# Getting Started with SuperMap Objects Java 6R

**SuperMap Software Co., Ltd.**

**Beijing. China**

# Legal Statement

1. The copyright of this document is covered by SuperMap Software Co., Ltd. in accordance with the Copyright Law of the People's Republic of China and the Universal Copyright Convention. If, without the written permission of the company, any part of the document shall not in any way or any reason be used, copied, modified, transmitted, or bundled with other products to be used, sold, tort reserved.
2. "超图", "SuperMap", and  are the registered trademarks of SuperMap Software Co., Ltd., protected by the Copyright Law of the People's Republic of China. If, without the written permission of the company, the trademarks shall not in any way or any reason be used, copied, modified, transmitted, or bundled with other products to be used, sold, tort reserved.
3. This document represents no responsibilities of any supplier or agent. Without statement, SuperMap Software Co. Ltd. has right to do modifications to this document.
4. The copyright of trademarks mentioned in this document belongs to the corresponding companies. Without the written permission of these companies, the trademarks shall not in any way or any reason be used, copied, modified, or transmitted.
5. The concerned software products and the updated products hereinafter in this document are developed and sold by SuperMap Software Co., Ltd.

Hereby declare

SuperMap Software Co., Ltd.

Add: 7/F Tower B, Technology Fortune Center, No. 8 Xueqing Road,

Haidian District, Beijing, 100192, P. R. China

Tel: +86-10-82736655-4170

Fax: +86-10-82734630

HomePage: www.supermap.com

Sales: request@supermap.com

Tech Support: globalsupport@supermap.com

SuperMap Software welcomes all advices and suggestions from you.

# Preface

## What's the purpose of this book?

In order to make users understand and rapidly grasp SuperMap Objects Java, SuperMap Software Co., Ltd. wrote SuperMap Objects Java 6R Getting Started on the basis of Eclipse 3.2. After reading this book, you should grasp the following contents:

- How to add SuperMap Objects Java component to a project
- How to open SuperMap workspace file
- How to develop a program to browse a map, such as zoom in, zoom out, pan, etc.
- How to query attributes by identifying features
- How to query features by SQL

## Who can use this book?

This book fits the developers who acquire the Java language and for the first time use SuperMap Objects Java. It helps to acquire SuperMap Objects Java component if you know how to develop using SuperMap Objects COM.

## What's the content of this book?

There are three chapters in this book. Chapter One introduces the required installation environment of SuperMap Objects Java, and the data and interfaces used in Chapter Two and Three are also mentioned here. Chapter Two describes how to add SuperMap Objects Java 6R components to a project and how to develop a program to browse a map in Eclipse. The aim of the Chapter Two is to help the developer to rapidly grasp the development method of SuperMap Objects Java. SuperMap Objects Java 6R provides you the source codes for better study. Chapter three introduces how to run these codes in Eclipse developing environment respectively. If you have any other problems or suggestions, please visit our website: **www.supermap.com**，**www.gischina.com**，or contact us directly.

# Content

# 1

# Introduction

## 1.1 Brief introduction

SuperMap Objects Java 6R is a fundamental development platform of the SuperMap GIS Universal series. It is based on Java technology and can be applied to all kinds of application development. SuperMap Objects Java 6R is based on the high-efficiency UGC core and has unique cross-platform capability. SuperMap Objects Java 6R is not an upgraded version of SuperMap Objects, but a component product directly constructed on the basis of UGC, which has reconstructed the architecture, and regulated its interfaces.

SuperMap Objects Java 6R fully supports Java EE standards as well as various integrated developing environments such as Eclipse, JBuilder, NetBeans, etc, and has some new features:

- Cross-platform capability
- Supporting Java EE standards
- High efficiency of UGC core
- Flexible and practical components
- Moderate encapsulation granularity
- Simple and easy-to-use interfaces
- Embedded SDX+
- Suitable for building Java applications on the server side

You can develop the simplest GIS system in a short time by following this book, to realize browsing the map, querying attributes by identifying features and querying

features by SQL.

### 1.1.1. Installation

You can develop applications using SuperMap Objects Java in various developing environments, such as Eclipse. The requirements of the operating system and the configuration of software and hardware are as follows:

**For hardware:**

Minimum hardware requirements:

- CPU: 800MHz processor
- RAM: 256 MB
- Hard Driver Space:10 GB

Recommended hardware requirements:

- CPU: 2.00Ghz processor
- RAM: 1 GB or higher
- Hard Driver Space:40 GB

For the 3D effect displaying, please refer to the following hardware requirements:

Minimum hardware requirements:

- CPU: 2.00Ghz single core processor
- RAM: 512 MB
- Hard Driver Space:10 GB
- Video card：128 MB

Recommended hardware requirements:

- CPU: 3.00Ghz single core processor/2.00Ghz dual core processor
- RAM: 2 GB
- Hard Driver Space: 80 GB
- Video card:3D graphic accelerator with 256MB RAM

NOTE: Do not run the 3D module in the 16 byte color environment.

**For software:**

Operating system requirements:

- Microsoft Windows 2000 (SP4 series)
- Microsoft Windows XP (SP2 series)
- Microsoft Windows Server 2003 (SP1 series)
- Microsoft Windows Vista series
- Microsoft Windows Server 2008 series
- Microsoft Windows 7 series

Other software requirements:

- Visual C++ 2008 redistributable package (Contained in install package)
- JDK 1.5 or higher version
- Eclipse 3.2 or higher version (For developing user)

### 1.1.2. Data

The GIS information distributed by SuperMap Objects Java is the map information. So you need to make a map to perform operations on it before programming. You can use SuperMap Deskpro or applications developed by SuperMap Objects to make the map. As for the flows and methods of making a map, please refer to the documents and help of SuperMap Deskpro. In this book, the map we use is the SampleData – world map. The directory of this map is: SuperMap Objects Java Installation directory\SampleData\world, and its workspace –World.smw includes a map named "world". The demonstration project in this book will use this map, and perform the query on multi-layers.

The program in this book uses SuperMap Objects Java components to open the map,

and performs operations on it. The data used is the world map (World.sdb), the engine type is SDB, and the main layers include World (the word map), Capital (the capitals of countries). The default map to be opened is the world map, and you can change the name of layers to be opened by modifying the codes.

**Tab 1 Data Illustration**

| Name | Illustration |
|---|---|
| World.smw | Workspace of the world map |

### 1.1.3. Classes (including Control) and methods

**Tab 2 Classes and Methods used in Chapter Two and Three**

| Classes | Method | Event |
|---|---|---|
| Workspace Class | getMaps Method<br>open Method | |
| WorkspaceConnectionInfo Class | setType Method<br>setServer Method | |
| MapControl Control | getMap Method<br>setAction Method | |
| Recordset Class | getFieldCount Method<br>getRecordCount Method<br>getFieldValue Method<br>getFieldInfos Method<br>moveNext Method | |
| Layers Class | getCount Method<br>get Method | |
| Layer Class | getSelection Method<br>getDataset Method | |
| Map Class | viewEntire Method<br>setWorkspace Method<br>refresh Method<br>getLayers Method | |
| Maps Class | get Method | |
| Action Class | SELECT2 Field | |

| | | |
|---|---|---|
| | PAN Field<br>ZOOMIN Field<br>ZOOMOUT Field<br>ZOOMFREE Field | |
| Selection Class | getCount Method<br>fromRecordset Method<br>toRecordset Method | |
| DatasetVector Class | Open Method | |
| QueryParameter Class | setCursorType Method<br>setAttributeFilter Method<br>disposeMethod | |
| FieldInfos Class | get Method | |
| FieldInfo Class | getName Method | |
| CursorType Class | STATIC Field | |

## 1.2 Brief summary

If this is your first time to develop a program on SuperMap Objects Java platform, in the following chapters, you will learn how to use MapControl, Workspace, etc. to develop a simple application to perform map browser and other operations.

