Experiment 2

Title:Generate Junit Test cases

Procedure:

Step1. Click on file menu and select project

Step2. Right click on project and create class (eg circle)

Step3. Declare a data member (eg float radius)

Step4. Right click on class *ie circle* →go to source→click on generate getters and setters

Step5. Create a function *ie area(implement the function)*

Step6. Right click on package →click new→click on Junit

Step7. Write the class name for Junit testcase and browse the class *ie circle.*

Step8. Click next button right down the window

Step9. Select all the methods on which you want write the testcase *ie setradius() and area()*

Step10. Write following statements

Create instance of class which function are to be tested

```
package circle;

import junit.framework.TestCase;

public class temp extends TestCase {

    public void testSetRadius() {
        //fail("Not yet implemented");
        circle circle1=new circle();
        float radius=4;

        circle1.setRadius(radius);
        assertEquals(radius,circle1.getRadius());


    }

}
```

Step 11. Run testcase class ie Right click on testcase class and click on Run as →Junit

Step12. Repeat step 2 to 10 to create new class and testcase for same.

Step13. To run all testcases together we create suite.

Right click on package →new→other→ Java→ Junit→Junit Suite

Browse the package name in which classes are created

Select all the testcase classes need to be run

Click Finish

Right Click on suite class (AllTest) → Run it as Junit

Experiment number 3

Title: Debugging the code

Procedure:

Step 1: declare the variables such as name,rollno,marks,percentage.

Step 2:generate the getters and setters for each of these variables

Step 3:calculate the percentage based on marks.

Step 4:generate thejunit test case to test the percentage

Step 5:on the main program put the toggle breakpoint on a particular statement

Step 6:debug as java applicarion and then click on step into to see each statement output or press F5 and then F6 to skip the final step

Expeiment No: 4

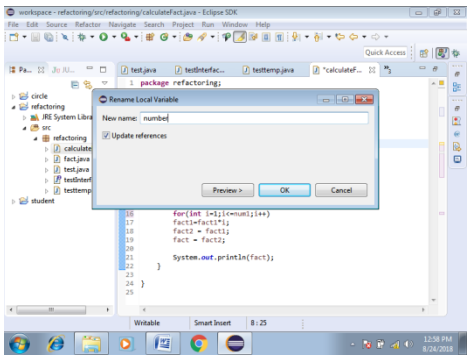
Aim: To use refactoring methods.

Procedure:

1:Rename-

Select variable/class name/method name

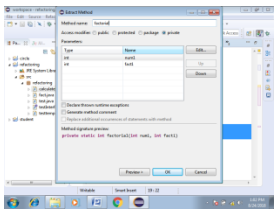
Right click →refactor→rename.



2:Extract Method-

Select **relevant** code

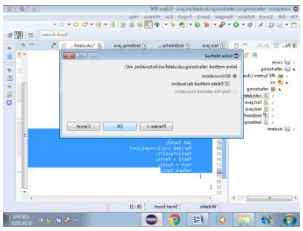
Right click →refactor→extract method.



3:Inline-

Select relevant code

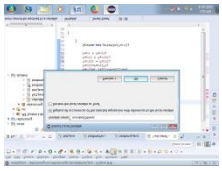
Right click →refactor→inline.



4:Extract Local Variable-

Select relevant code

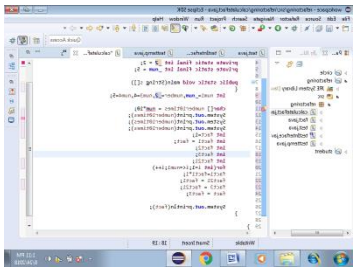
Right click →refactor→extract local variable.



