Computer Graphics

Instructor: MSc. Trần Khai Minh
Email: tkminh@hcmiu.edu.vn

Lab 1 - Introduction to Computer Graphics

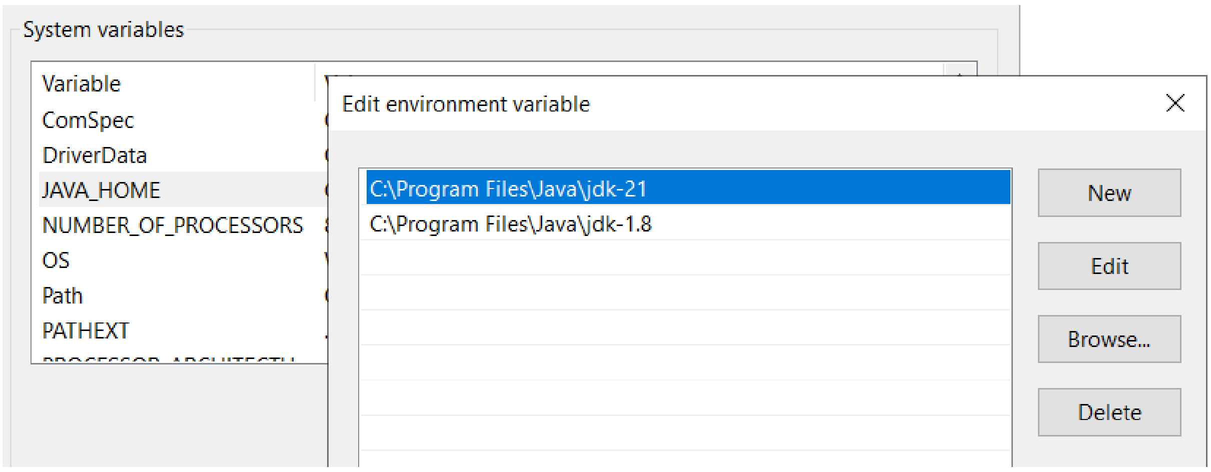
Your name:
ID:

Contents:
- **Create environment for Java graphic programming.**
- **Explore the Java graphic Libraries in Netbeans**
- **Install JOGL**
- **Practices and exercises**

Duration: 3 hours

Part 1: Set up the environment for Java graphic programming.

A. At first, you should install JDK (the latest version).



C:\Program Files\Java\jdk-21
C:\Program Files\Java\jdk-1.8

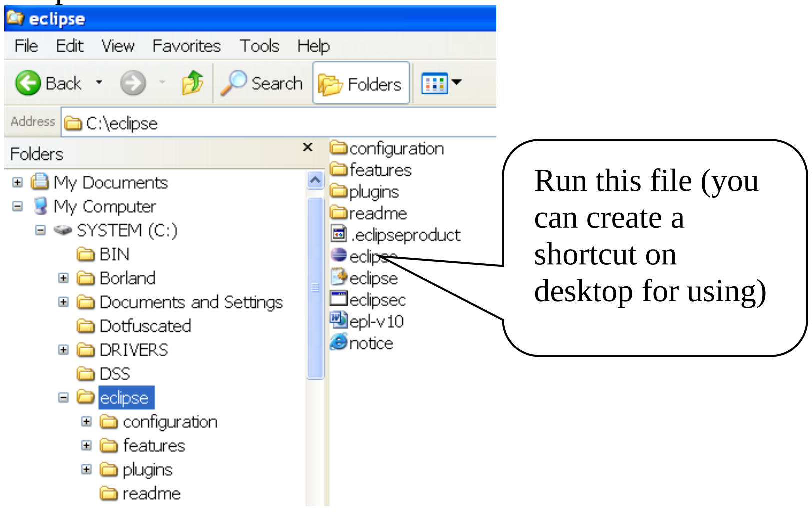
Platform	**Command**
Windows	Set the environment variable JAVA_HOME to C:\ProgramFiles\Java\jdk-21
Linux	Export JAVA_HOME=/usr/local/java-current
MAC	Export JAVA_HOME=/Library/Java/Home

Append Java compiler location to System Path as follows –

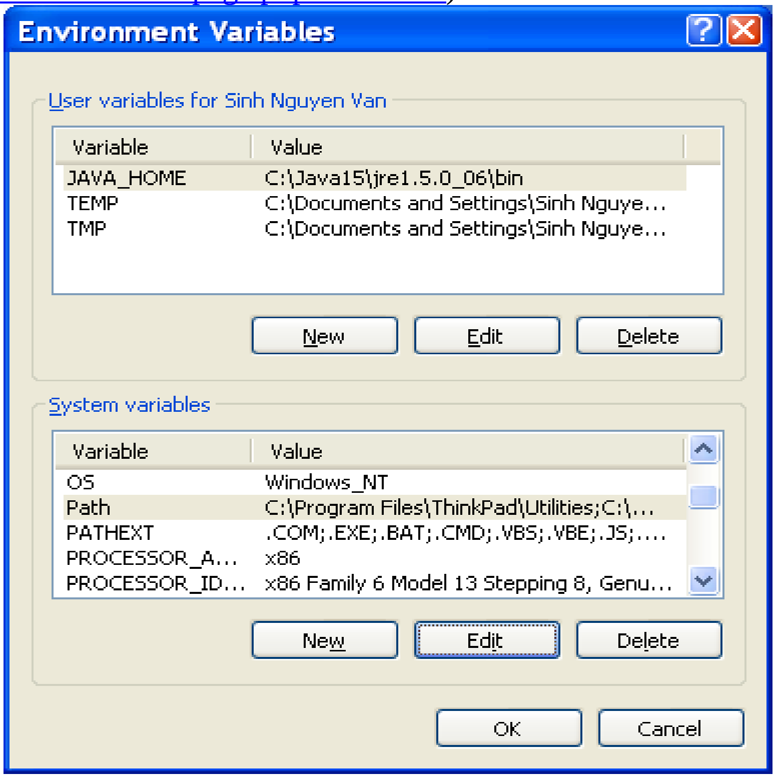
Platform	**Command**
Windows	Append the string ;%JAVA_HOME% bin at the end of the system variable and path
Linux	Export PATH=$PATH:$JAVA_HOME/bin/
MAC	Not required

B. Eclipse (optional):

- Download the Eclipse from website: http://www.eclipse.org/downloads/
- Extract file .rar to c:\eclipse



- Set up java JDK.
- Set up the environment for working by using command line: (https://kb.wisc.edu/cae/page.php?id=24500).



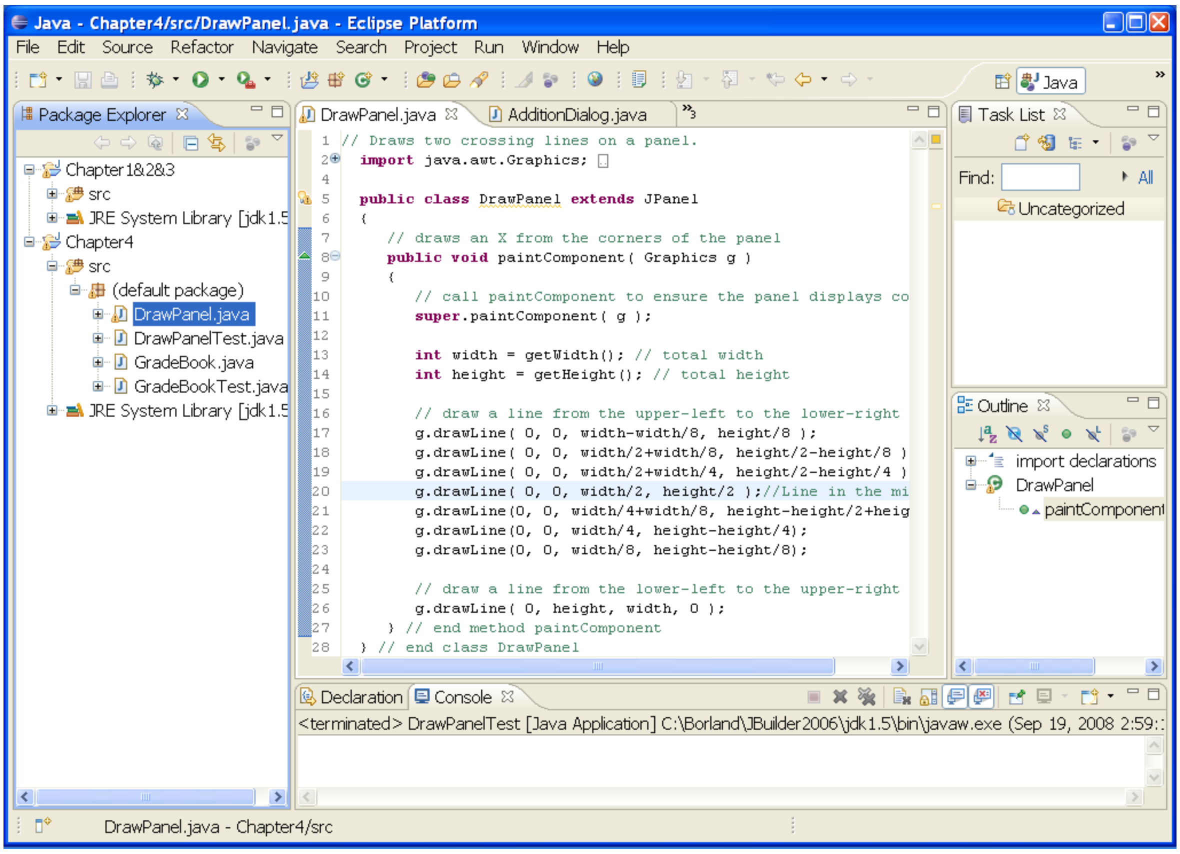
User variables for Sinh Nguyen Van

Variable	Value
JAVA_HOME	C:\Java15\jre1.5.0_06\bin
TEMP	C:\Documents and Settings\Sinh Nguye...
TMP	C:\Documents and Settings\Sinh Nguye...

System variables

Variable	Value
OS	Windows_NT
Path	C:\Program Files\ThinkPad\Utilities;C:\...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;....
PROCESSOR_A...	x86
PROCESSOR_ID...	x86 Family 6 Model 13 Stepping 8, Genu...

- The interface of Eclipse:



```
// Draws two crossing lines on a panel.
  import java.awt.Graphics; 

  public class DrawPanel extends JPanel
  {
     // draws an X from the corners of the panel
     public void paintComponent( Graphics g )
     {
        // call paintComponent to ensure the panel displays co
        super.paintComponent( g );

        int width = getWidth(); // total width
        int height = getHeight(); // total height

        // draw a line from the upper-left to the lower-right
        g.drawLine( 0, 0, width-width/8, height/8 );
        g.drawLine( 0, 0, width/2+width/8, height/2-height/8 )
        g.drawLine( 0, 0, width/2+width/4, height/2-height/4 )
        g.drawLine( 0, 0, width/2, height/2 );//Line in the mi
        g.drawLine(0, 0, width/4+width/8, height-height/2+heig
        g.drawLine(0, 0, width/4, height-height/4);
        g.drawLine(0, 0, width/8, height-height/8);

        // draw a line from the lower-left to the upper-right
        g.drawLine( 0, height, width, 0 );
     } // end method paintComponent
  } // end class DrawPanel
```

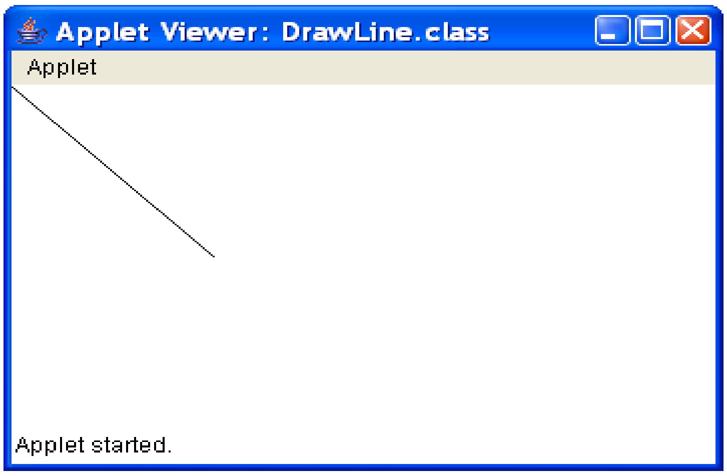
```
<terminated> DrawPanelTest [Java Application] C:\Borland\JBuilder2006\jdk1.5\bin\javaw.exe (Sep 19, 2008 2:59:
```

- Create a new project and check the system by testing a java file **DrawLine.java** as below:

```
import java.applet.Applet;
import java.awt.*;
/**
 * @author Sinh Nguyen Van //username during the installation
 */
public class DrawLine extends Applet {
    public void paint(Graphics g){
        g.drawLine(0, 0, 100, 100);
    }
}
```

```
import javax.swing.*;
import java.awt.*;
public class DrawLine extends JPanel {
    @Override
    public void paint(Graphics g) {

        // Draw the division line from top right to bottom left
        g.drawLine(0, 0, 100, 100);
}
}
```

This is the result:



The definition of drawLine function

```
public abstract void drawLine(int x1,
            int y1,
            int x2,
            int y2)
```

Draws a line, using the current color, between the points (x1, y1) and (x2, y2) in this graphics context's coordinate system.

