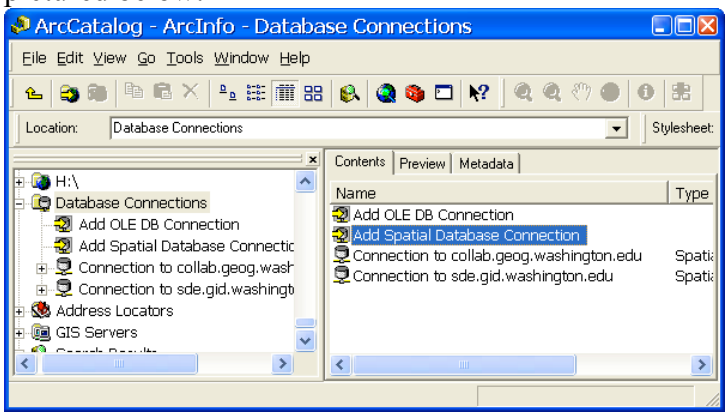
Lab 2. ArcSDE access with Java

Recap:

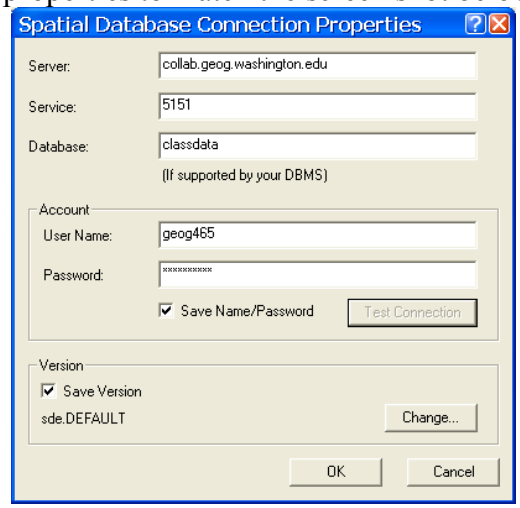
- Package declaration and import declaration
- Constructor of a class
- Main method of a class
- Class inheritance – the *extends* keyword
- Class variables
- Accessor method and mutator method
- Window application

Part 1: Hands on experience with ArcSDE

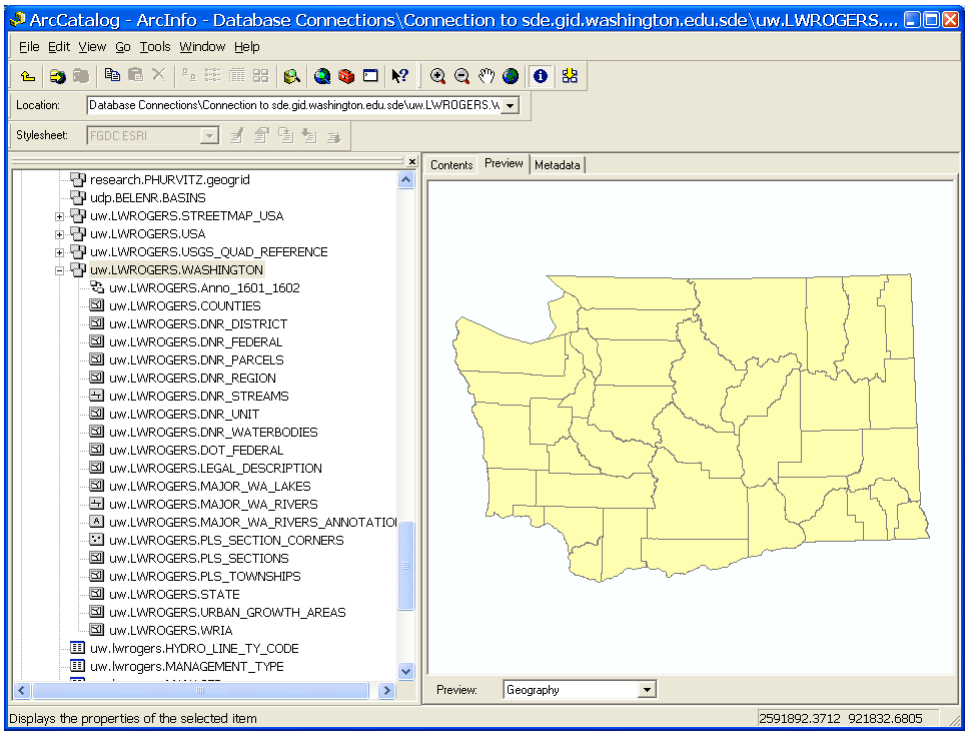
Start ArcCatalog. Connect to ArcSDE running on geography's Collab server (collab.geog.washington.edu), by using the "Add Spatial Database Connection" link, pictured below:



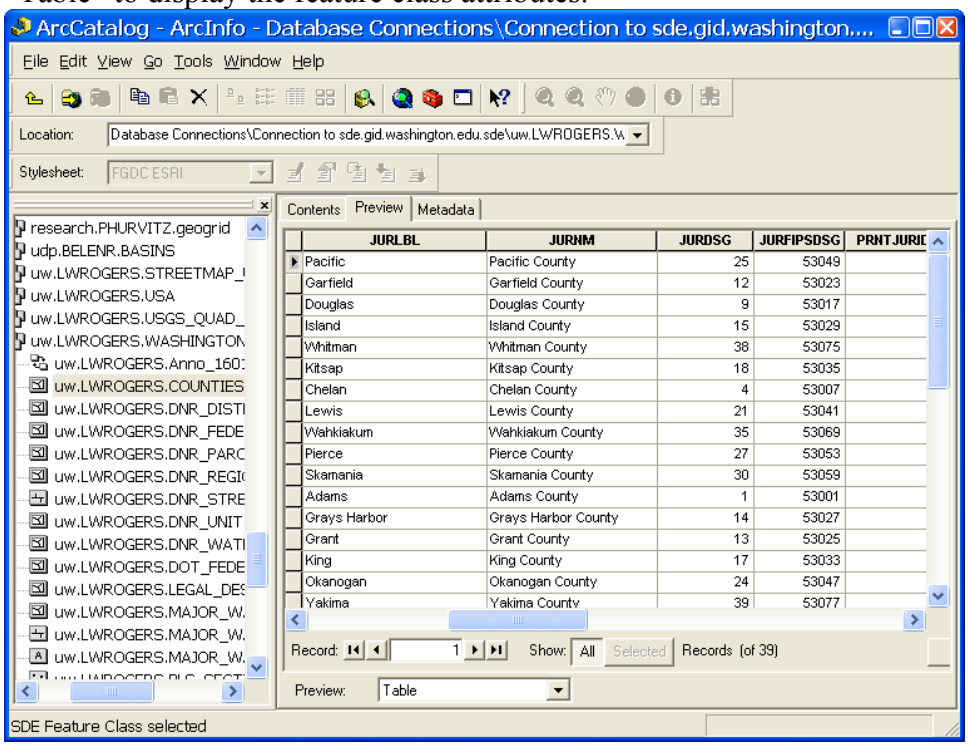
Double click the "Add Spatial Database Connection", to access the properties. Specify the properties to match the screen shot below:



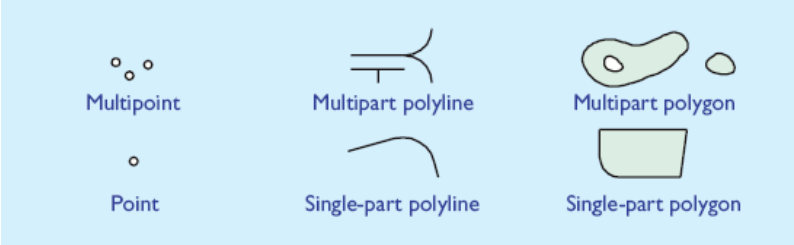
Explore the differences between a feature dataset and a feature class. In the screen shot below, notice that the feature dataset selected is "uw.LWROGERS.WASHINGTON" and a list of feature classes follow:



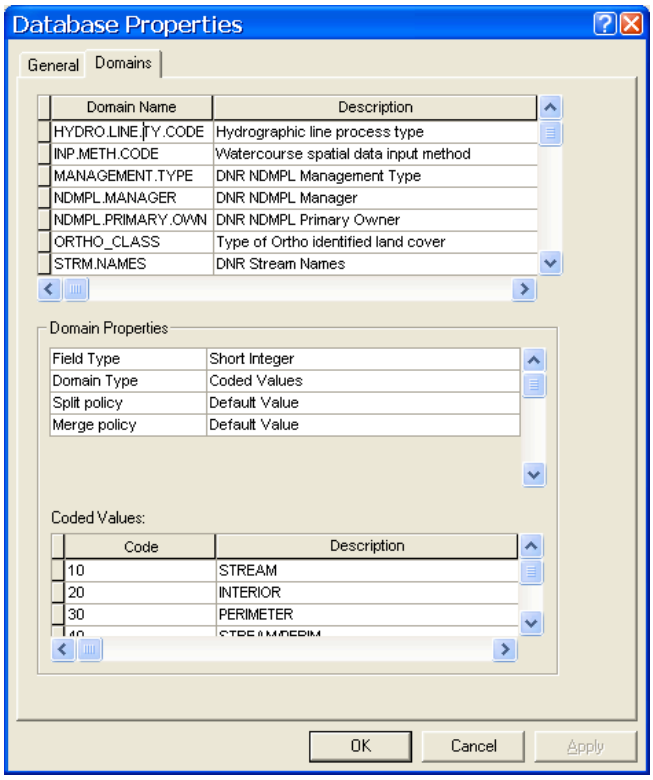
With a feature class selected, select the "Preview" tab and the Preview drop-down to select "Table" to display the feature class attributes:



Shape is one particular attribute of a feature class. There are six basic types of shapes, pictured below:



Right click on the SDE connection name, and select "Properties". The window below will be displayed if you select the "Domains" tab to find all the domains defined in this spatial database.



Part 2: Fetch feature data from geodatabase using JBuilder

After exploring ArcSDE using ArcCatalog, start JBuilder. We will now build a link to the geodatabase to obtain feature data, using Java. This portion of the lab will need to be completed with instructor support.

Create a new project in JBuilder, and save the project file in your student directory on the P:/ drive. Create a new class called "SDEAccess".

To access all the SDE java packages, you need to reference the two .jar files located in P:/geog465win05/sde_jars/. In JBuilder, navigate to the Project → Project Properties… menu. Select Paths in the left pane of the Project Properties window. Select the Required Libraries tab in the main pane of the window. Click the Add… button. Select the Archives tab and navigate to P:/geog465win05/sde_jars/. While holding the Shift key on the keyboard, select the two .jar files (multiple selection). Click OK to close the Add to

Project Classpath window. Click OK to close the Project Properties window. Now you have loaded the two required .jar files for this lab.

Return to the class you have created, called "SDEAccess" in JBuilder. Begin by importing the following packages:

```
import javax.swing.*;   // required for user interface
import javax.swing.border.*;  // required for interface layout
import java.awt.*;   // required for user interface
import java.util.Vector;  // required to use Vectors to store GIS layers
import com.esri.sde.sdk.client.*;  // required to use ArcSDE, jsde90.jar
```

Now, following the code below, we will explore the "SDEAccess" class, which will build a connection to a geodatabase called "classdata" on the collab.geog.washington.edu server. Begin typing the code into your new class, spending time to understand what is being coded.

Think through the answers to the following questions while exploring this code:

1. What information is needed to build an SDE connection?
2. What is a layer? How do you access a layer in ArcSDE?
3. How do you access a feature from a layer?