# 2

# Getting Started in Eclipse

## 2.1 Step One: Create a new project

The following steps show you how to create a project in Eclipse 3.2, and develop an application by SuperMap Objects Java.

### 2.1.1. Configure the workspace

Start Eclipse 3.2 and configure the workspace, as the following illustration shows:

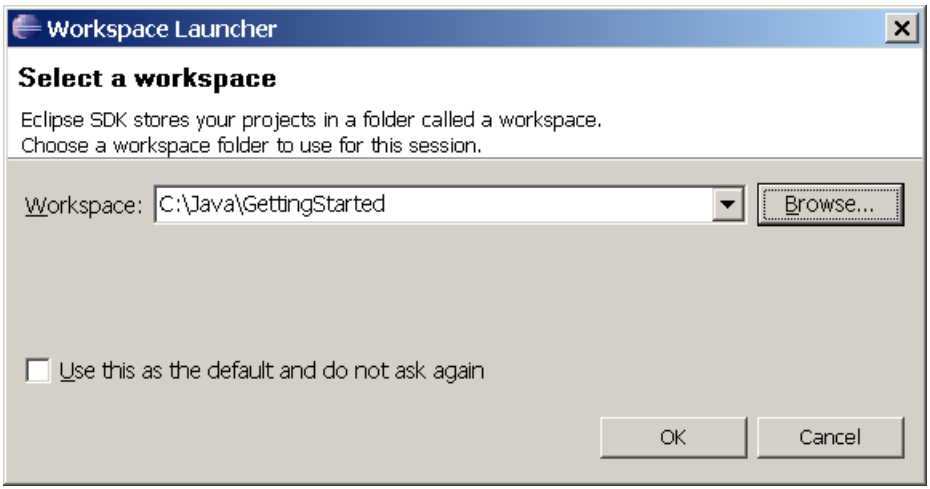



Illustration 1 Select the path for the workspace

### 2.1.2. Create a new project

Click OK in Illustration 1, and enter the Eclipse developing environment in Illustration 2.

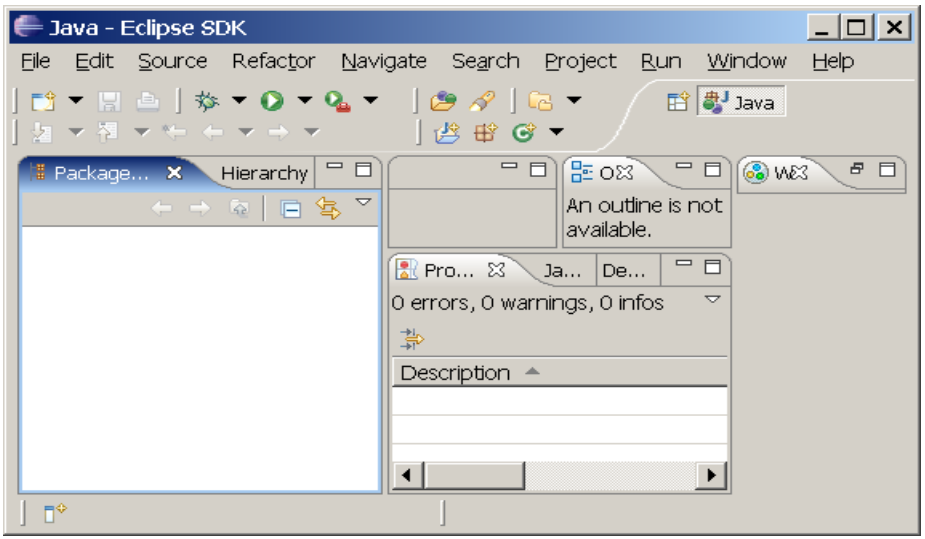



Illustration 2 Eclipse developing environment

Click **File** > **New** > **Project**…to display the project wizard. Select Java Project in

Illustration 3, and click **Next**. Name the new project GettingStarted in Illustration 4.

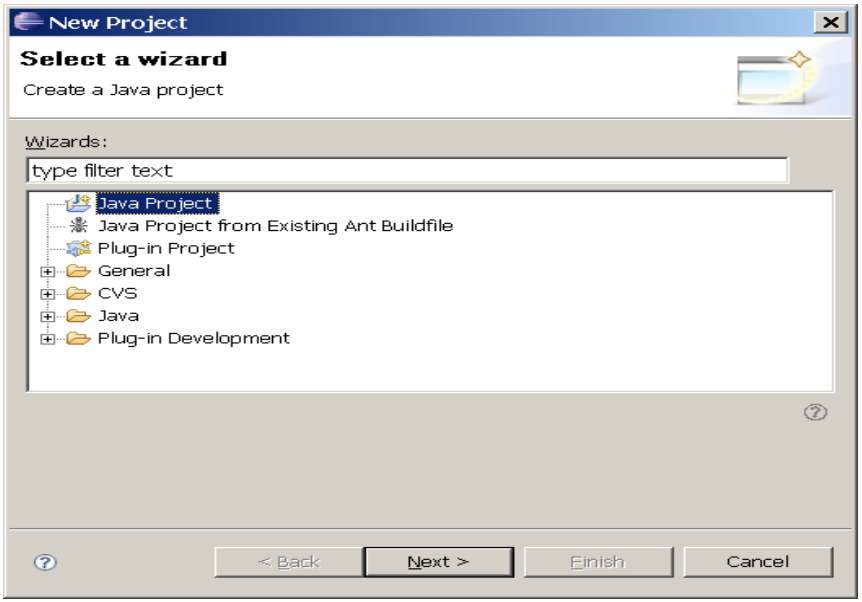



Illustration 3 Select a wizard

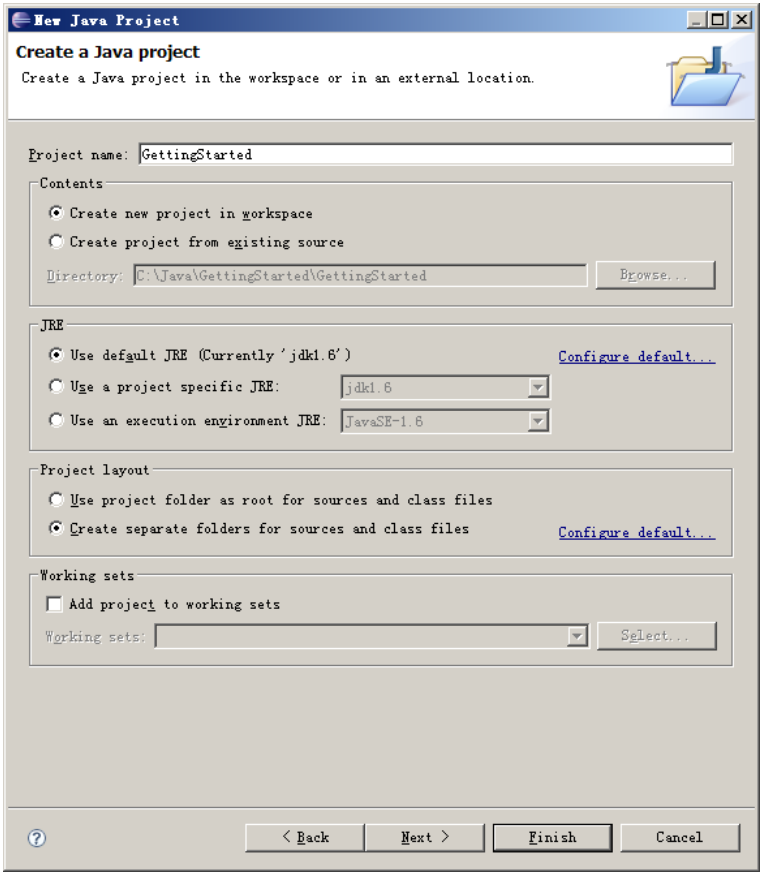



Illustration 4 Name the new project

Click **Next** in Illustration 4 to display Illustration 5**.** Select Libraries, and click **Add External JARs**… .