5:Extract Constants-

Select relevant code

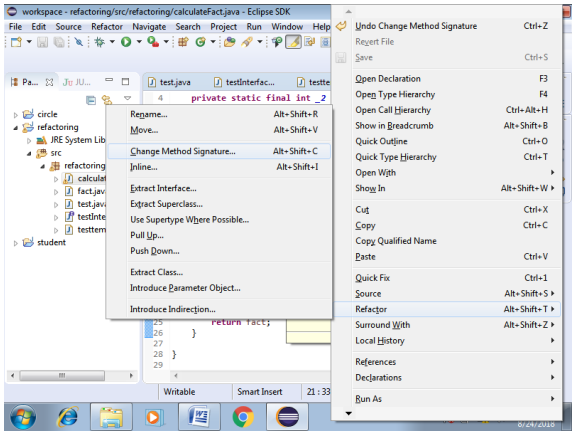
Right click →refactor→extract constants



6:Change method signature-

Select method name

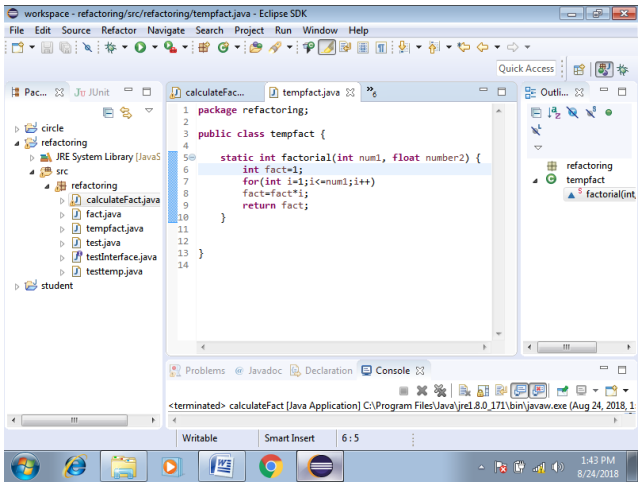
Right click →refractor→change method signature→add



7: Move method-

Select the entire method

Right click →refractor→move→click on browse→choose the new class→ok



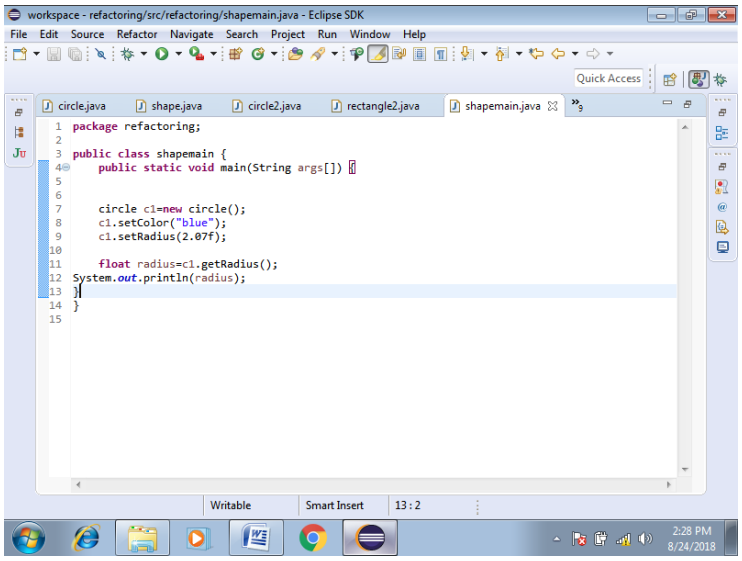
8: Extract Superclass-

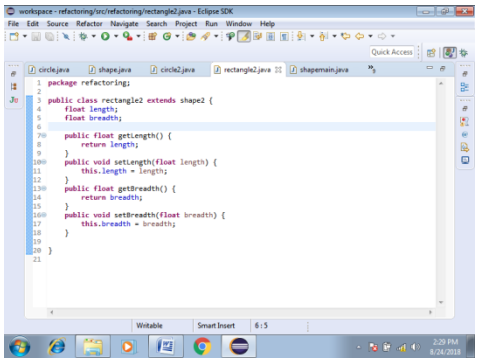
Create 2 classes: circle and rectangle.

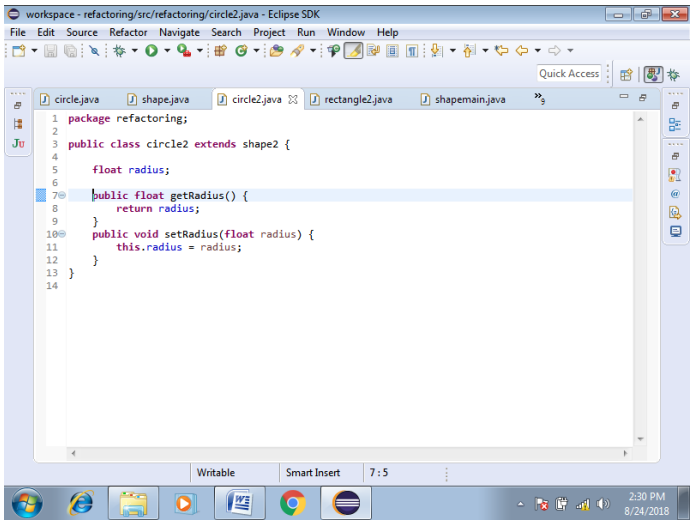
Circle with data members color and radius