Parameters:

x1 - the first point's *x* coordinate.

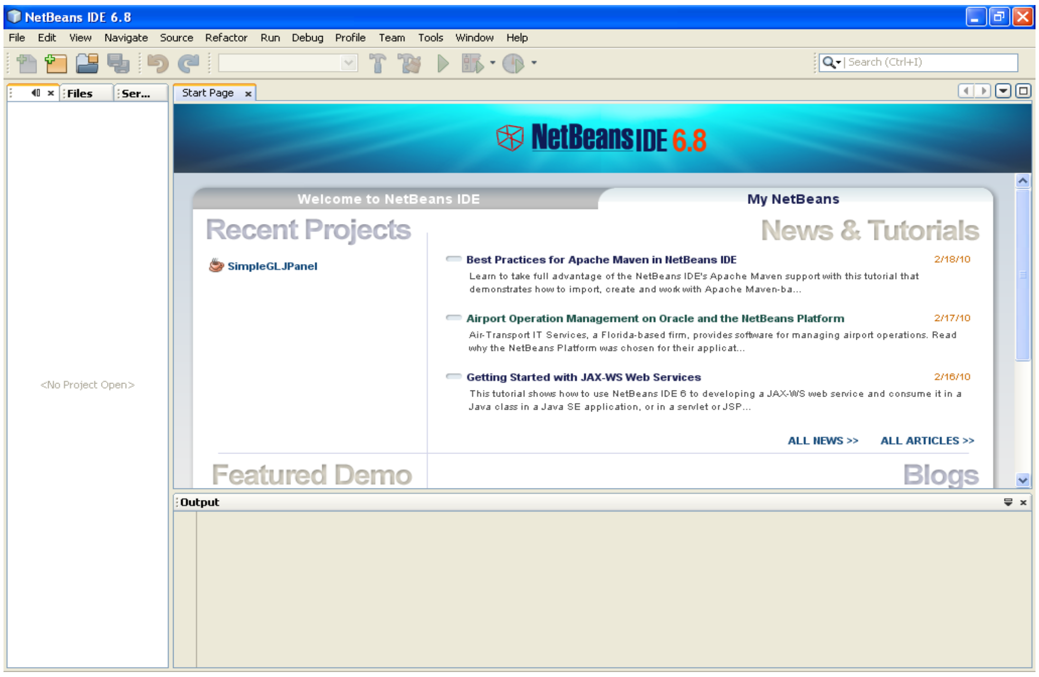
y1 - the first point's *y* coordinate.

x2 - the second point's *x* coordinate.

y2 - the second point's *y* coordinate.

C. Netbean:

You can download NetBeans from the website: http://netbeans.org/ (the latest version) and set up on your computer (checked: Tomcat, Glassfish, JDK, and full IDE environment).
The interface of NetBean IDE is as below:



Best Practices for Apache Maven in NetBeans IDE 2/18/10
Learn to take full advantage of the NetBeans IDE's Apache Maven support with this tutorial that demonstrates how to import, create and work with Apache Maven-ba...

Airport Operation Management on Oracle and the NetBeans Platform 2/17/10
Air-Transport IT Services, a Florida-based firm, provides software for managing airport operations. Read why the NetBeans Platform was chosen for their applicat...

Getting Started with JAX-WS Web Services 2/16/10
This tutorial shows how to use NetBeans IDE 6 to developing a JAX-WS web service and consume it in a Java class in a Java SE application, or in a servlet or JSP...

ALL NEWS >> ALL ARTICLES >>

Figure 1: Netbean 6.8

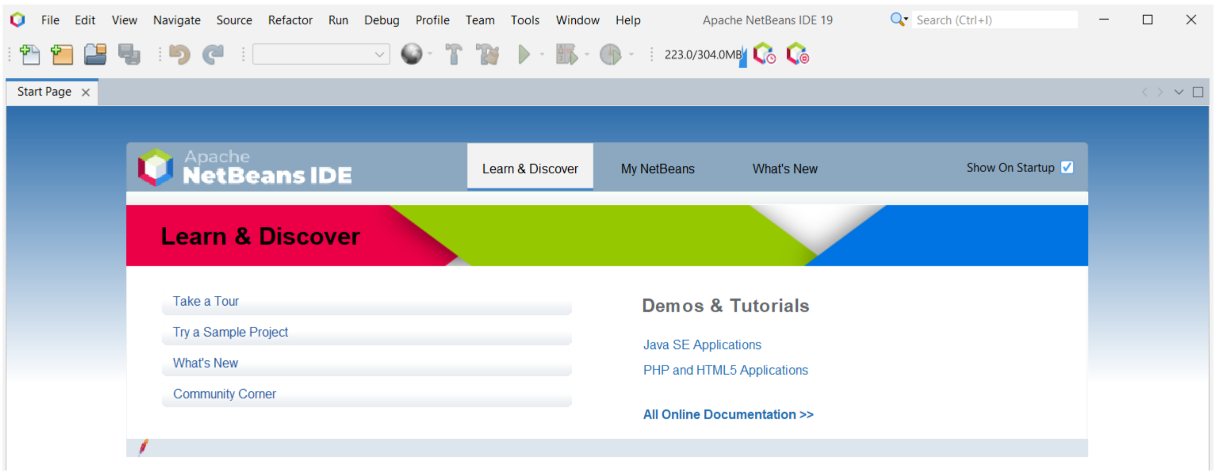
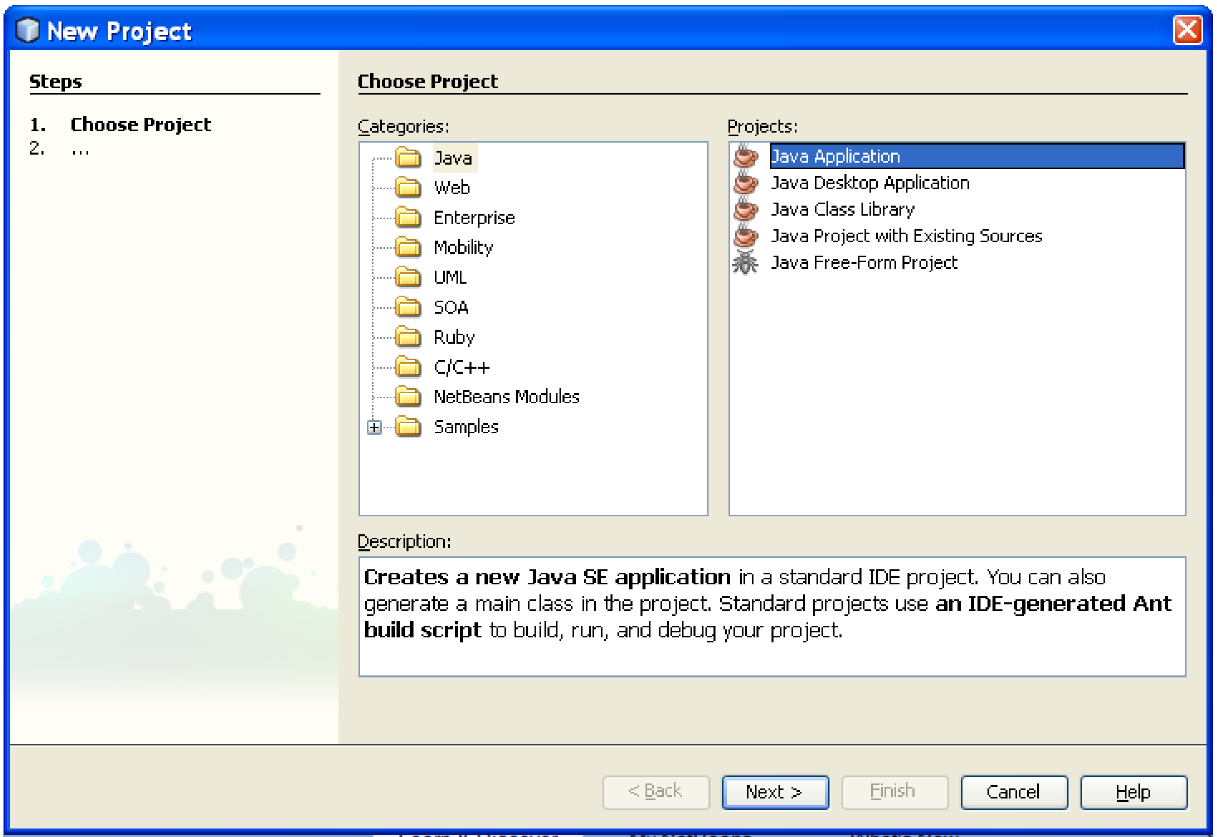
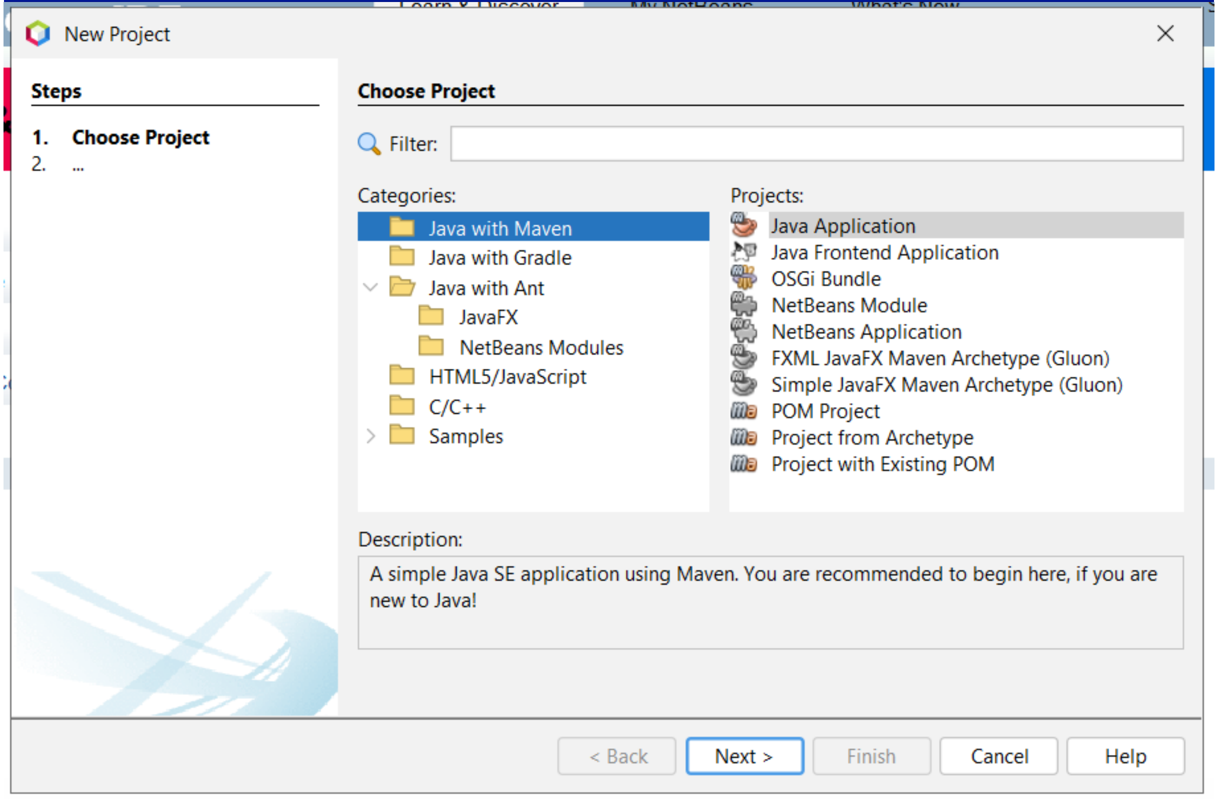