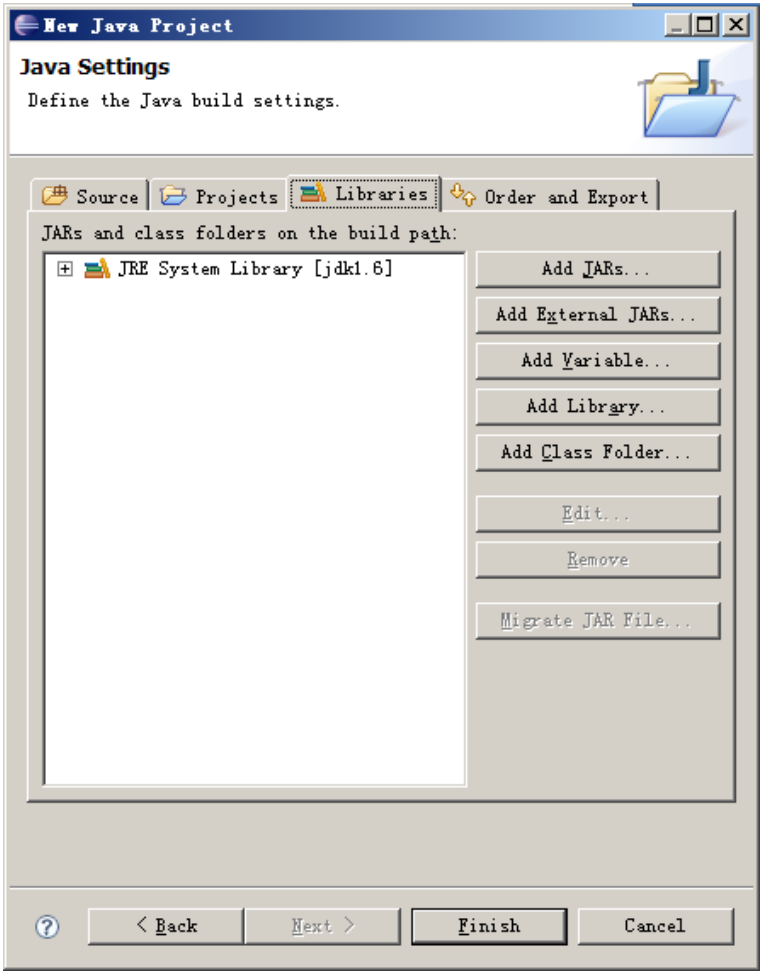



Illustration 5 Display Java settings page

In the pop-up dialogue box, select .jar files in the Bin folder as shown in Illustration 6.

Click **Open**.

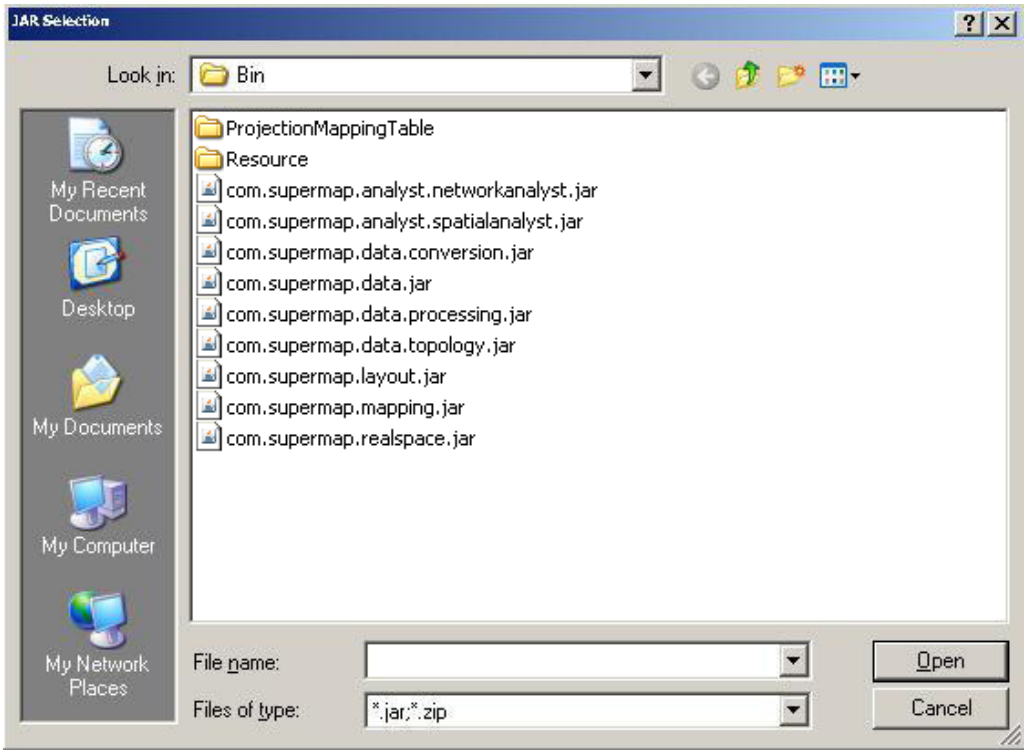



Illustration 6 Select JARs

Illustration 7 shows the result after adding .jar files. You can continue adding .jar files by Add External JARs. You can also edit or remove these files by clicking the **Edit**… or **Remove**….

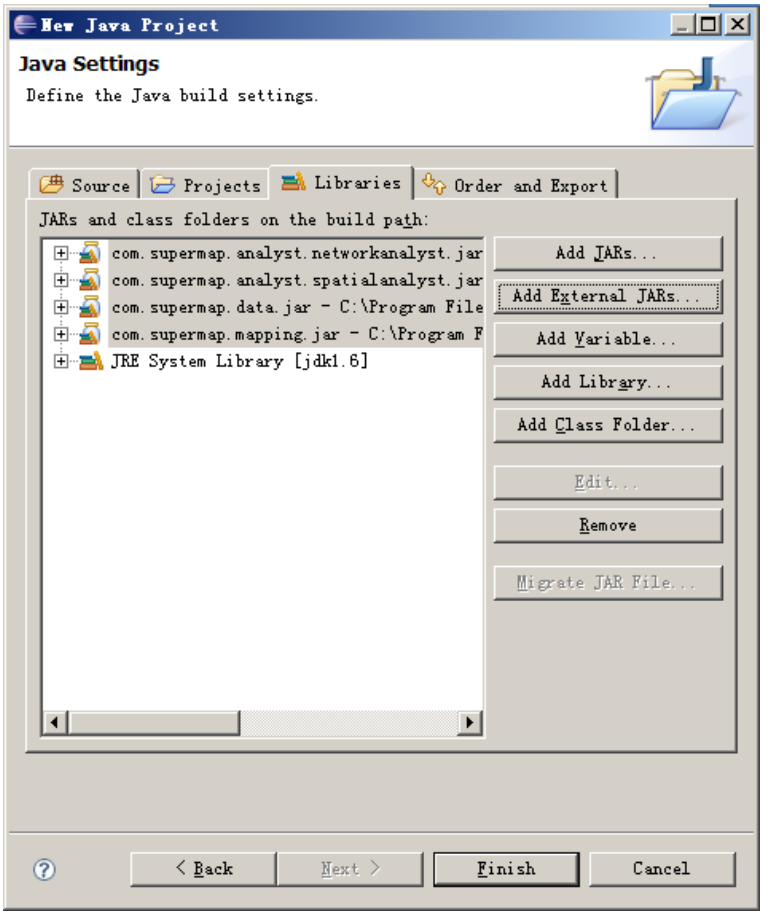



Illustration 7 The result of adding External JARs

Click Finish and the new project is created.

