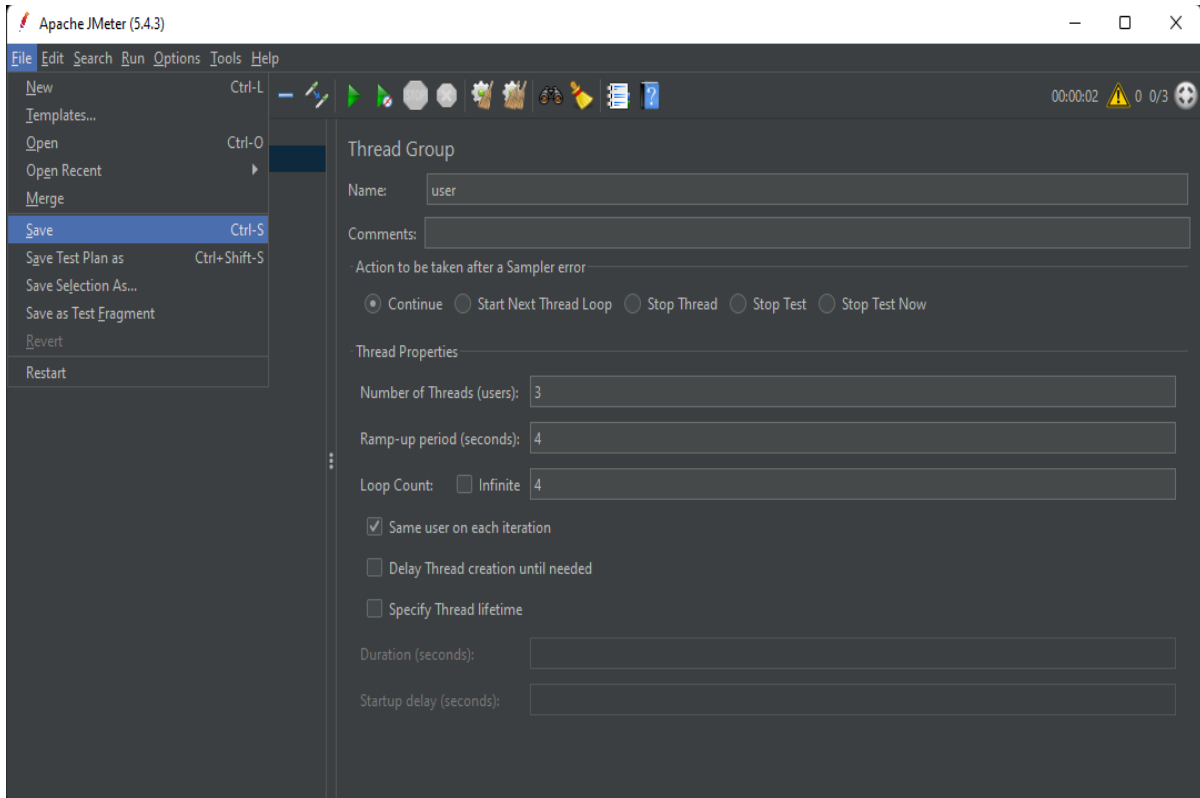
➤ The HTTP request page is as shown below:



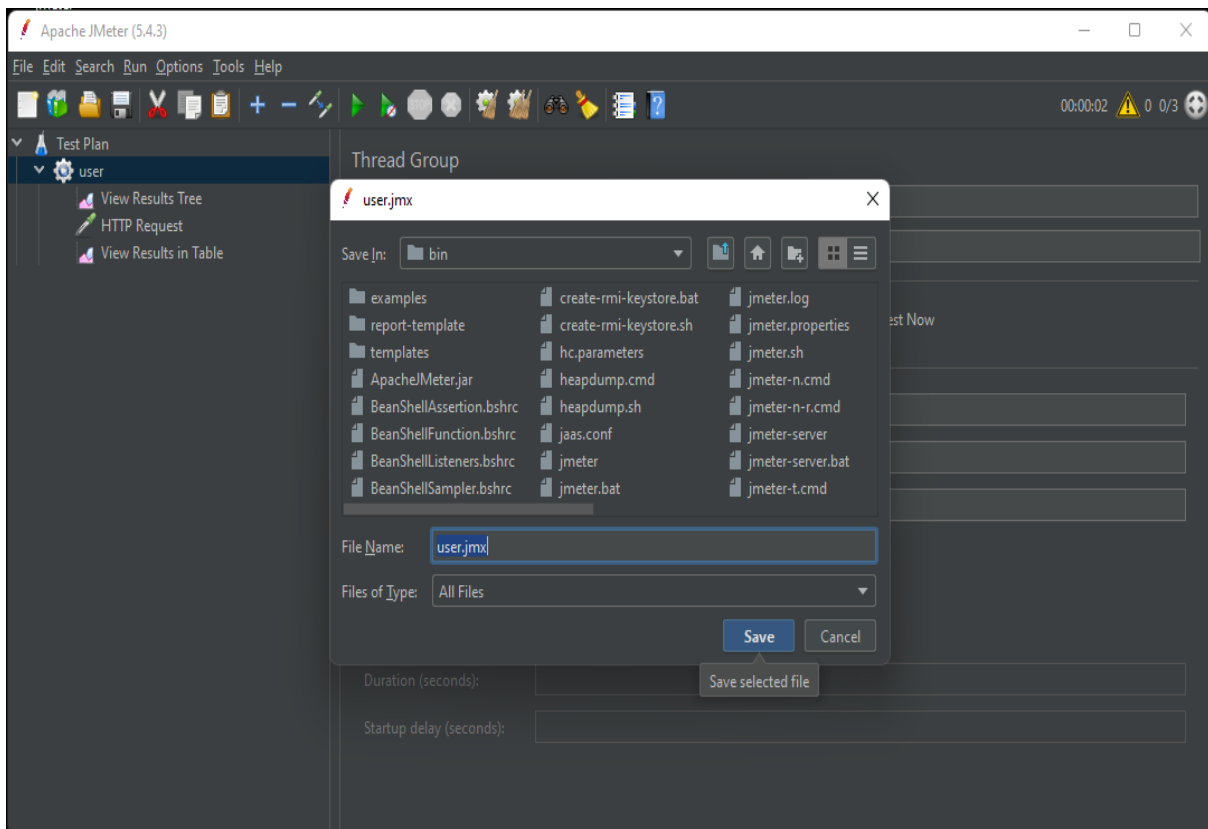
➤ Now add listener by clicking on add and then click on view results in table.



➤ Now save the file.



➤ Save the file with .jmx extension.



➤ Now click on 'view results in table' to see the results in a table form.

user.jmx (C:\apache-jmeter-5.4.3\bin\user.jmx) - Apache JMeter (5.4.3)

File Edit Search Run Options Tools Help

00:00:02 0 0/3

Test Plan

- Thread Group
 - HTTP Request
 - View Results Tree
 - View Results in Table**

View Results in Table

Name: View Results in Table

Comments:

Write results to file / Read from file

Filename Browse... Log/Display Only: ☐ Errors ☐ Successes

Sample #	Start Time	Thread Name	Label	Sample Time...	Status	Bytes	Sent Bytes	Latency
1	20:13:44.415	Thread Grou...	HTTP Request	33	✖	2291	0	0
2	20:13:44.449	Thread Grou...	HTTP Request	1	✖	2095	0	0
3	20:13:44.450	Thread Grou...	HTTP Request	0	✖	2095	0	0
4	20:13:44.451	Thread Grou...	HTTP Request	0	✖	2095	0	0
5	20:13:45.734	Thread Grou...	HTTP Request	0	✖	2095	0	0
6	20:13:45.735	Thread Grou...	HTTP Request	1	✖	2095	0	0
7	20:13:45.736	Thread Grou...	HTTP Request	0	✖	2095	0	0
8	20:13:45.737	Thread Grou...	HTTP Request	0	✖	2095	0	0
9	20:13:47.081	Thread Grou...	HTTP Request	1	✖	2095	0	0
10	20:13:47.082	Thread Grou...	HTTP Request	0	✖	2095	0	0
11	20:13:47.082	Thread Grou...	HTTP Request	1	✖	2095	0	0
12	20:13:47.083	Thread Grou...	HTTP Request	0	✖	2095	0	0

➤ You can also view results in a tree form by clicking on 'view results tree'.

user.jmx (C:\apache-jmeter-5.4.3\bin\user.jmx) - Apache JMeter (5.4.3)

File Edit Search Run Options Tools Help

00:00:02 0 0/3

Test Plan

- Thread Group
 - HTTP Request
 - View Results Tree**
 - View Results in Table

View Results Tree

Name: View Results Tree

Comments:

Write results to file / Read from file

Filename Browse... Log/Display Only: ☐ Errors ☐ Successes

Search: ☐ Case sensitive ☐ Regular exp.

Text

- ✖ HTTP Request
- ✖ HTTP Request
- ✖ HTTP Request
- ✖ HTTP Request
- ✖ HTTP Request
- ✖ HTTP Request
- ✖ HTTP Request
- ✖ HTTP Request
- ✖ HTTP Request
- ✖ HTTP Request
- ✖ HTTP Request

RESULT:

Performance testing is performed successfully by using **Apache jmeter**.