Rectangle with data members length, breadth and color

Right click →refractor→extractSuperClass→click on Add (choose appropriate class)→Click on add and select parameters→ok







Right click on

Circle :

package refactoring;

```
public class circle extends shape {
	float radius;

	public float getRadius() {
		return radius;
	}
	public void setRadius(float radius) {
		this.radius = radius;
	}

}
```

Rectangle:

```
package refactoring;

public class rectangle extends shape {
	float breadth;
	float length;
	public float getBreadth() {
		return breadth;
	}
	public void setBreadth(float breadth) {
		this.breadth = breadth;
	}
	public float getLength() {
		return length;
	}
	public void setLength(float length) {
		this.length = length;
	}

}
```

Shape:

```
package refactoring;

public class shape implements shapeinterface {

	protected String color;

	public shape() {
		super();
	}
```

```
	@Override
	public String getColor() {
		return color;
	}

	@Override
	public void setColor(String color) {
		this.color = color;
	}

}
```

Experiment 5

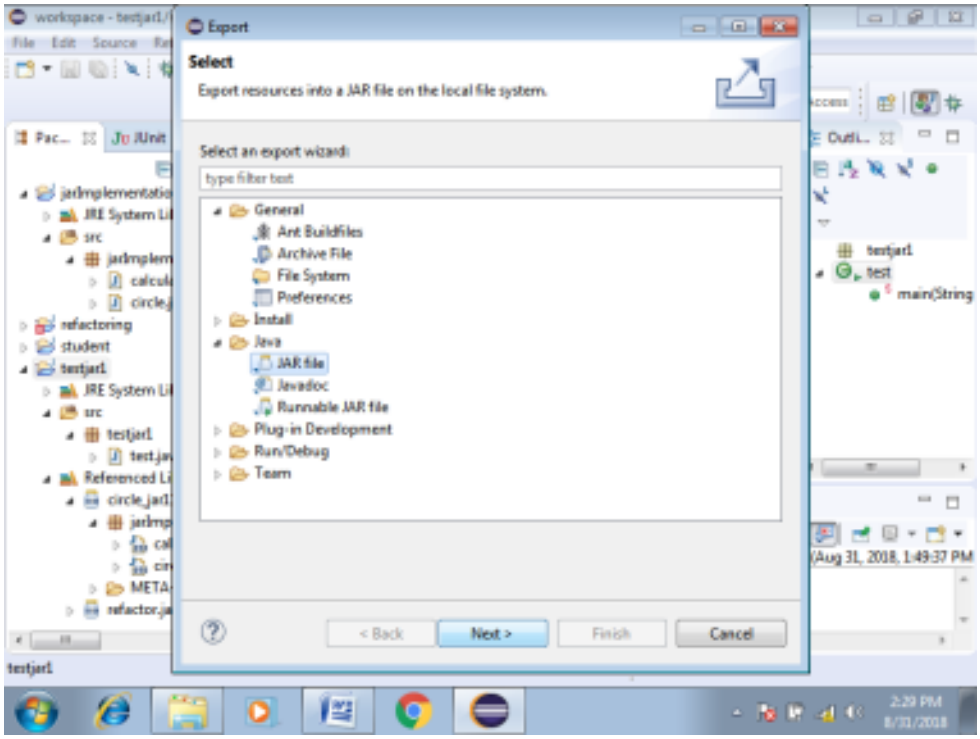