Figure 2: Netbean 19 (newer)

After Installing the Netbeans, create a new project:

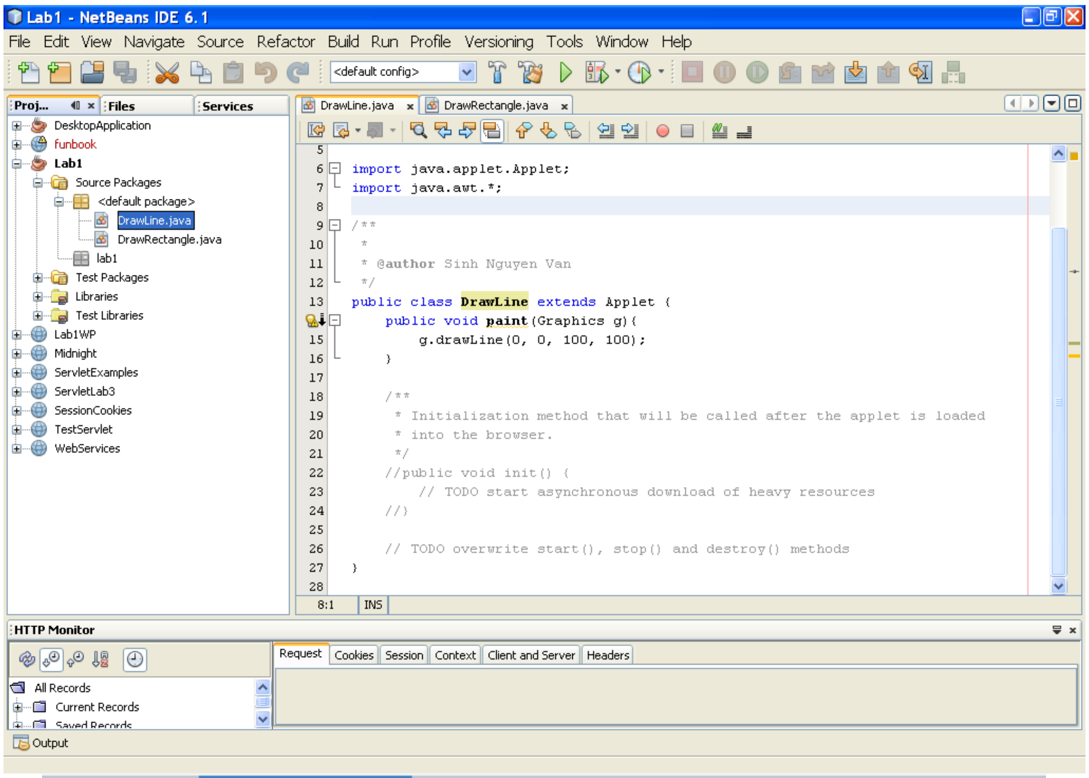


Creates a new Java SE application in a standard IDE project. You can also generate a main class in the project. Standard projects use **an IDE-generated Ant build script** to build, run, and debug your project.

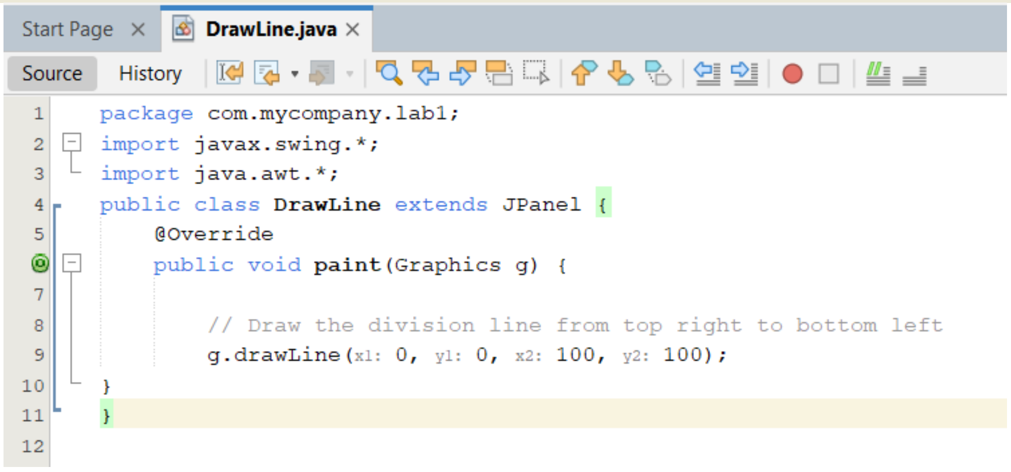


A simple Java SE application using Maven. You are recommended to begin here, if you are new to Java!

Click Next -> Finish



```java

import java.applet.Applet;
import java.awt.*;

/**
 *
 * @author Sinh Nguyen Van
 */
public class DrawLine extends Applet {
    public void paint(Graphics g){
        g.drawLine(0, 0, 100, 100);
    }

    /**
     * Initialization method that will be called after the applet is loaded
     * into the browser.
     */
    //public void init() {
        // TODO start asynchronous download of heavy resources
    //}

    // TODO overwrite start(), stop() and destroy() methods
}

```

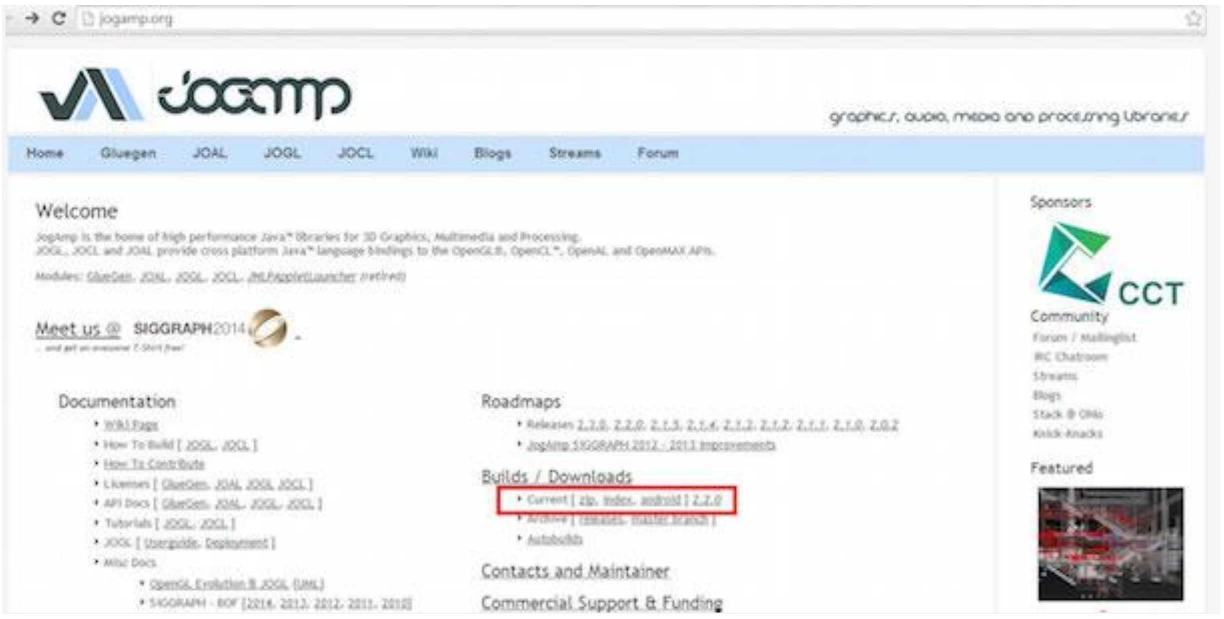


```java
package com.mycompany.lab1;
import javax.swing.*;
import java.awt.*;
public class DrawLine extends JPanel {
    @Override
    public void paint(Graphics g) {

        // Draw the division line from top right to bottom left
        g.drawLine(x1: 0, y1: 0, x2: 100, y2: 100);
    }
}

```

D. Set up JOGL on Netbeans

JOGL Installation on Netbeans 8:
(Source: https://www.tutorialspoint.com/jogl/jogl_installation.htm)

Downloading JOGL
- You can download latest version of JOGL from the website www.jogamp.org
- Go to the home page of www.jogamp.org
- Click on Builds/Downloads > Current (zip).


jogamp.org

graphics, audio, media and processing libraries

Home Gluegen JOAL JOGL JOCL Wiki Blogs Streams Forum

Welcome

JogAmp is the home of high performance Java™ libraries for 3D Graphics, Multimedia and Processing.
JOGL, JOCL and JOAL provide cross platform Java™ language bindings to the OpenGL®, OpenCL™, OpenAL and OpenMAX APIs.

Modules: GlueGen, JOAL, JOGL, JOCL, JNLPAppletLauncher (retired)

Meet us @ SIGGRAPH2014
... and get an awesome T-Shirt free!

Documentation
- Wiki Page
- How To Build [JOGL, JOCL]
- How To Contribute
- Licenses [GlueGen, JOAL, JOGL, JOCL]
- API Docs [GlueGen, JOAL, JOGL, JOCL]
- Tutorials [JOGL, JOCL]
- JOGL [Userguide, Deployment]
- Misc Docs
  - OpenGL Evolution & JOGL (UML)
  - SIGGRAPH - BOF [2014, 2013, 2012, 2011, 2010]

Roadmaps
- Releases 2.3.0, 2.2.0, 2.1.5, 2.1.4, 2.1.3, 2.1.2, 2.1.1, 2.1.0, 2.0.2
- JogAmp SIGGRAPH 2012 - 2013 Improvements

Builds / Downloads
- Current [zip, index, android] 2.2.0
- Archive [releases, master branch]
- Autobuilds