## 2.2 Step Two: Create a Java file

Click **File > New > Package…**, name gettingstarted for the package file, as shown in Illustration 8. Click **Finish**.

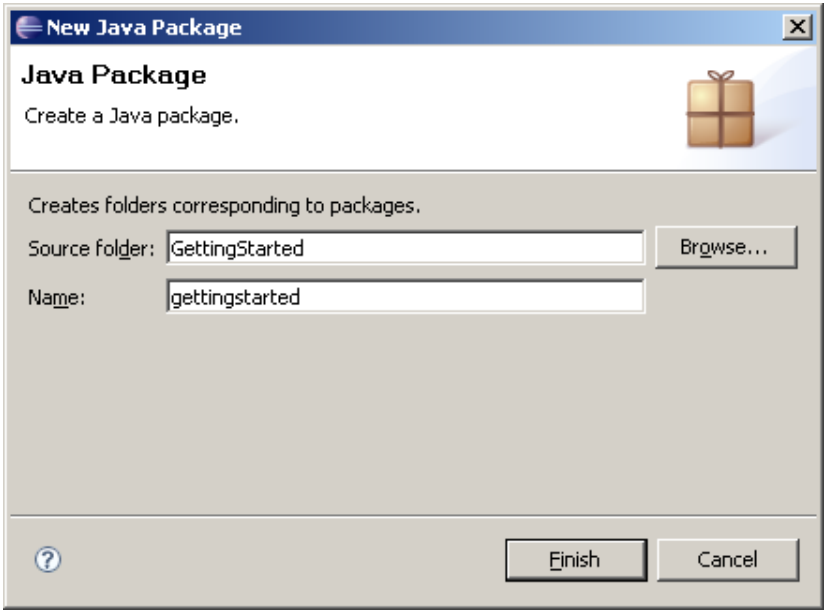



Illustration 8 Name the package

Click **File** > **New** > **Class…**, select GettingStarted folder as the Source folder in the page pops up, name the Java folder as Frame1 and click Finish(as shown in Illustration 9). The Java file is created.

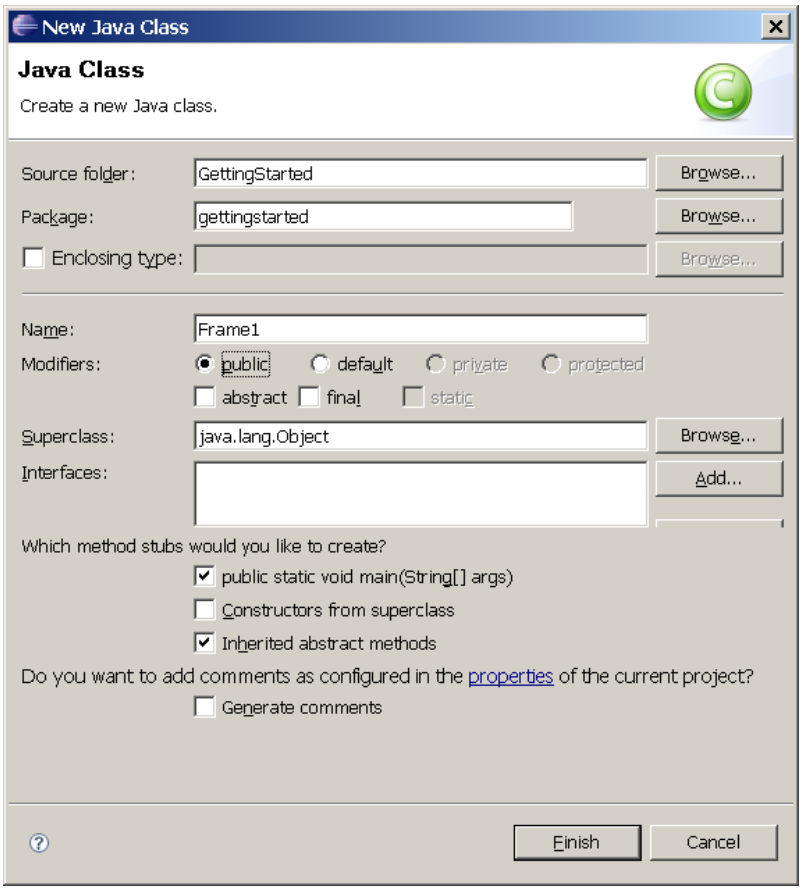



Illustration 9 Name the Java file

Illustration 10 shows the contents of the newly created Java file:

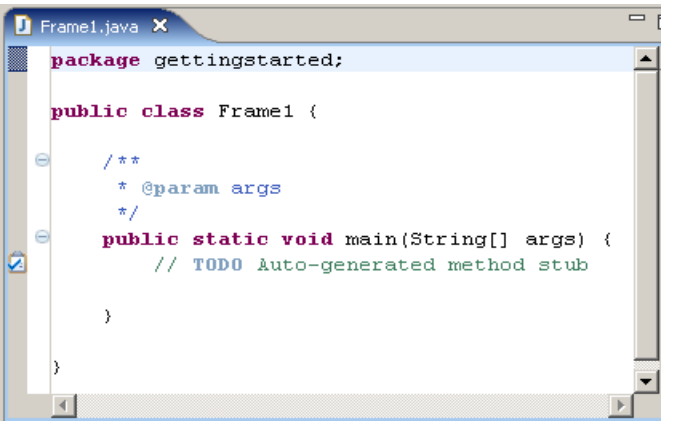



Illustration 10 The contents of the newly created Java file

## 2.3 Step Three: Design interface

The output window is shown in Illustration 11. The buttons, from left to right, are used to open the workspace, pan the map, zoom out on the map, zoom in on the map, zoom free on the map, view the entire map, select, query attributes by identifying features, query filter, and query features by SQL.

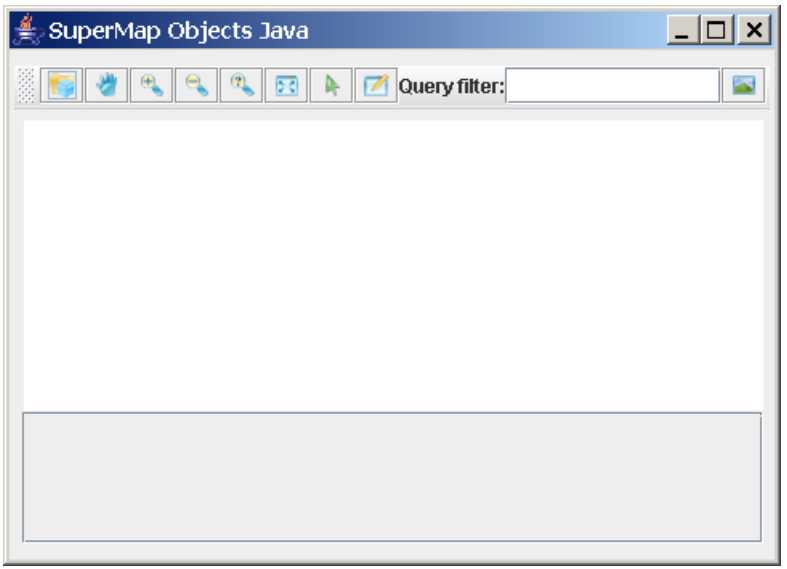



Illustration 11 Output window

The background icon for each button is stored in the installation directory\ GettingStarted\ Eclipse\GettingStarted\Resources\. Import the background icon by the following steps.