Aim: Extracting *jar* file

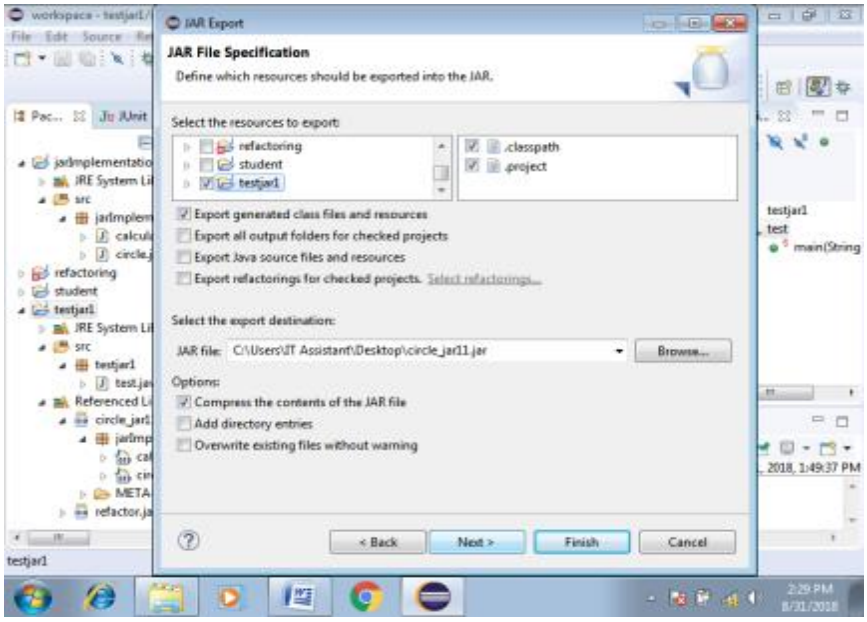
Procedure:

Step 1: Create a project testjar

Step 2: Create 2files in it i.e., circle file

Step3: Right click on the project and →export→java→jarfile

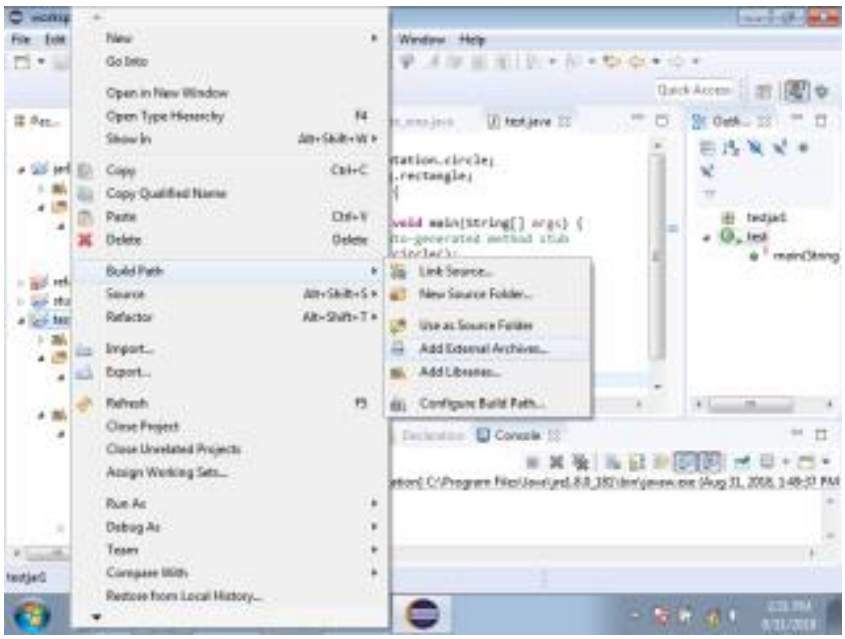




To import

Step4:create a new project

Step5:right click on project→buildpath→general→add external archives



Code:

```
package testjar1;
import jarImplementation.circle;
```

```
import refactoring.rectangle;
public class test {

        public static void main(String[] args) {
                // TODO Auto-generated method stub
    circle c=new circle();
    c.setRadius(3f);
    System.out.println(c.area());
    rectangle r=new rectangle();
    r.setColor("red");

    System.out.println(r.getColor());
        }

}
```

Experiment 6

Aim: Generating a Java documentation

Procedure:

Step 1: Select your project and click on project on menu bar and select Generate

Step 2: Click on project name and give destination path for saving javadoc and click finish

You can view the documentation as shown below.