Contacts and Maintainer

Commercial Support & Funding

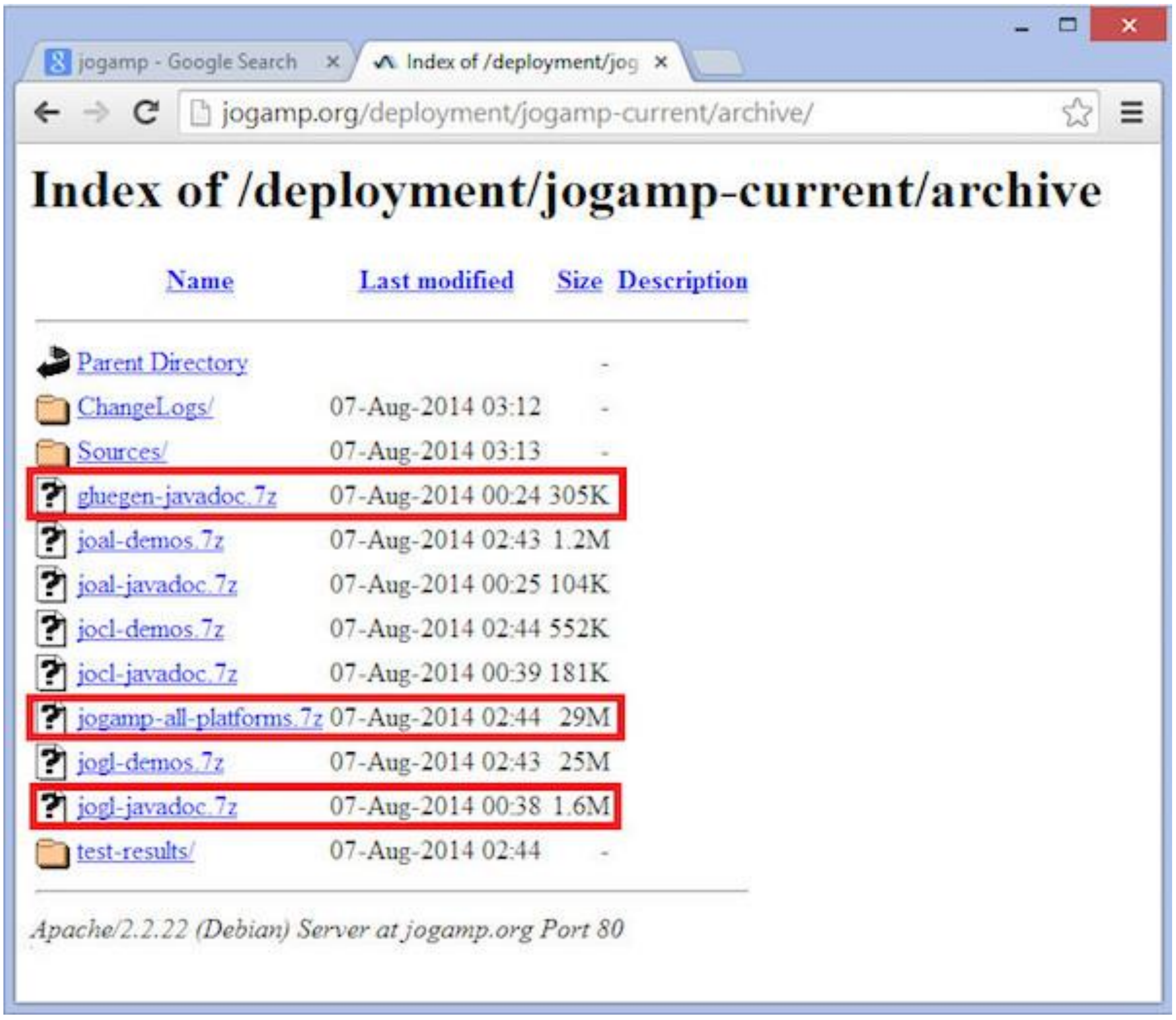
Sponsors

CCT

Community
Forum / Mailinglist
IRC Chatroom
Streams
Blogs
Stack @ OHo
Knick-Knacks

Featured

This takes you to the list of .jar files for all APIs maintained by the website.



jogamp - Google Search | Index of /deployment/jog

jogamp.org/deployment/jogamp-current/archive/

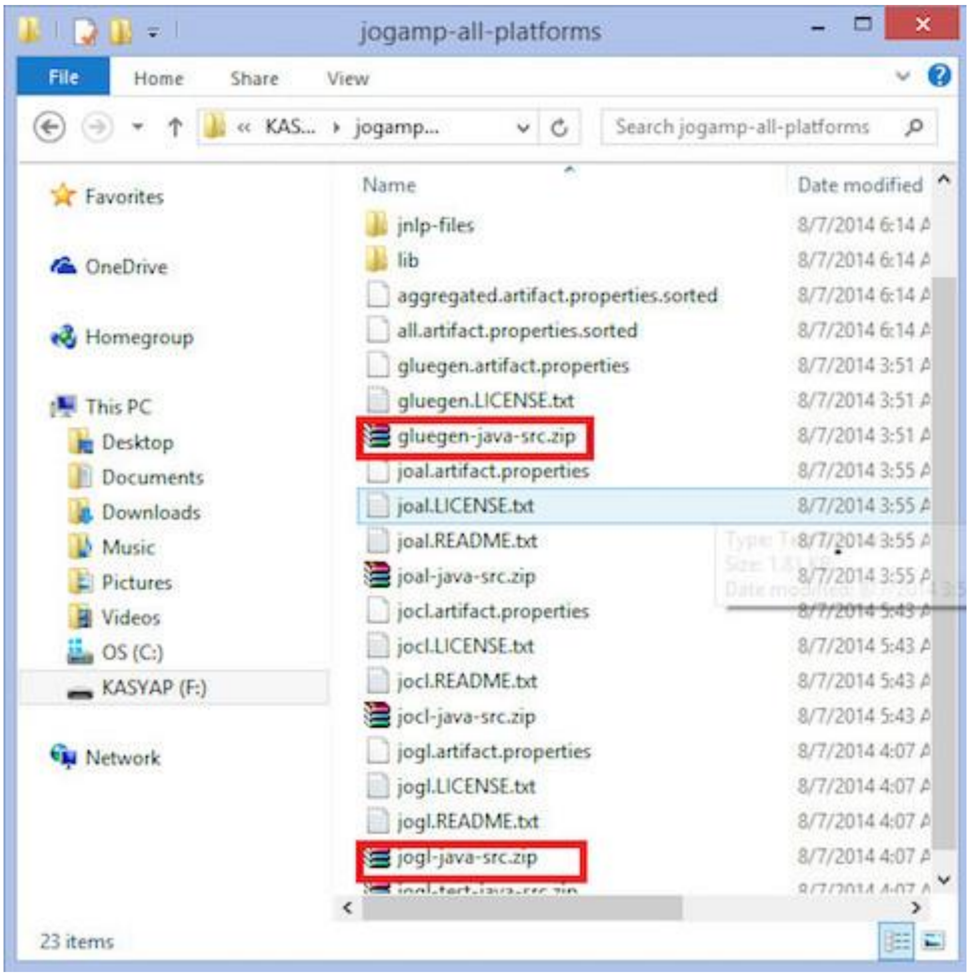
Index of /deployment/jogamp-current/archive

Name	Last modified	Size	Description
Parent Directory		-	
ChangeLogs/	07-Aug-2014 03:12	-	
Sources/	07-Aug-2014 03:13	-	
gluegen-javadoc.7z	07-Aug-2014 00:24	305K	
joal-demos.7z	07-Aug-2014 02:43	1.2M	
joal-javadoc.7z	07-Aug-2014 00:25	104K	
jocl-demos.7z	07-Aug-2014 02:44	552K	
jocl-javadoc.7z	07-Aug-2014 00:39	181K	
jogamp-all-platforms.7z	07-Aug-2014 02:44	29M	
jogl-demos.7z	07-Aug-2014 02:43	25M	
jogl-javadoc.7z	07-Aug-2014 00:38	1.6M	
test-results/	07-Aug-2014 02:44	-	

Apache/2.2.22 (Debian) Server at jogamp.org Port 80

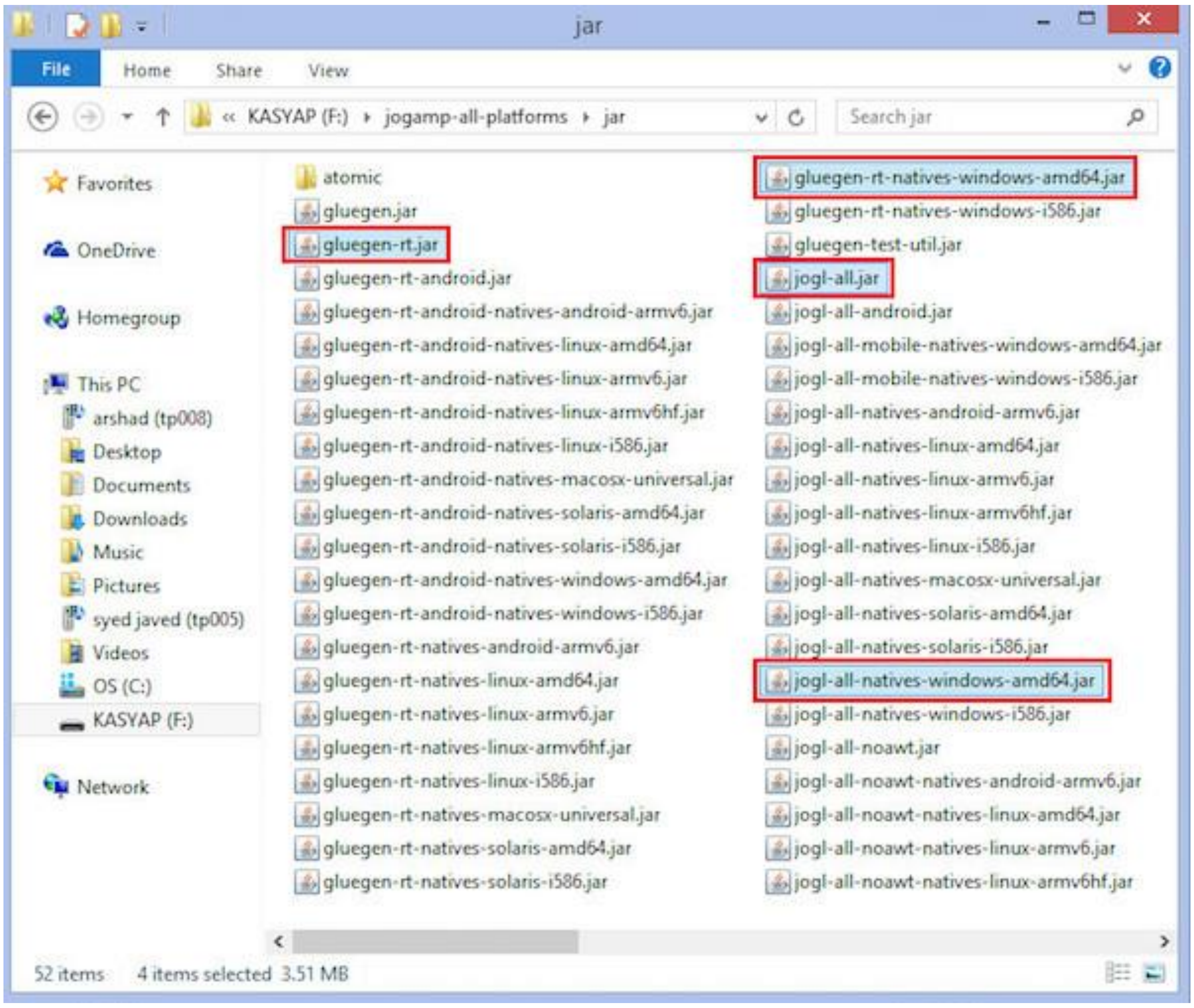
- Download the library .jar file jogamp-all-platforms.7z.
- Extract the downloaded .jar files using any zip extracting software.

When you open the extracted folder, you will find jar folder, source-codes, and other files.



Name	Date modified
jnlp-files	8/7/2014 6:14 A
lib	8/7/2014 6:14 A
aggregated.artifact.properties.sorted	8/7/2014 6:14 A
all.artifact.properties.sorted	8/7/2014 6:14 A
gluegen.artifact.properties	8/7/2014 3:51 A
gluegen.LICENSE.txt	8/7/2014 3:51 A
gluegen-java-src.zip	8/7/2014 3:51 A
joal.artifact.properties	8/7/2014 3:55 A
joal.LICENSE.txt	8/7/2014 3:55 A
joal.README.txt	8/7/2014 3:55 A
joal-java-src.zip	8/7/2014 3:55 A
jocl.artifact.properties	8/7/2014 5:43 A
jocl.LICENSE.txt	8/7/2014 5:43 A
jocl.README.txt	8/7/2014 5:43 A
jocl-java-src.zip	8/7/2014 5:43 A
jogl.artifact.properties	8/7/2014 4:07 A
jogl.LICENSE.txt	8/7/2014 4:07 A
jogl.README.txt	8/7/2014 4:07 A
jogl-java-src.zip	8/7/2014 4:07 A