Click **File>New>Folder…,** and select GettingStarted in the pop-up page. In the Folder name area, type the name: Resources (see Illustration 12).

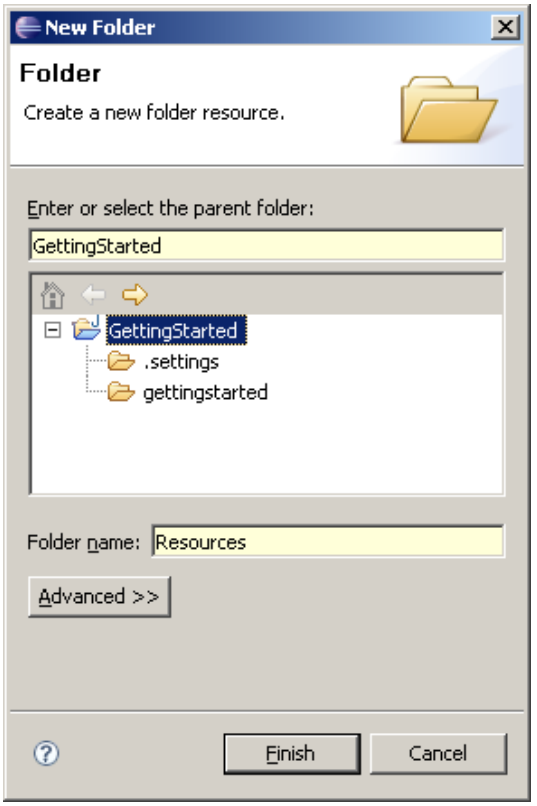



Illustration 12 Create a new folder

Click **Finish** and the Resources folder is added to the directory tree. Keep the folder selected. Click **File** > **Import**…, select File System in the General folder, and click **Next**.

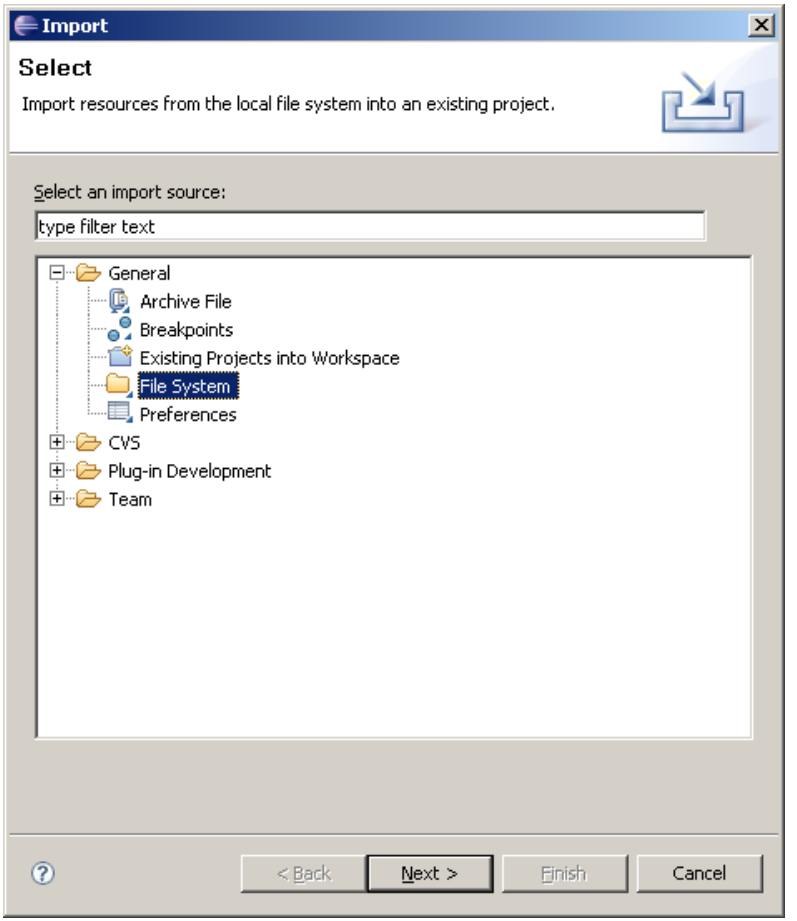



Illustration 13 Import the icon (I)

Set the directory to be installation directory\GettingStarted\Eclipse\GettingStarted at From directory and select Resources (as is circled in orange in Illustration 14). Click **Browse**… button next to Info folder and select the newly created folder named Resources.

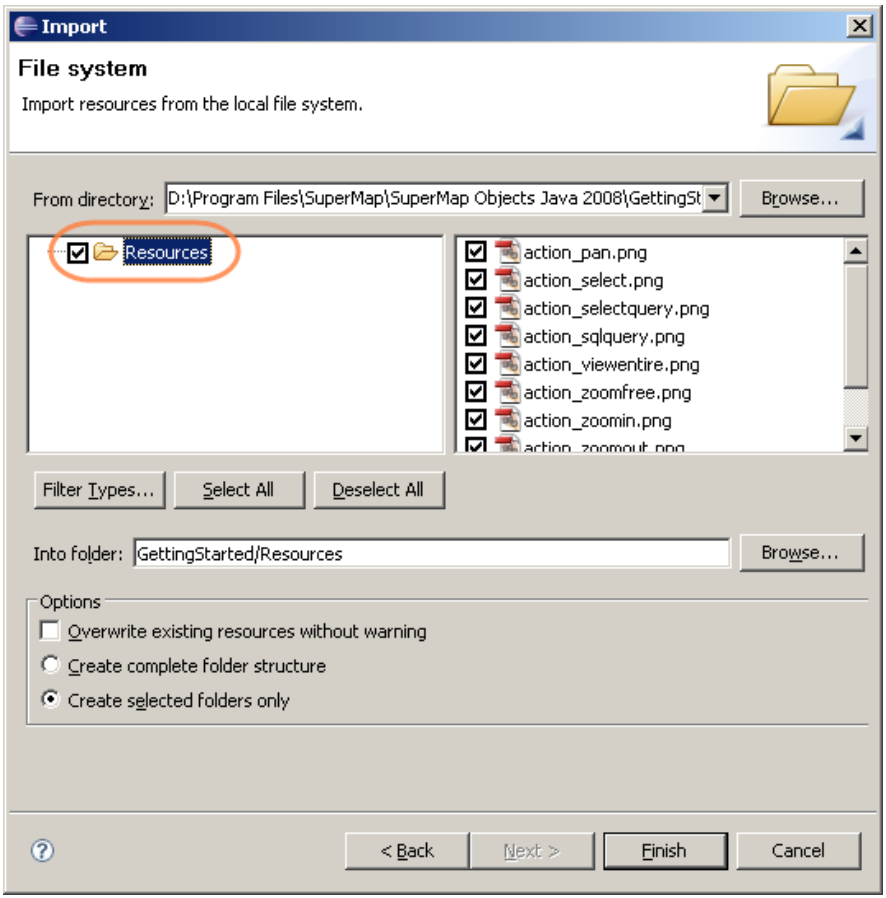



Illustration 14 Import the icon (II)

Use the following code to import necessary packages before you start to design the window (the following code should be written below **package** `gettingstarted`):

```
import com.supermap.data.*;
import com.supermap.mapping.*;
import com.supermap.ui.*;
import com.supermap.ui.Action;

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