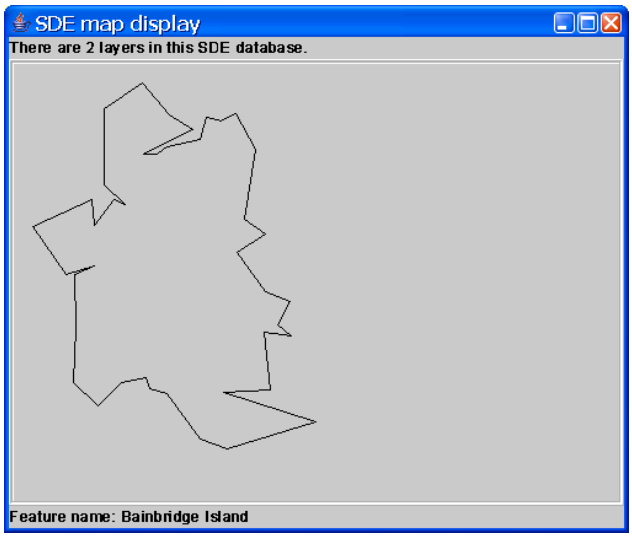
```
/**
 * <p>Title: ArcSDE access using Java client</p>
 * <p>Description: </p>
 * <p>Copyright: Copyright (c) GEOG 465 2005</p>
 * <p>Department of Geography, University of Washington</p>
 *
 * @author Guirong Zhou
 * @version 1.0
 */
public class SDEAccess extends JFrame{
    JLabel jLabel1;
    JLabel jLabel2;

    /**
     * Constructor:
     *
     * */
    public SDEAccess() {
        super("SDE map display");
        setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
        try {
            jbInit();
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }

    /**
     * Run this application by instantiating the class
     * */
    public static void main(String[] args) {
        new SDEAccess();
    }

    /**
     * This mutator method will do these steps:
     * <li>Initiate a connection to ArcSDE</li>
     * <li>Get the layers list</li>
     * <li>Get a feature from a layer</li>
     * <li>Display the feature' attrbutes using labels</li>
```

```
     * */
    private void jbInit() throws Exception {
        Container window = getContentPane();

        //Add three window controls to the window
        jLabel1 = new JLabel();
        jLabel2 = new JLabel();

        //prepare connection parameters
        SeConnection conn = null;
        String server     = "collab.geog.washington.edu";
        int instance      = 5151;
        String database   = "classdata";
        String user       = "geog465";
        String password   = "geogwin_05";
        try {
           //connect to SDE
           conn = new SeConnection(server, instance, database, user, password);

           //get the list of layers available in this SDE database
           Vector layerList = conn.getLayers();
           jLabel1.setText("There are " + layerList.size() + " layers in this SDE database.");

           //get a layer
           int i = 1; //getLayerByName(layerList, "bainbridge");
           SeLayer theLayer = (SeLayer) layerList.elementAt(i);

           //get feature from a layer
           SeSqlConstruct sqlConstruct = new SeSqlConstruct(theLayer.getName());
           String[] cols = new String[2];
           cols[0] = "name";
           cols[1] = "shape";
           SeQuery query = new SeQuery(conn, cols, sqlConstruct);
           query.prepareQuery();
           query.execute();

           SeRow row = query.fetch();

           //it is always important to release resource when done
           query.close();
           conn.close();

           window.add(jLabel1, BorderLayout.NORTH);
           window.add(jLabel2, BorderLayout.SOUTH); // Size window and make it visible.
           pack();
           setVisible(true);

       }catch (SeException e) {
           e.printStackTrace();
       }
    }

}
```

Part 3: Display feature shape in user interface window, using JBuilder

Now that the link has been established with the server and the data has been fetched using Java, we are now able to build drawing procedures to display the data, which includes three important tasks:

1. Coordinate data structure – [][][]
2. Transform real world coordinates into screen display coordinates
3. Drawing procedures in Java



We will now create a new class in the project file that you created earlier. Name the class, "SDEMap". Use the same import statements as before, shown below. You will notice that the code below has additional code from the previous class, "SDEAccess".

Think through the answers to the following questions while exploring this code:

1. How do you interpret the coordinates for a feature?
2. What are the values of the coordinates?
3. What transformations are needed to display these coordinates on a screen?

```
import javax.swing.*;
import javax.swing.border.*;
import java.awt.*;
import java.util.Vector;
import com.esri.sde.sdk.client.*;

/**
 * <p>Title: ArcSDE access using Java client</p>
 *
 * <p>Description: </p>
 *
 * <p>Copyright: Copyright (c) GEOG 465 2005</p>
 *
 * <p>Department of Geography, University of Washington</p>
 *
 * @author Guirong Zhou
 * @version 1.0
 */
public class SDEMap extends JFrame{
    MapPane jPanel1;
    JLabel jLabel1;
    JLabel jLabel2;
    Polygon theShape;

    /**
     * Constructor:
     *
     * */
    public SDEMap() {
        super("SDE map display");
        setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
        try {
            jbInit();
        } catch (Exception ex) {
            ex.printStackTrace();
```

```
        }
    }

    /**
     * Run this application by instantiating the class
     * */
    public static void main(String[] args) {
        SDEMap sdemap = new SDEMap();
    }

    /**
     * This mutator method will do these steps:
     * <li>Initiate a connection to ArcSDE</li>
     * <li>Get the layers list</li>
     * <li>Get a feature from a layer</li>
     * <li>Display the feature</li>
     * */
    private void jbInit() throws Exception {
        Container window = getContentPane();

        //Add three window controls to the window
        jLabel1 = new JLabel();
        jPanel1 = new MapPane();
        jLabel2 = new JLabel();

        //prepare connection parameters
        SeConnection conn = null;
        String server     = "collab.geog.washington.edu";
        int instance      = 5151;
        String database   = "classdata";
        String user       = "geog465";
        String password   = "geogwin_05";
        try {
           //connect to SDE
           conn = new SeConnection(server, instance, database, user, password);

           //get the list of layers available in this SDE database
           Vector layerList = conn.getLayers();
           jLabel1.setText("There are " + layerList.size() + " layers in this SDE
database.");

           //get a layer
           int i = 1; //getLayerByName(layerList, "bainbridge");
           SeLayer theLayer = (SeLayer) layerList.elementAt(i);

           //get feature from a layer
           SeSqlConstruct sqlConstruct = new SeSqlConstruct(theLayer.getName());
           String[] cols = new String[2];
           cols[0] = "name";
           cols[1] = "shape";
           SeQuery query = new SeQuery(conn, cols, sqlConstruct);
           query.prepareQuery();
           query.execute();

           SeRow row = query.fetch();

           //get the feature's geometry
           SeShape shp = row.getShape(1);
           double coordinates[][][] = shp.getAllCoords();

           //make a polygon from the coordinates:
           theShape = makePolygon(coordinates[0][0]);

           //display an attribute value
           jLabel2.setText("Feature name: " + row.getString(0));

           //it is always important to release resource when done
           query.close();
           conn.close();

           window.add(jLabel1, BorderLayout.NORTH);
           window.add(jPanel1, BorderLayout.CENTER);
           window.add(jLabel2, BorderLayout.SOUTH); // Size window and make it visible.
```

```
            pack();
            setVisible(true);

        }catch (SeException e) {
            e.printStackTrace();
        }
    }

    /**
     * Instantiate a drawable polygon using the given coordinates
     * @param: coords A series of world coordinates as x, y in alternating order
     * @return: A polygon with integer coordinates
     * */
    private Polygon makePolygon(double[] coords){
        double scale = 2000.0;
        double xShift = 122.59359 + 0.01;
        double yShift = -47.72183 - 0.01;

        if(coords.length < 1)return null;

        int xx[] = new int[coords.length/2];
        int yy[] =  new int[coords.length/2];
        for(int i=0; i<coords.length; i=i+2){
            xx[i/2] = (int)((coords[i] + xShift)* scale);
            yy[i/2] = (int)((coords[i+1] + yShift)* scale * (-1));
            System.out.println("(" +  xx[i/2] +  ", " + yy[i/2] +  ")");
        }

        return(new Polygon(xx, yy, coords.length/2));
    }

    /**
     * This class is used to draw the map on a panel
     */
    private class MapPane extends JPanel
    {
        public MapPane(){
            //super();
            this.setPreferredSize(new Dimension(500, 400));
            Border raisedbevel = BorderFactory.createEtchedBorder(EtchedBorder.RAISED);
            Border loweredbevel = BorderFactory.createEtchedBorder(EtchedBorder.LOWERED);
            Border compound = BorderFactory.createCompoundBorder(raisedbevel,
loweredbevel);
            this.setBorder(compound);
            this.setBackground(Color.WHITE);
        }

        protected void paintComponent(Graphics g)
        {
            //g.drawRect(0, 0, 200, 200);
            if(theShape != null){
                g.drawPolygon(theShape);
            }
        }

    }

}
```