Get the source codes gluegen-java-src.zip and jogl-java-src.zip for supporting IDE. This is optional.
Inside the jar folder, there are multiple .jar files. This collection of files belongs to Glugen and JOGL.
JOAMP provides native libraries that support various operating systems such as Windows, Solaris, Linux and Android. Hence, you need to take appropriate jar files which can execute on your desired platform. For example, if you are using Windows 64-bit operating system, then get the following .jar files from the jar folder
gluegenrt.jar
jogl-all.jar
gluegen-rt-natives-windows-amd64.jar
jogl-all-natives-windowsamd64.jar



atomic
gluegen.jar
gluegen-rt.jar
gluegen-rt-android.jar
gluegen-rt-android-natives-android-armv6.jar
gluegen-rt-android-natives-linux-amd64.jar
gluegen-rt-android-natives-linux-armv6.jar
gluegen-rt-android-natives-linux-armv6hf.jar
gluegen-rt-android-natives-linux-i586.jar
gluegen-rt-android-natives-macosx-universal.jar
gluegen-rt-android-natives-solaris-amd64.jar
gluegen-rt-android-natives-solaris-i586.jar
gluegen-rt-android-natives-windows-amd64.jar
gluegen-rt-android-natives-windows-i586.jar
gluegen-rt-natives-android-armv6.jar
gluegen-rt-natives-linux-amd64.jar
gluegen-rt-natives-linux-armv6.jar
gluegen-rt-natives-linux-armv6hf.jar
gluegen-rt-natives-linux-i586.jar
gluegen-rt-natives-macosx-universal.jar
gluegen-rt-natives-solaris-amd64.jar
gluegen-rt-natives-solaris-i586.jar
gluegen-rt-natives-windows-amd64.jar
gluegen-rt-natives-windows-i586.jar
gluegen-test-util.jar
jogl-all.jar
jogl-all-android.jar
jogl-all-mobile-natives-windows-amd64.jar
jogl-all-mobile-natives-windows-i586.jar
jogl-all-natives-android-armv6.jar
jogl-all-natives-linux-amd64.jar
jogl-all-natives-linux-armv6.jar
jogl-all-natives-linux-armv6hf.jar
jogl-all-natives-linux-i586.jar
jogl-all-natives-macosx-universal.jar
jogl-all-natives-solaris-amd64.jar
jogl-all-natives-solaris-i586.jar
jogl-all-natives-windows-amd64.jar
jogl-all-natives-windows-i586.jar
jogl-all-noawt.jar
jogl-all-noawt-natives-android-armv6.jar
jogl-all-noawt-natives-linux-amd64.jar
jogl-all-noawt-natives-linux-armv6.jar
jogl-all-noawt-natives-linux-armv6hf.jar

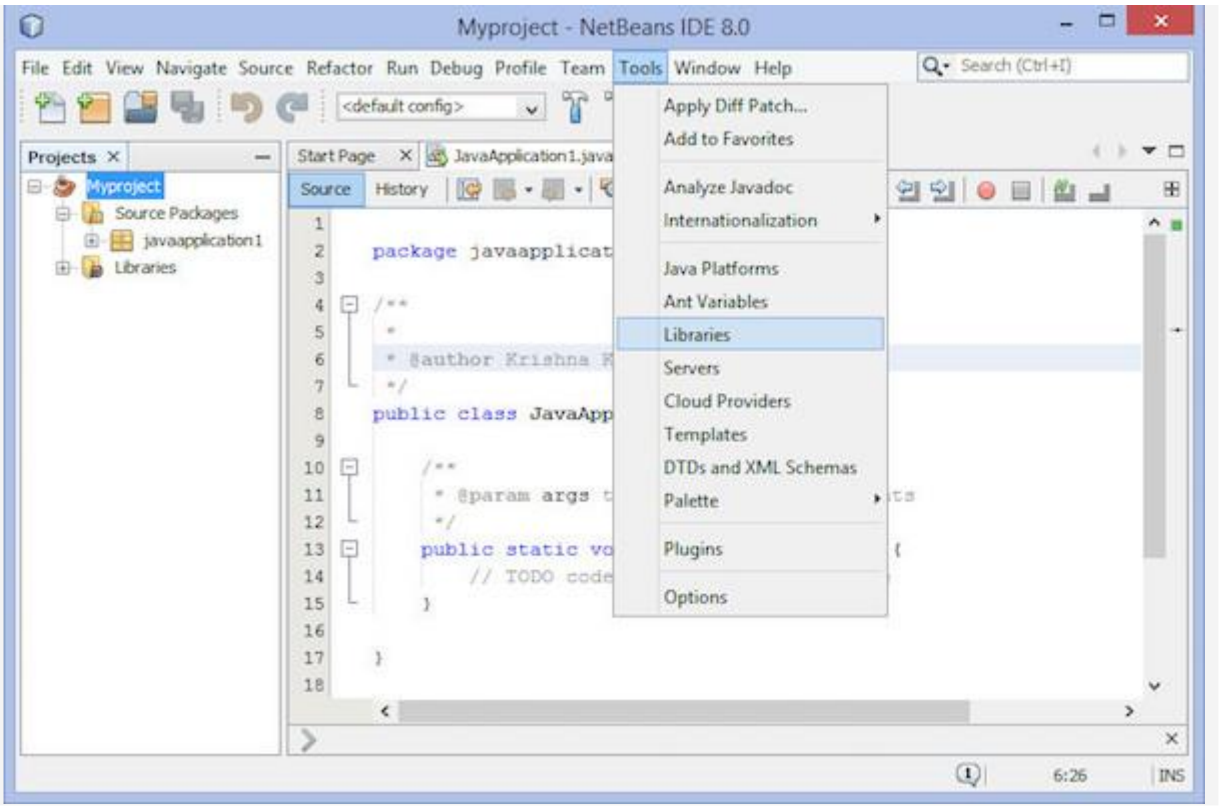
52 items 4 items selected 3.51 MB

1.1 Setting up JOGL for NetBeans 8.2

Let us go through the steps for setting up JOGL for NetBeans 8.2 −

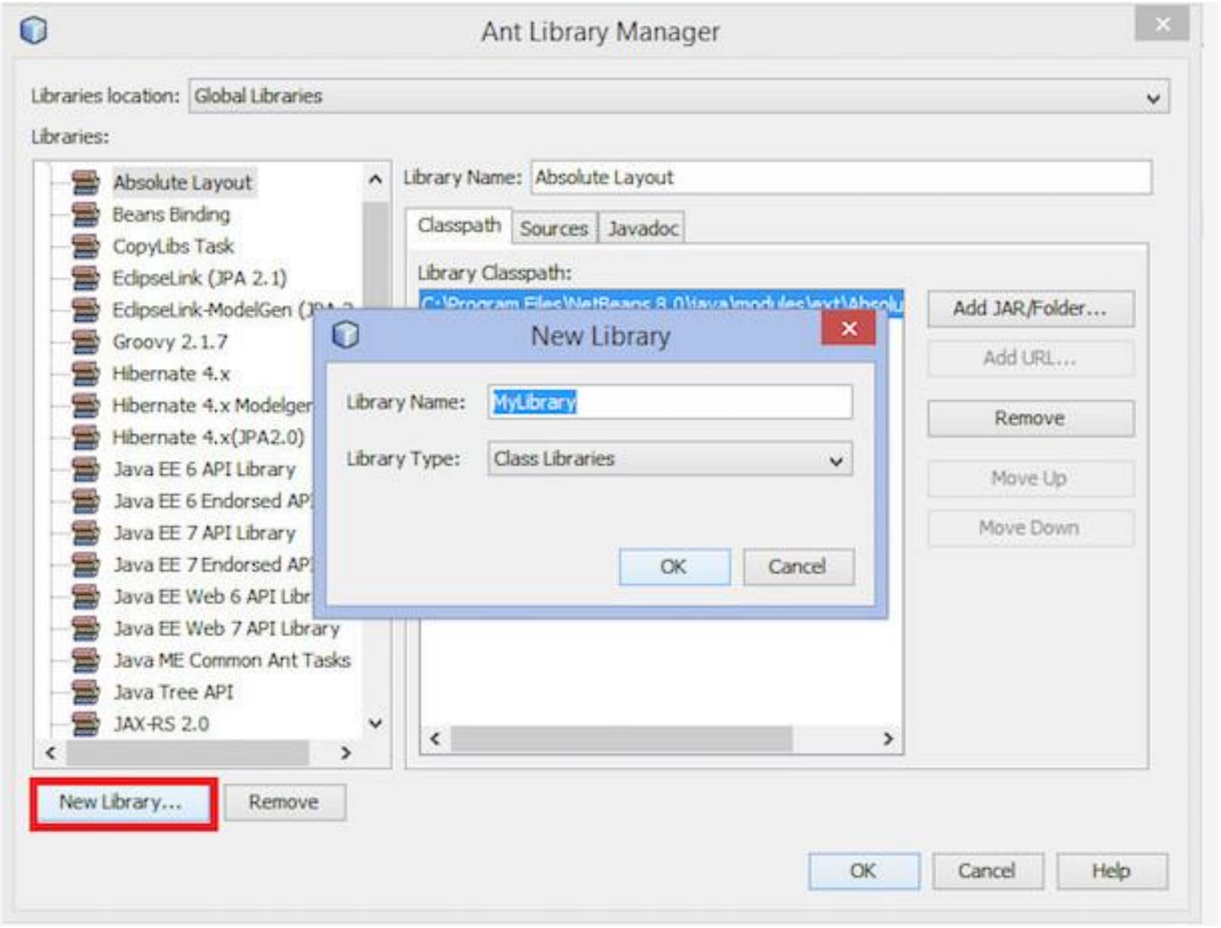
Adding Libraries

Step 1 − In the main menu, select **Tools > Libraries**.



```
package javaapplicat
/**
 *
 * @author Krishna K
 */
public class JavaApp
    /**
     * @param args t
     */
    public static vo
        // TODO code
    }
}
```

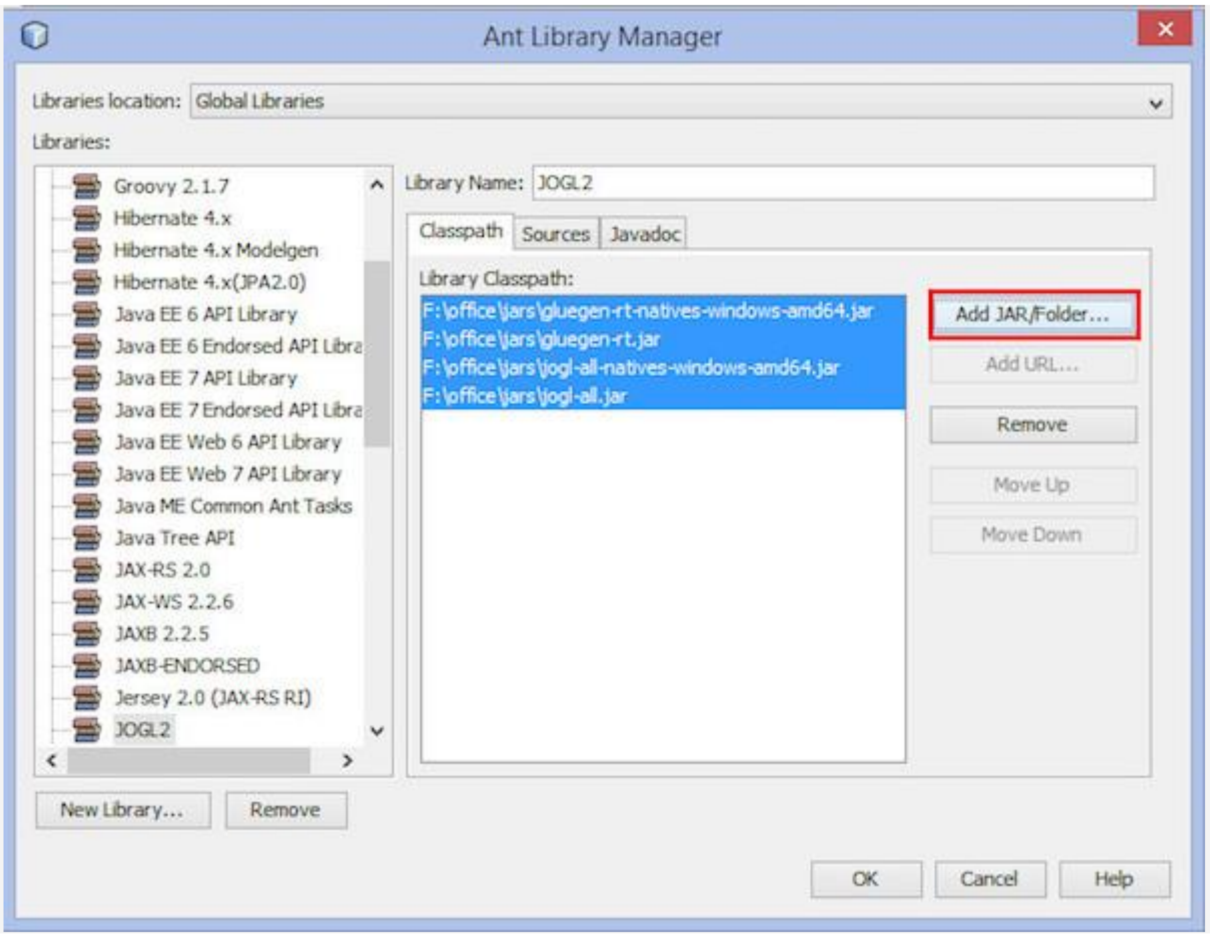
Step 2 − It leads you to **Ant Library Manager**.