Experiment no 7

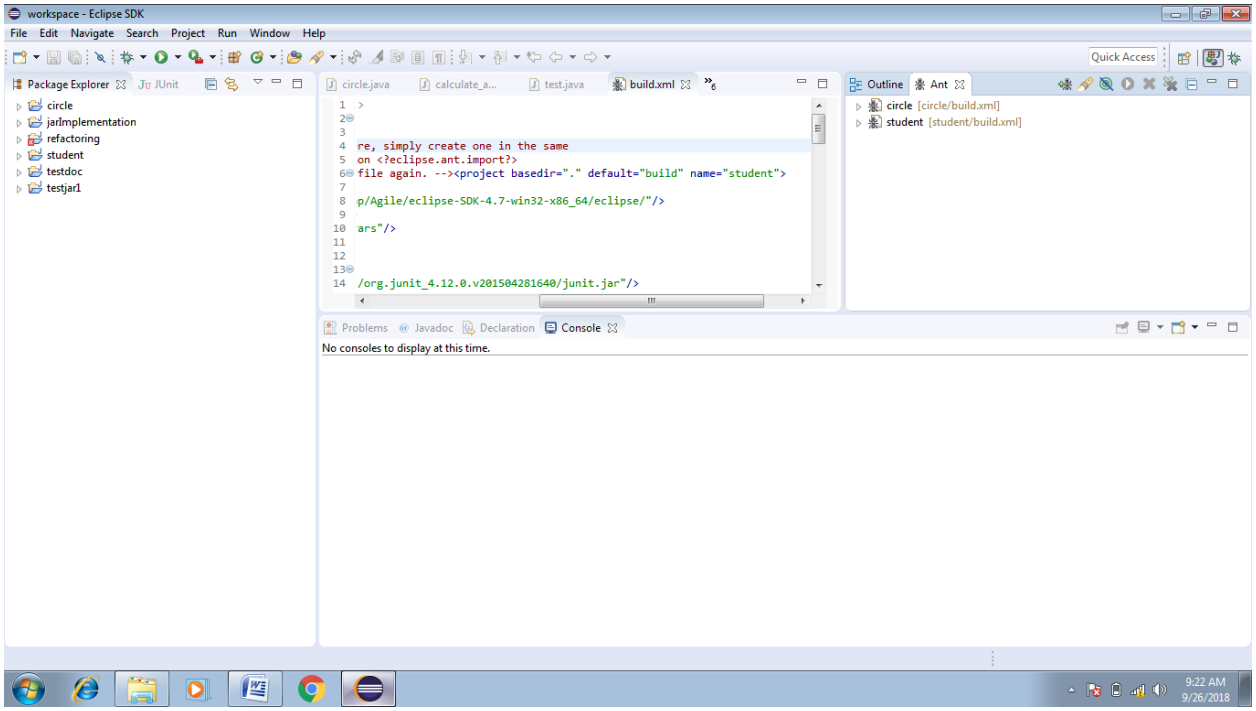
Title: Ant build tools

Procedure:

Step 1: To create build.xml file. click on File→Export→AntbuildFiles→Next

Step 2: To run the build.xml file. Click on windows → Show view→ ant

Step 3: Drag the build.xml file in the ant window as shown below



Step 4: Right click on build project and click on run

workspace - student/build.xml - Eclipse SDK

File Edit Navigate Search Project Run Window Help

Quick Access

Package Explorer | JUnit

- circle
- jarImplementation
- refactoring
- student
- testdoc
- testjar1

circle.java | calculate_a... | test.java | build.xml

```
1  <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2  <!-- WARNING: Eclipse auto-generated file.
3               Any modifications will be overwritten.
4               To include a user specific buildfile here, simply create
5               directory with the processing instruction <?eclipse.ant.i
6               as the first entry and export the buildfile again. --><pr
7      <property environment="env"/>
8      <property name="ECLIPSE_HOME" value="../../Desktop/Agile/eclipse-SD
9      <property name="junit.output.dir" value="junit"/>
10     <property name="debuglevel" value="source,lines,vars"/>
11     <property name="target" value="1.8"/>
12     <property name="source" value="1.8"/>
13     <path id="JUnit 3.libraryclasspath">
14         <pathelement location="${ECLIPSE_HOME}plugins/org.junit_4.12.0.
```

Outline | Ant

- circle [circle/build.xml]
  - AllTests
  - build [default]
  - build-eclipse-compiler
  - build-project
  - build-refprojects
  - build-subprojects
  - clean
  - cleanall
  - init
  - init-eclipse-compiler
  - junitreport
  - temp

Problems | Javadoc | Declaration | Console

```
<terminated> circle build.xml [Ant Build] C:\Program Files\Java\jdk1.8.0_181\bin\javaw.exe (Sep 26, 2018, 9:35:05 AM)
Buildfile: C:\Users\IT Assistant\workspace\circle\build.xml
build-subprojects:
init:
build-project:
     [echo] circle: C:\Users\IT Assistant\workspace\circle\build.xml
build:
BUILD SUCCESSFUL
Total time: 2 seconds
```

Writable | Insert | 4 : 87

9:35 AM
9/26/2018