```
import javax.swing.table.*;
```

### 2.3.1. Programming

The code for interface design is as follows, and use the code to replace the **public class** Frame1function in Illustration 10:

```
// Instantiates the class used in the following program.
public class Frame1 extends JFrame {
    // Declares and instantiates a MapControl.
    MapControl mapControl1 = new MapControl();
    // Declares and instantiates a Workspace.
    Workspace workspace = null;
    public Workspace getWorkspace(){
        if (workspace == null){
            workspace = new Workspace();
        }
        return workspace;
    }
    JPanel contentPane;
    JButton jButtonOpen = new JButton();
    JButton jButtonPan = new JButton();
    JButton jButtonZoomIn = new JButton();
    JButton jButtonZoomOut = new JButton();
    JButton jButtonZoomFree = new JButton();
    JButton jButtonViewEntire = new JButton();
    JButton jButtonSelect = new JButton();
    JButton jButtonSelectQuery = new JButton();
    JScrollPane jScrollPane1 = new JScrollPane();
    JTable table = new JTable();
    JLabel lblTag = new JLabel();
    JButton jButtonSQLQuery = new JButton();
```

```
    JTextField txtFilter = new JTextField();
    JToolBar jToolBar1 = new JToolBar();

    // Sets the icons
    ImageIcon selection = new ImageIcon("Resources/action_select.png");
    ImageIcon openicon = new ImageIcon("Resources/workspace_open.png");
    ImageIcon panicon = new ImageIcon("Resources/action_pan.png");
    ImageIcon zoominicon = new ImageIcon("Resources/action_zoomin.png");
    ImageIcon zoomouticon = new
ImageIcon("Resources/action_zoomout.png");
    ImageIcon zoomfreeicon = new
ImageIcon("Resources/action_zoomfree.png");
    ImageIcon viewentireicon = new
ImageIcon("Resources/action_viewentire.png");
    ImageIcon selectqueryicon = new
ImageIcon("Resources/action_selectquery.png");
    ImageIcon sqlqueryicon= new
ImageIcon("Resources/action_sqlquery.png");

    public Frame1() {
        try {
            setDefaultCloseOperation(EXIT_ON_CLOSE);
            jbInit();
        } catch (Exception exception) {
            exception.printStackTrace();
        }
    }

    // Designs the interface。
    private void jbInit() throws Exception {

        // Sets the default window size and window name.
        contentPane = (JPanel) getContentPane();
```

```
        setSize(new Dimension(485, 350));
        setTitle("SuperMap Objects Java");
        this.addWindowListener(new Frame1_this_windowAdapter(this));
        this.addComponentListener(new
Frame1_this_componentAdapter(this));

        // The default size of jScrollPane.
        jScrollPane1.setBounds(new Rectangle(8, 226, 462, 82));

        // Adds MapControl to the window.
        contentPane.setLayout(null);
        mapControl1.setLayout(null);
        mapControl1.setBounds(new Rectangle(9, 44, 461, 181));

        // Adds JToolBar and add the following controls to the JtoolBar.
        jToolBar1.setBounds(new Rectangle(3, 10, 471, 27));
        contentPane.add(jToolBar1);
        jToolBar1.add(jButtonOpen);
        jToolBar1.add(jButtonPan);
        jToolBar1.add(jButtonZoomIn);
        jToolBar1.add(jButtonZoomOut);
        jToolBar1.add(jButtonZoomFree);
        jToolBar1.add(jButtonViewEntire);
        jToolBar1.add(jButtonSelect);
        jToolBar1.add(jButtonSelectQuery);
        jToolBar1.add(lblTag);
        jToolBar1.add(txtFilter);
        jToolBar1.add(jButtonSQLQuery);
        contentPane.add(mapControl1, null);
        contentPane.add(jScrollPane1);
        jScrollPane1.getViewport().add(table);

        // Adds Open button and sets the icon
```

```
        jButtonOpen.setToolTipText("Open");
        jButtonOpen.setIcon(openicon);

        // Adds pan button and sets the icon
        jButtonPan.setToolTipText("Pan the map");
        jButtonPan.setIcon(panicon);

        // Adds zoon in button and sets the icon
        jButtonZoomIn.setToolTipText("Zoom in on the map");
        jButtonZoomIn.setIcon(zoominicon);

        // Adds zoom out button and sets the icon
        jButtonZoomOut.setToolTipText("Zoom out on the map");
        jButtonZoomOut.setSelectedIcon(null);
        jButtonZoomOut.setIcon(zoomouticon);

        // Adds zoom free button and sets the icon
        jButtonZoomFree.setToolTipText("Zoom free on the map");
        jButtonZoomFree.setIcon(zoomfreeicon);

        // Adds view entire map button and sets the icon
        jButtonViewEntire.setToolTipText("View the entire map");
        jButtonViewEntire.setIcon(viewentireicon);

        // Adds select button and sets the icon
        jButtonSelect.setToolTipText("Select the features");
        jButtonSelect.setIcon(selection);

        // Adds select query button and sets the icon
        jButtonSelectQuery.setToolTipText("Query attributes by
identifying feature");
        jButtonSelectQuery.setIcon(selectqueryicon);
```

```
        // Adds SQL query button and sets the icon
        lblTag.setToolTipText("");
        lblTag.setText("Query filter ");
        jButtonSQLQuery.setToolTipText("Query features by SQL");
        jButtonSQLQuery.setIcon(sqlqueryicon);
}

    // Sets the map and attribute table resizing with the window size.
    public void this_componentResized(ComponentEvent e) {
        mapControl1.setSize(this.getWidth() - 25,
            7 * (int)this.getHeight() / 12);

        jScrollPane1.setLocation(10, 7 * (int)this.getHeight() / 12 + 46);
        jScrollPane1.setSize(this.getWidth() - 25,
             (int)5 * (int)this.getHeight() / 12 - 80);
        table.setLocation(10,
            7 * (int)this.getHeight() / 12 + 46);
        table.setSize(this.getWidth() - 25,
            (int)this.getHeight() / 2 - 130);
    }

    // For UDB datasource, you need to close MapControl and then close the
Workspace.
     public void this_windowClosing(WindowEvent e) {
        if (mapControl1 != null) {
            mapControl1.dispose();
             mapControl1 = null;
        }
        if (workspace != null) {
            workspace.dispose();
            workspace = null;
        }
    }
```

```
    // Main function
    public static void main(String[] args) {
        Frame1 frame = new Frame1();

        // Displays the window in the middle of the screen.
        Dimension screenSize =
Toolkit.getDefaultToolkit().getScreenSize();
        Dimension frameSize = frame.getSize();
        if (frameSize.height > screenSize.height) {
            frameSize.height = screenSize.height;
        }
        if (frameSize.width > screenSize.width) {
            frameSize.width = screenSize.width;
        }
    frame.setLocation((screenSize.width - frameSize.width) / 2,
                (screenSize.height - frameSize.height) / 2);
    frame.setVisible(true);
    }
// Event code.
class Frame1_this_componentAdapter extends ComponentAdapter {
    private Frame1 adaptee;

    Frame1_this_componentAdapter(Frame1 adaptee) {
        this.adaptee = adaptee;
    }

    public void componentResized(ComponentEvent e) {
        adaptee.this_componentResized(e);
    }
}
class Frame1_this_windowAdapter extends WindowAdapter {
    private Frame1 adaptee;
```