Step 3 − Under the **Classpath** tab, click **New Library** button located on the left lower corner. It opens a small dialog box.

Step 4 − Enter Library name as **JoGl2.0.**

Step 5 − Click on "OK" button.



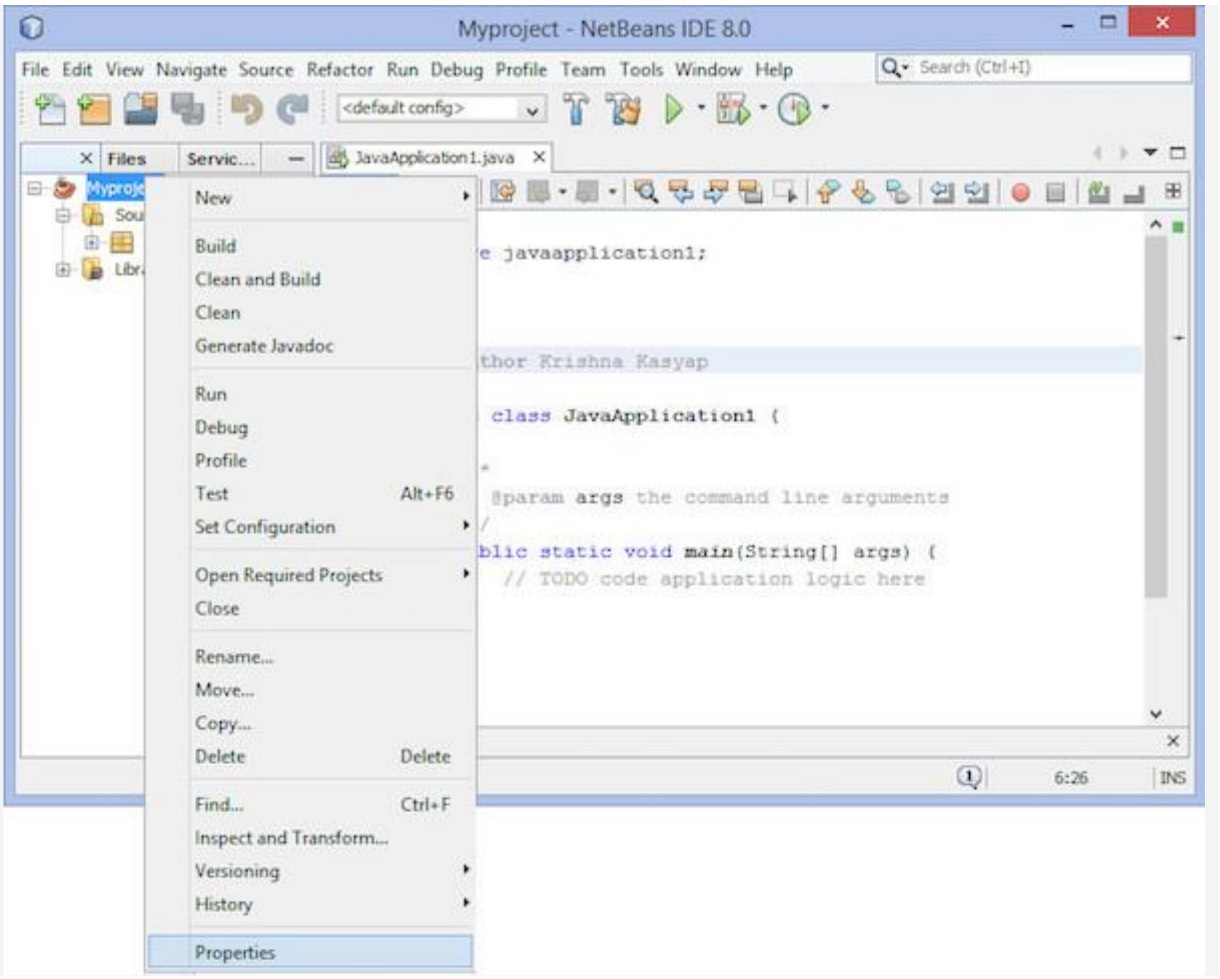
```
F:\office\jars\gluegen-rt-natives-windows-amd64.jar
F:\office\jars\gluegen-rt.jar
F:\office\jars\jogl-all-natives-windows-amd64.jar
F:\office\jars\jogl-all.jar
```

Step 6 − Click on "Add JAR/Folder…" button.

Step 7 − Select the path where .jar files **jogl.all.jar** and **gluegen-rt.jar** are located.

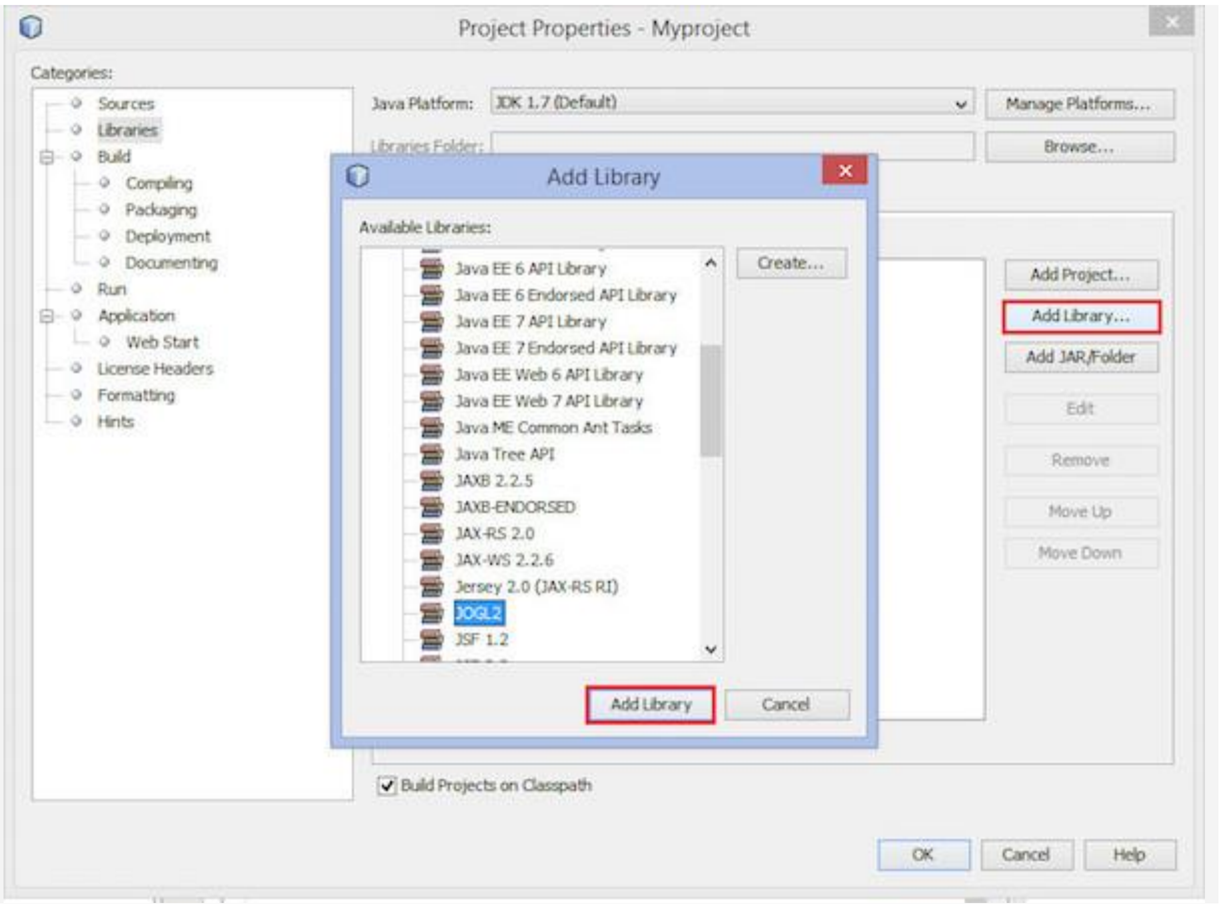
To include JOGL library into each project, follow the steps given below −

Step 1 − Right-click on the **project name**. It shows a short-cut menu.



```
e javaapplication1;

thor Krishna Kasyap

 class JavaApplication1 {

*
 @param args the command line arguments
/
blic static void main(String[] args) {
  // TODO code application logic here
```

Step 2 − Select **Properties.** It opens a window named **Project properties**.



Step 3 – Select **Libraries** from Categories on the left hand side.

Step 4 – Select **Compile tab** and click on "Add Library..." button. Add library dialog box comes up.

Step 5 – Now add JOGL2.0 library, which you created earlier.

Now you can test your first program with JOGL, please create the Java file: Triangledepthtest.java

```
import com.jogamp.opengl.GL2;
import com.jogamp.opengl.GLAutoDrawable;
import com.jogamp.opengl.GLCapabilities;
import com.jogamp.opengl.GLEventListener;
import com.jogamp.opengl.GLProfile;
import com.jogamp.opengl.awt.GLCanvas;
import com.jogamp.opengl.glu.GLU;
import javax.swing.JFrame;


import com.jogamp.opengl.util.FPSAnimator;

public class Triangledepthtest implements GLEventListener {

   private GLU glu = new GLU();
   private float rtri = 0.0f;

   @Override
```

```
public void display( GLAutoDrawable drawable ) {

  final GL2 gl = drawable.getGL().getGL2();

  gl.glShadeModel( GL2.GL_SMOOTH );
  gl.glClearColor( 0f, 0f, 0f, 0f );
  gl.glClearDepth( 1.0f );
  gl.glEnable( GL2.GL_DEPTH_TEST );
  gl.glDepthFunc( GL2.GL_LEQUAL );
  gl.glHint(GL2.GL_PERSPECTIVE_CORRECTION_HINT, GL2.GL_NICEST);

  // Clear The Screen And The Depth Buffer
  gl.glClear(GL2.GL_COLOR_BUFFER_BIT | GL2.GL_DEPTH_BUFFER_BIT);
  gl.glLoadIdentity(); // Reset The View
  gl.glTranslatef( -0.5f,0.0f,-6.0f ); // Move the triangle
  gl.glRotatef( rtri, 0.0f, 1.0f, 0.0f );
  gl.glBegin( GL2.GL_TRIANGLES );

  //drawing triangle in all dimensions
  //front
  gl.glColor3f( 1.0f, 0.0f, 0.0f ); // Red
  gl.glVertex3f( 1.0f, 2.0f, 0.0f ); // Top

  gl.glColor3f( 0.0f, 1.0f, 0.0f ); // Green
  gl.glVertex3f( -1.0f, -1.0f, 1.0f ); // Left

  gl.glColor3f( 0.0f, 0.0f, 1.0f ); // Blue
  gl.glVertex3f( 1.0f, -1.0f, 1.0f ); // Right)

  //right
  gl.glColor3f( 1.0f, 0.0f, 0.0f );
  gl.glVertex3f( 1.0f, 2.0f, 0.0f ); // Top

  gl.glColor3f( 0.0f, 0.0f, 1.0f );
  gl.glVertex3f( 1.0f, -1.0f, 1.0f ); // Left

  gl.glColor3f( 0.0f, 1.0f, 0.0f );
  gl.glVertex3f( 1.0f, -1.0f, -1.0f ); // Right

  //left
  gl.glColor3f( 1.0f, 0.0f, 0.0f );
  gl.glVertex3f( 1.0f, 2.0f, 0.0f ); // Top

  gl.glColor3f( 0.0f, 1.0f, 0.0f );
  gl.glVertex3f( 1.0f, -1.0f, -1.0f ); // Left

  gl.glColor3f( 0.0f, 0.0f, 1.0f );
  gl.glVertex3f( -1.0f, -1.0f, -1.0f ); // Right

  //top
  gl.glColor3f( 0.0f, 1.0f, 0.0f );
  gl.glVertex3f( 1.0f, 2.0f, 0.0f ); // Top
```

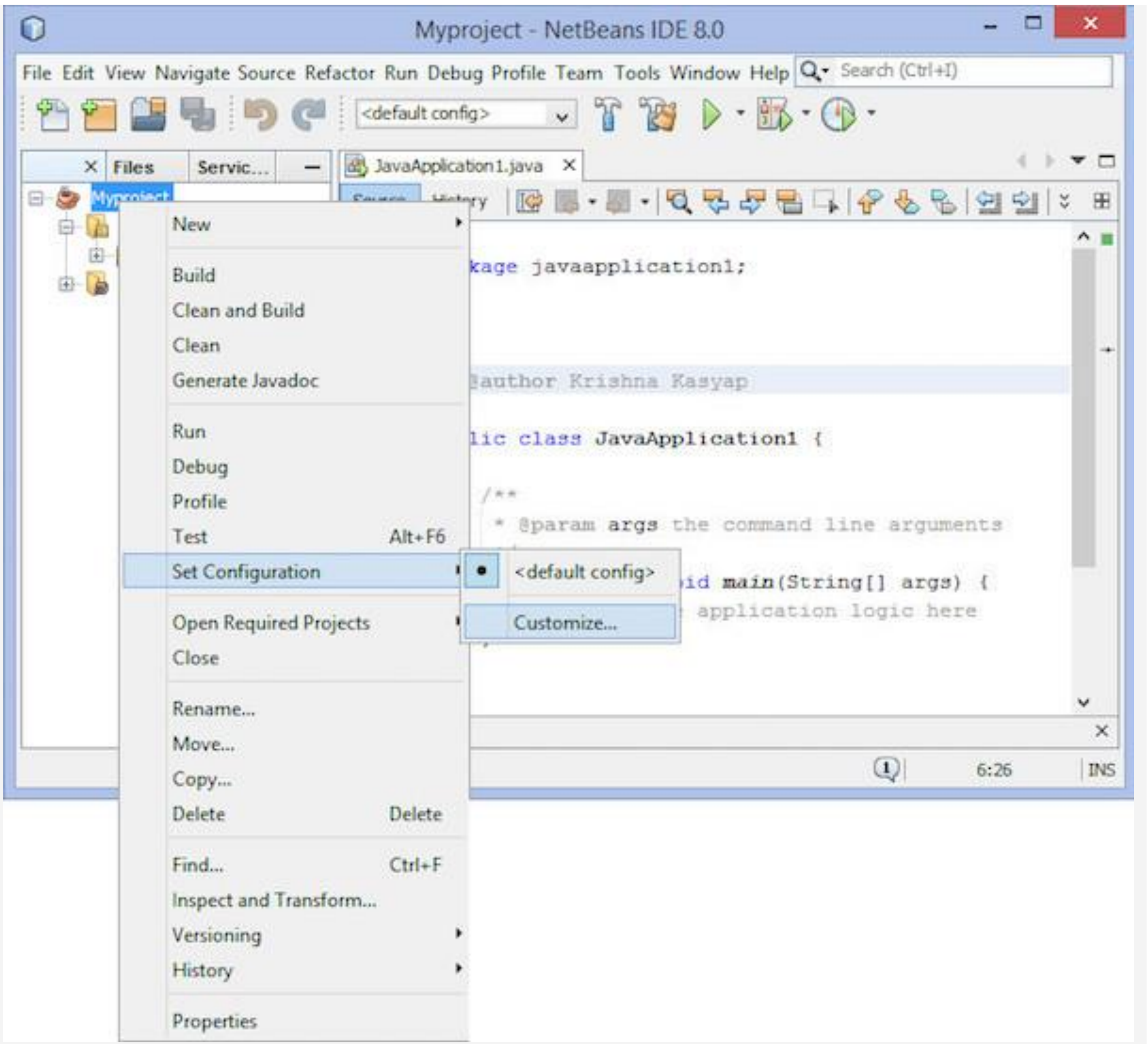
```
        gl.glColor3f( 0.0f, 0.0f, 1.0f );
        gl.glVertex3f( -1.0f, -1.0f, -1.0f ); // Left

        gl.glColor3f( 0.0f, 1.0f, 0.0f );
        gl.glVertex3f( -1.0f, -1.0f, 1.0f ); // Right

        gl.glEnd(); // Done Drawing 3d triangle (Pyramid)

        gl.glFlush();
        rtri += 0.2f;
    }

    @Override
    public void dispose( GLAutoDrawable drawable ) {
    }

    @Override
    public void init( GLAutoDrawable drawable ) {

        final GL2 gl = drawable.getGL().getGL2();

        gl.glShadeModel( GL2.GL_SMOOTH );
        gl.glClearColor( 0f, 0f, 0f, 0f );
        gl.glClearDepth( 1.0f );
        gl.glEnable( GL2.GL_DEPTH_TEST );
        gl.glDepthFunc( GL2.GL_LEQUAL );
        gl.glHint(GL2.GL_PERSPECTIVE_CORRECTION_HINT, GL2.GL_NICEST );
    }

    @Override
    public void reshape(GLAutoDrawable drawable, int x, int y, int width, int height ) {

        // TODO Auto-generated method stub
        final GL2 gl = drawable.getGL().getGL2();
        if( height <= 0 )
            height = 1;

        final float h = ( float ) width / ( float ) height;
        gl.glViewport( 0, 0, width, height );
        gl.glMatrixMode( GL2.GL_PROJECTION );
        gl.glLoadIdentity();

        glu.gluPerspective( 45.0f, h, 1.0, 20.0 );
        gl.glMatrixMode( GL2.GL_MODELVIEW );
        gl.glLoadIdentity();
    }

    public static void main( String[] args ) {

        // TODO Auto-generated method stub
        final GLProfile profile = GLProfile.get( GLProfile.GL2 );
    GLCapabilities capabilities = new GLCapabilities( profile );

    // The canvas
    final GLCanvas glcanvas = new GLCanvas( capabilities );
    Triangledepthtest triangledepthtest = new Triangledepthtest();

    glcanvas.addGLEventListener( triangledepthtest );
    glcanvas.setSize( 400, 400 );

    final JFrame frame = new JFrame ( "3d Triangle (solid)" );
    frame.getContentPane().add(glcanvas);
    frame.setSize( frame.getContentPane().getPreferredSize() );
    frame.setVisible( true );
    final FPSAnimator animator = new FPSAnimator( glcanvas, 300,true);

    animator.start();
  }

}
```

[OPTIONAL] Including Native Library in Each Project
(Source: https://www.tutorialspoint.com/jogl/jogl_installation.htm)

Follow the given steps to include native library in each project −

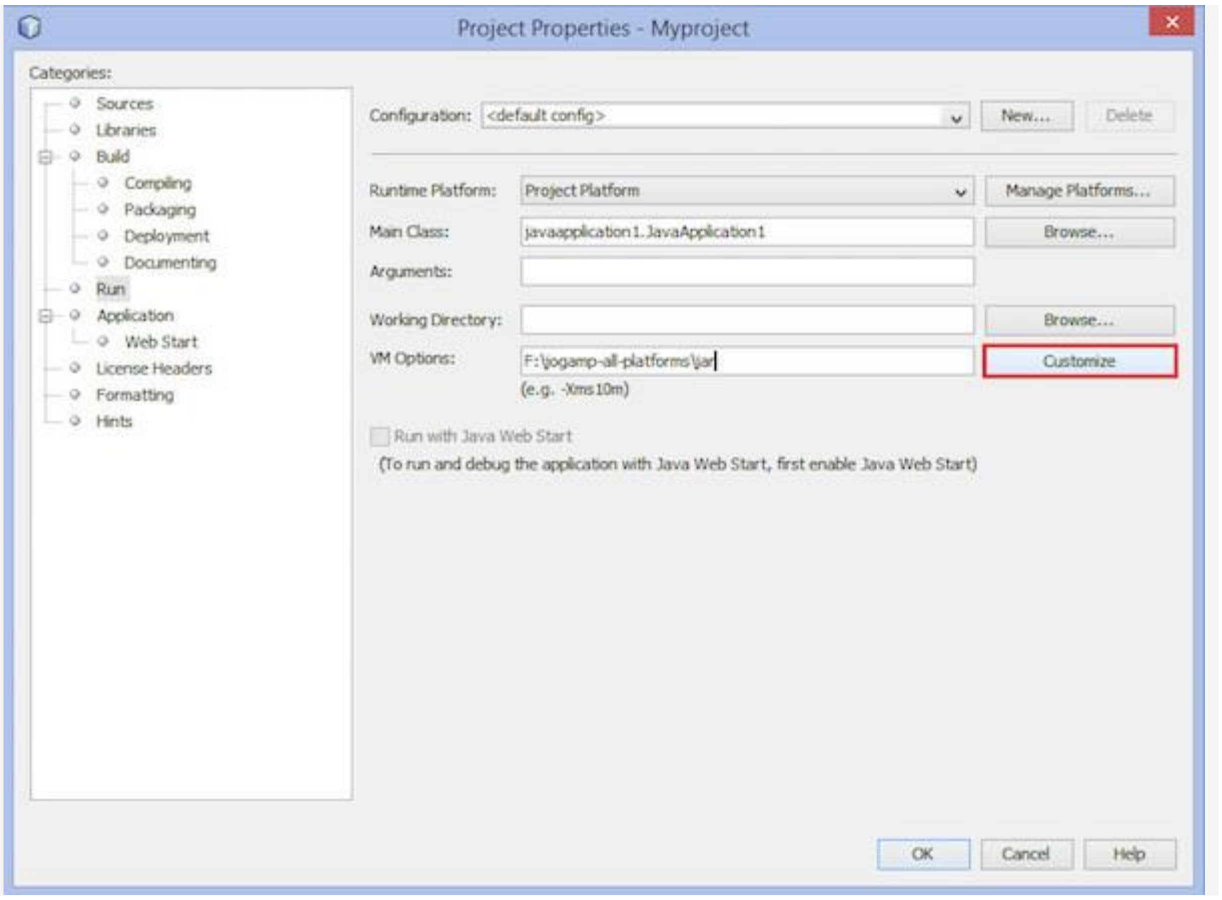
Step 1 − Right-click the project.

Step 2 − Select **Set Configuration > Customize...**



```
kage javaapplication1;
@author Krishna Kasyap
lic class JavaApplication1 {
/**
 * @param args the command line arguments
id main(String[] args) {
 application logic here
```

It leads you to the **Project Properties** window.



Configuration: <default config>

Runtime Platform: Project Platform

Main Class: javaapplication1.JavaApplication1

Arguments:

Working Directory:

VM Options: F:\jogamp-all-platforms\jar

(e.g. -Xms10m)

Run with Java Web Start

(To run and debug the application with Java Web Start, first enable Java Web Start)

Step 3 − On the right hand side, in **VM options,** click on "Customize" button.

Step 4 − Browse for the path that contains JOGL native libraries **gluegen-rtnatives-windows-amd64.jar''** and **'jogl-all-natives-windowsamd64.jar.**

Adding Java Documentation of Native Libraries

You need to open Ant library manager again to make sources and Javadoc available for each project. Follow the given procedure −

Step 1 − Open **main menu**.

Step 2 − Select **Tools > Libraries**. This leads you to **Library manager**.

Step 3 − Under the **JavaDoc** tab, click on "New Library…" button.

Step 4 − Enter **JOGLJavadoc** name. (You can enter any desired name.)

Step 5 − Click on "Add jars/libraries…" button.

Step 6 − Select the path where unzipped **JOGL documentation** code is located.

Adding Source Code of Native Libraries

Step 1 − Under **Sources** tab, click on "New Library…" button. Enter **JOGLsources** name.

Step 2 − Click on "Add jars/libraries…" button. Select the path where unzipped source code is located.

Customizing the JDK Editor

Step 1 − Set **Classpath** for files **jogl.all.jar** and **gluegen-rt.jar.**

Step 2 − Set path to native libraries *gluegen-rt-natives-windows-amd64.jar* and *joglall-natives-windowsamd64.jar* or copy all the jar files from the folder where you have downloaded them and paste them into the **jse lib** folder.

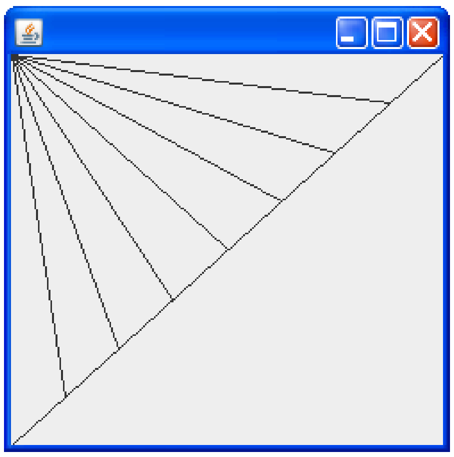
Part 2: Explore the Java graphic Libraries in Netbeans
Link: **https://docs.oracle.com/javase/7/docs/api/java/awt/Graphics.html**

- The Graphics class in AWT (e.g. in public void paint(Graphics g){…}) supports many methods to draw the following basic shapes:
- Lines
- Circle and Ellipes
- Rectangle and Polygon
- Images
- Text and Fonts
- Graphic 2D and 3D
- …
- Test with all above shapes.

Part 3: Practices and Exercises

Exercise 1:

Modify **DrawLine.java** and the result like this:

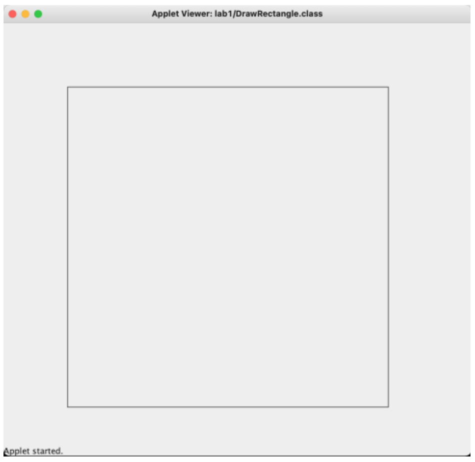


Please capture screenshots of your work then paste them here

Exercise 2:

1. Create a new project
2. Create a package (optional)
3. Create a java file: DrawRectangle.java, DrawEclipse.java, DrawStar.java

a) DrawRectangle.java



```
public void drawRect(int x,
          int y,
          int width,
          int height)
```

Draws the outline of the specified rectangle. The left and right edges of the rectangle are at x and x + width. The top and bottom edges are at y and y + height. The rectangle is drawn using the graphics context's current color.

Parameters:

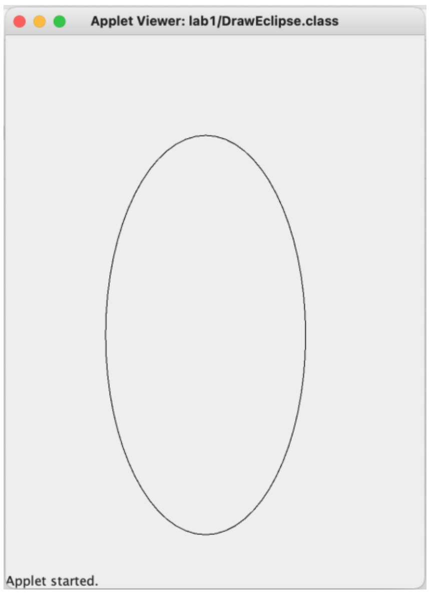
x - the x coordinate of the rectangle to be drawn.

y - the y coordinate of the rectangle to be drawn.

width - the width of the rectangle to be drawn.

height - the height of the rectangle to be drawn.

b) DrawEclipse.java



public abstract void drawOval(int x,
    int y,
    int width,
    int height)

Draws the outline of an oval. The result is a circle or ellipse that fits within the rectangle specified by the x, y, width, and height arguments.

The oval covers an area that is width + 1 pixels wide and height + 1 pixels tall.

Parameters:

x - the *x* coordinate of the upper left corner of the oval to be drawn.

y - the *y* coordinate of the upper left corner of the oval to be drawn.

width - the width of the oval to be drawn.

height - the height of the oval to be drawn.

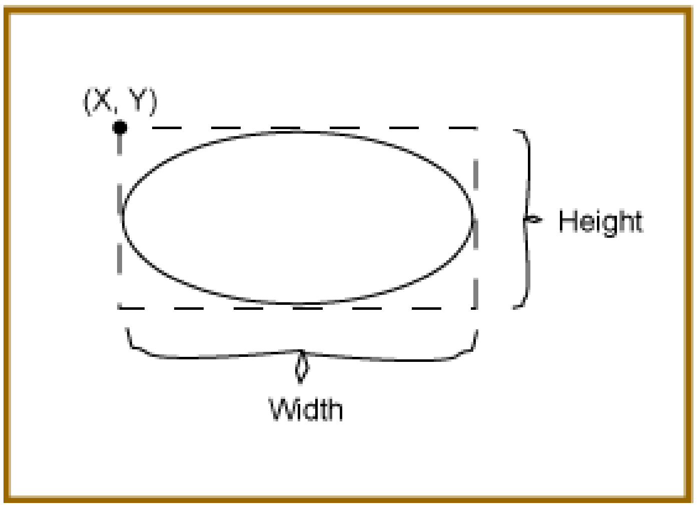
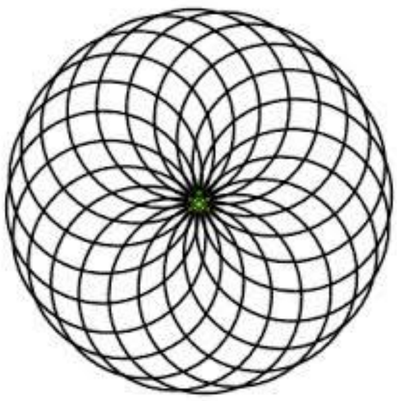


Figure. x, y, width, and height.

Please capture screenshots of your work then paste them here

Exercise 3:

1. Draw a set of circles as the below picture:



2. Fill in the different colors for each circle.

The definition of fillOval function

```
public abstract void fillOval(int x,
            int y,
            int width,
            int height)
```

Fills an oval bounded by the specified rectangle with the current color.

Parameters:

x - the *x* coordinate of the upper left corner of the oval to be filled.

y - the *y* coordinate of the upper left corner of the oval to be filled.

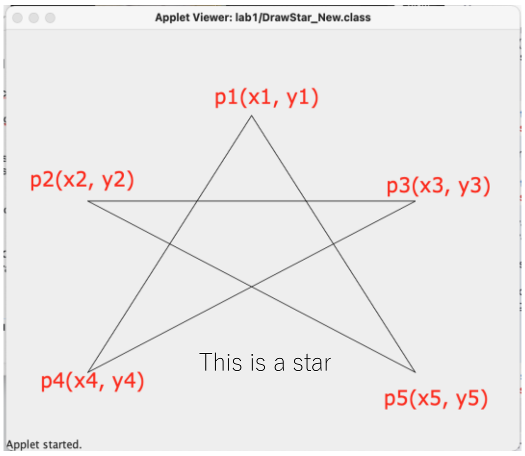
width - the width of the oval to be filled.

height - the height of the oval to be filled.

Please capture screenshots of your work then paste them here

Exercise 4:

1. Modify and fill color to all shapes
2. Calculate the points to draw a Star
3. Create 5 points (p1→p5)
4. Draw lines between the points in order to obtain a star.
5. Draw a Caption "This is a star"



The definition of drawString function

public abstract void drawString(String str,
int x,
int y)

Draws the text given by the specified string, using this graphics context's current font and color. The baseline of the leftmost character is at position (*x*, *y*) in this graphics context's coordinate system.

Parameters:

str - the string to be drawn.

x - the *x* coordinate.

y - the *y* coordinate.

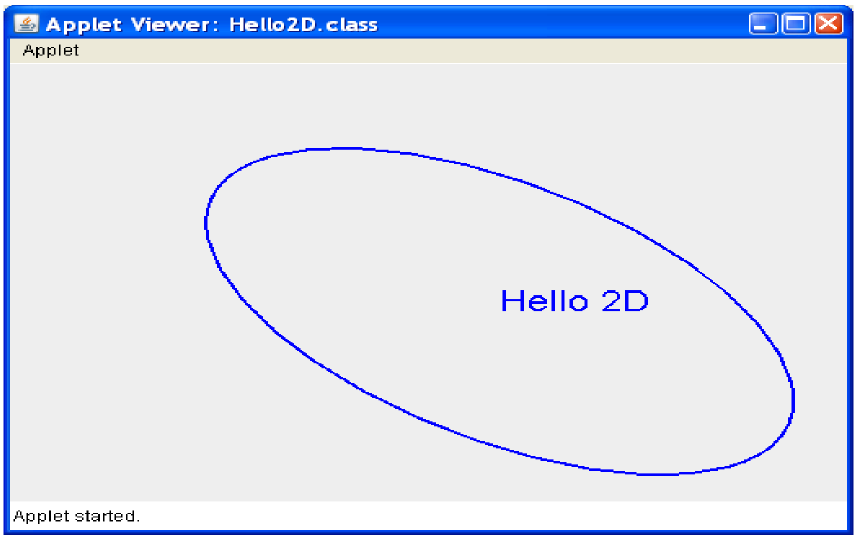
Please capture screenshots of your work then paste them here

Exercise 5:

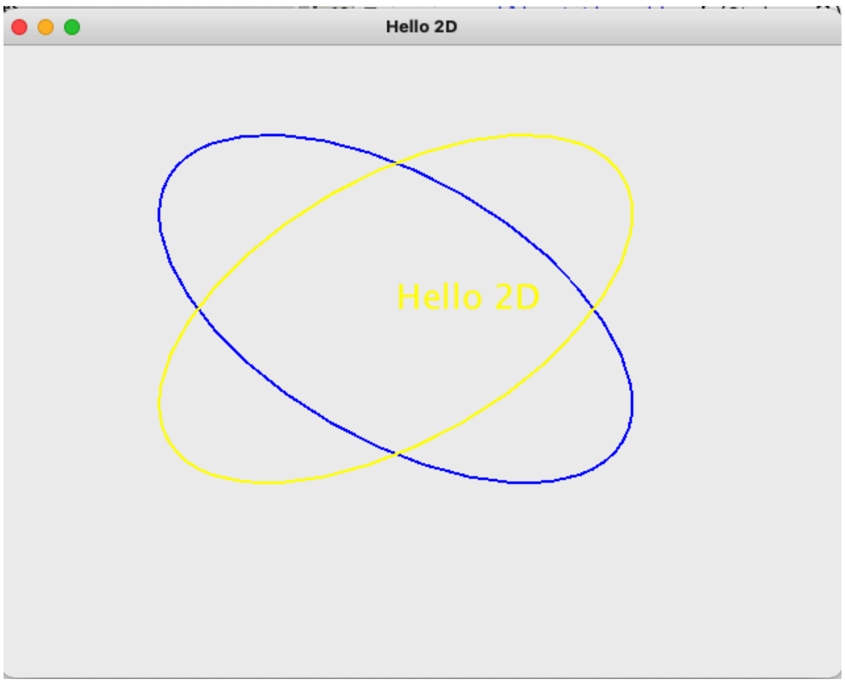
Create a new file "Hello2D.java", then copy source code below and compile it.

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.awt.geom.*;
```

```
public class Hello2D extends JApplet {
  public static void main(String s[]) {
    JFrame frame = new JFrame();
    frame.setTitle("Hello 2D");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    JApplet applet = new Hello2D();
    applet.init();
    frame.getContentPane().add(applet);
    frame.pack();
    frame.setVisible(true);
  }
  public void init() {
    JPanel panel = new Hello2DPanel();
    getContentPane().add(panel);
  }
}
class Hello2DPanel extends JPanel {
  public Hello2DPanel() {
    setPreferredSize(new Dimension(640, 480));
  }
  public void paintComponent(Graphics g) {
    super.paintComponent(g);
    Graphics2D g2 = (Graphics2D)g;
    g2.setColor(Color.blue);
    Ellipse2D e = new Ellipse2D.Double(-100, -50, 200, 100);
    AffineTransform tr = new AffineTransform();
    tr.rotate(Math.PI / 6.0);
    Shape shape = tr.createTransformedShape(e);
    g2.translate(300,200);
    g2.scale(2,2);
    g2.draw(shape);
    g2.drawString("Hello 2D", 0, 0);
  }
}
```



Modify the source code to obtain the last result as below:



Please capture screenshots of your work then paste them here