```
    Frame1_this_windowAdapter(Frame1 adaptee) {
        this.adaptee = adaptee;
    }
     public void windowClosing(WindowEvent e) {
        adaptee.this_windowClosing(e);
    }
}
}
```

Note: If any error occurs, please check the coding method of the program, and change it to UTF-8 by checking **Project > Properties > Info > Text file encoding > Other.**

### 2.3.2. Running

Click Frame1.java, select Run as> Java Application, and the running result is shown in Illustration 11.

## 2.4 Step Four: Open a map

### 2.4.1. Programming

Here we take SMW workspace as an example. For how to open other kinds of workspace, please refer to the corresponding help document.

1. Add the following code before the main function:

```
public void jButtonOpen_actionPerformed(ActionEvent e) {

    // Creates a file dialog window for loading a Workspace file.
    FileDialog opendialog = new FileDialog(this, "Open a workspace",
FileDialog.LOAD);
    opendialog.setVisible(true);
```

```
    // Instantiates a Workspace by getWorkspace() method.
    workspace = getWorkspace();

    // Instantiates a WorkspaceConnectionInfo.
    WorkspaceConnectionInfo workspaceconnect = new
WorkspaceConnectionInfo();

    // Sets the type of the workspace.
    workspaceconnect.setType(WorkspaceType.SMW);

    // Sets the name of the path used to connect the workspace.
    String filename = opendialog.getFile();
    String filedir = opendialog.getDirectory();
    String fileworkspace = filedir + filename;
    workspaceconnect.setServer(fileworkspace);

    // Determines whether successfully open the workspace or not.
    boolean bOpen = supermapSpace.open(workspaceconnect);
    if (bOpen == false) {
        System.out.println("Fail to open the workspace!");
        return;
    }

    // Connects MapControl and Workspace.
    mapControl1.getMap().setWorkspace(supermapSpace);

    // Determines whether there are maps in the workspace, if does get
the Maps in the workspace.
    Maps maps = null;
    if (supermapSpace.getMaps().getCount() <= 0) {
    JOptionPane.showMessageDialog(this, "There is no map in the
 workspace!");
```

```
            return;
        } else {
            maps = supermapSpace.getMaps();
        }

        // Opens the map according to its name.
        mapControl1.getMap().open(maps.get(0));
        mapControl1.getMap().refresh();

        //Disables the button for opening the map
        jButtonOpen.setEnabled(false);

}
```

2. Add the following code after the event code:

```
class Frame1_jButton1_actionAdapter implements ActionListener {

    private Frame1 adaptee;
    Frame1_jButton1_actionAdapter(Frame1 adaptee) {
        this.adaptee = adaptee;
    }

    public void actionPerformed(ActionEvent e) {
        adaptee.jButtonOpen_actionPerformed(e);
    }
}
```

3. Add the following code before closing jbInit() of Frame1:

```
    jButtonOpen.addActionListener(new
Frame1_jButton1_actionAdapter(this));
```

### 2.4.2. Running

The running result is as follows:

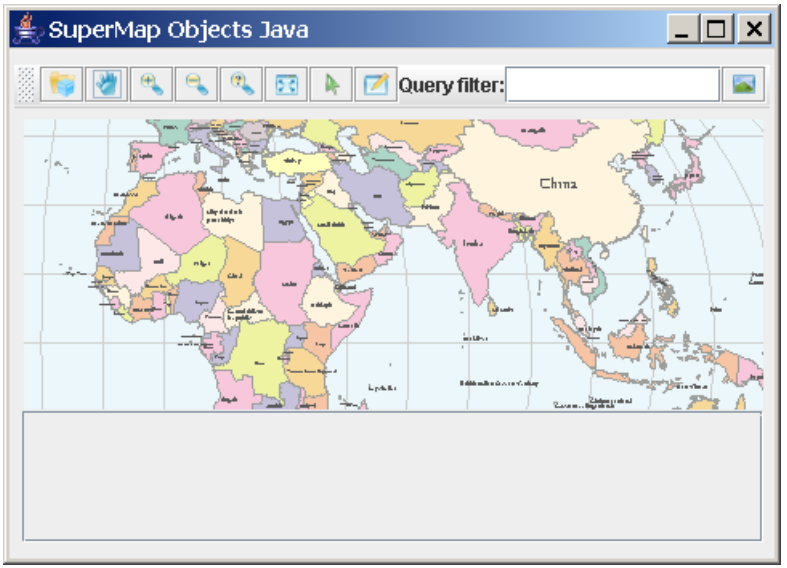



Illustration 15 Open a map

## 2.5 Step Five: Browse the map

### 2.5.1. Programming

1. Add the following code before the main function:

```
public void jButtonZoomIn_actionPerformed(ActionEvent e) {


    // Zooms in on the map.
    mapControl1.setAction(Action.ZOOMIN);
}
```

```
    public void jButtonPan_actionPerformed(ActionEvent e) {

        // Pans the map.
        mapControl1.setAction(Action.PAN);
    }

    public void jButtonZoomOut_actionPerformed(ActionEvent e) {

        // Zooms out on the map.
        mapControl1.setAction(Action.ZOOMOUT);
    }

    public void jButtonZoomFree_actionPerformed(ActionEvent e) {

        // Zooms free on the map.
        mapControl1.setAction(Action.ZOOMFREE);
    }

    public void jButtonViewEntire_actionPerformed(ActionEvent e) {

        // Views the entire map.
        mapControl1.getMap().viewEntire();
        mapControl1.getMap().refresh();
    }

    public void jButtonSelect_actionPerformed(ActionEvent e) {

        // Select
        mapControl1.setAction(Action.SELECT2);
    }
```

2. Add the following code after the event code:

```
class Frame1_jButtonSelect_actionAdapter implements
```

```
ActionListener {

    private Frame1 adaptee;
    Frame1_jButtonSelect_actionAdapter(Frame1 adaptee) {
        this.adaptee = adaptee;
    }

    public void actionPerformed(ActionEvent e) {
        adaptee.jButtonSelect_actionPerformed(e);
    }
}


class Frame1_jButtonViewEntire_actionAdapter implements
ActionListener {
    private Frame1 adaptee;
    Frame1_jButtonViewEntire_actionAdapter(Frame1 adaptee) {
        this.adaptee = adaptee;
    }

    public void actionPerformed(ActionEvent e) {
        adaptee.jButtonViewEntire_actionPerformed(e);
    }
}


class Frame1_jButtonZoomFree_actionAdapter implements
ActionListener {
    private Frame1 adaptee;
    Frame1_jButtonZoomFree_actionAdapter(Frame1 adaptee) {
        this.adaptee = adaptee;
    }

    public void actionPerformed(ActionEvent e) {
```

```
        adaptee.jButtonZoomFree_actionPerformed(e);
    }
}


class Frame1_jButtonZoomOut_actionAdapter implements
ActionListener {
    private Frame1 adaptee;
    Frame1_jButtonZoomOut_actionAdapter(Frame1 adaptee) {
        this.adaptee = adaptee;
    }

    public void actionPerformed(ActionEvent e) {
        adaptee.jButtonZoomOut_actionPerformed(e);
    }
}


class Frame1_jButtonPan_actionAdapter implements
ActionListener {
    private Frame1 adaptee;
    Frame1_jButtonPan_actionAdapter(Frame1 adaptee) {
        this.adaptee = adaptee;
    }

    public void actionPerformed(ActionEvent e) {
        adaptee.jButtonPan_actionPerformed(e);
    }
}


class Frame1_jButtonZoomIn_actionAdapter implements
ActionListener {
```

```
    private Frame1 adaptee;
    Frame1_jButtonZoomIn_actionAdapter(Frame1 adaptee) {
        this.adaptee = adaptee;
    }

    public void actionPerformed(ActionEvent e) {
        adaptee.jButtonZoomIn_actionPerformed(e);
    }
}
```

3. Add the following code before closing jbInit() of Frame1:

```
        jButtonPan.addActionListener(new
Frame1_jButtonPan_actionAdapter(this));

        jButtonZoomIn.addActionListener(new
Frame1_jButtonZoomIn_actionAdapter(this));
        jButtonZoomOut.addActionListener(new
Frame1_jButtonZoomOut_actionAdapter(this));
        jButtonZoomFree.addActionListener(new
Frame1_jButtonZoomFree_actionAdapter(this));
        jButtonViewEntire.addActionListener(new
Frame1_jButtonViewEntire_actionAdapter(this));
        jButtonSelect.addActionListener(new
Frame1_jButtonSelect_actionAdapter(this));
```

### 2.5.2. Running

The running result is as follows:

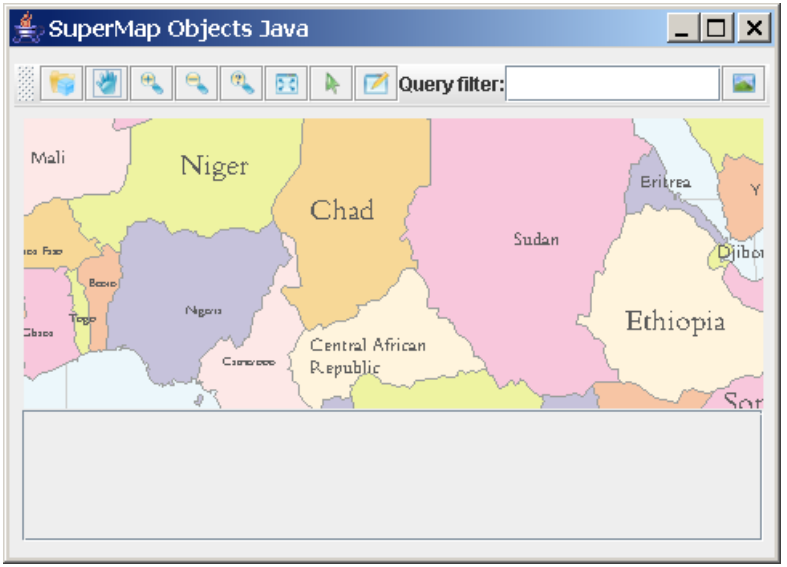



Illustration 16 Zoom in on a map

## 2.6 Step Six: Query attributes by identifying features

### 2.6.1. Programming

1. Add the following code before the main function:

```
public void jButtonSelectQuery_actionPerformed(ActionEvent e) {

    // Declares the Recordset.
    Recordset recordset;

    // Gets the layer set.
    Layers layers = mapControl1.getMap().getLayers();
    // Gets the selection.
    Selection selection = null;
    for (int i = 0; i < layers.getCount(); i++) {
        if (layers.get(i).getSelection().getCount() != 0) {
            selection = layers.get(i).getSelection();
```

```
            }
        }

        // Determines whether the Selection is empty or not.
        if ((selection == null) || (selection.getCount() == 0)) {
            JOptionPane.showMessageDialog(this, " Please select the
feature to be queried.");
            return;
        }

        // Converts selection to recordset.
        recordset = selection.toRecordset();

        // Gets the count of the fields of the record in the recordset.
        int fieldcount = recordset.getFieldCount();

        // Gets the count of the records.
        int recordcount = recordset.getRecordCount();

        // Declares and instantiates the table
        Object[][] tableValues = new Object[recordcount][fieldcount];
        Object[] tableColumns = new Object[fieldcount];

        // Adds the field value to the jTable control
        for (int j = 0; j < fieldcount; j++) {
            tableColumns[j] =
recordset.getFieldInfos().get(j).getName();
        }

        for (int i = 0; i < recordcount; i++) {
            for (int j = 0; j < fieldcount; j++) {
                tableValues[i][j] = recordset.getFieldValue(j);
            }
```

```
            recordset.moveNext();
        }
        recordset.dispose();
        recordset = null;

        // Displays in the table
        DefaultTableModel tableModel = new
DefaultTableModel(tableValues,
                tableColumns);
        this.table.setModel(tableModel);
    }
```

2. Add the following code after the event code:

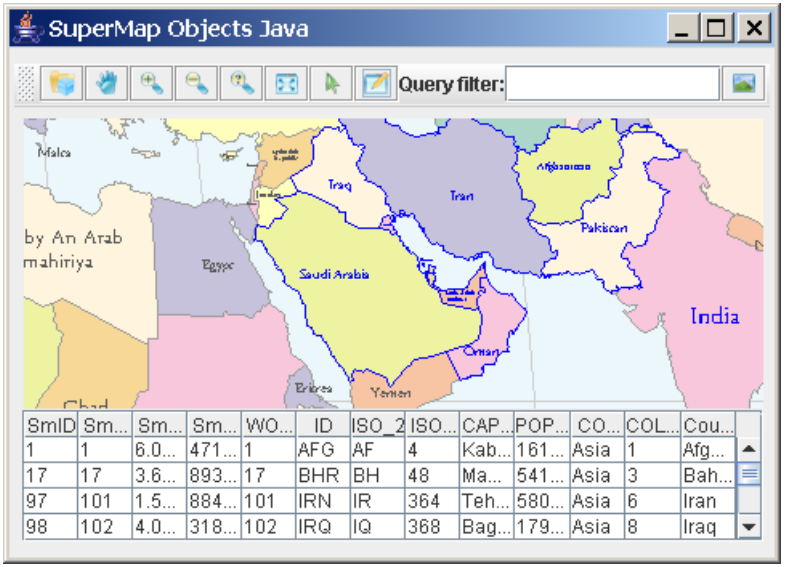
```
    class Frame1_jButtonSelectQuery_actionAdapter implements
ActionListener {

        private Frame1 adaptee;
        Frame1_jButtonSelectQuery_actionAdapter(Frame1 adaptee) {
            this.adaptee = adaptee;
        }
        public void actionPerformed(ActionEvent e) {
            adaptee.jButtonSelectQuery_actionPerformed(e);
        }
    }
```

3. Add the following code before closing jbInit() of Frame1:

```
        jButtonSelectQuery.addActionListener(new
Frame1_jButtonSelectQuery_actionAdapter(this));
```

### 2.6.2. Running

The running result is as follows:




| SmID | Sm... | Sm... | Sm... | WO... | ID | ISO_2 | ISO... | CAP... | POP... | CO... | COL... | Cou... |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 6.0... | 471... | 1 | AFG | AF | 4 | Kab... | 161... | Asia | 1 | Afg... |
| 17 | 17 | 3.6... | 893... | 17 | BHR | BH | 48 | Ma... | 541... | Asia | 3 | Bah... |
| 97 | 101 | 1.5... | 884... | 101 | IRN | IR | 364 | Teh... | 580... | Asia | 6 | Iran |
| 98 | 102 | 4.0... | 318... | 102 | IRQ | IQ | 368 | Bag... | 179... | Asia | 8 | Iraq |

Illustration 17 Query attributes by identifying features

# 2. 7 Step Seven: Query features by SQL

## 2.7.1. Programming

1. Add the following code before the main function:

```
    public void jButtonSQLQuery_actionPerformed(ActionEvent e) {

        // Determines whether the JTextField is empty.
        if (txtFilter.getText().trim().equals("")) {
            JOptionPane.showMessageDialog(this, " There is no query
filter.");
            return;
        }

        // Gets the query filter.
        QueryParameter queryDef = new QueryParameter();
        queryDef.setAttributeFilter(txtFilter.getText());
```

```
        queryDef.setCursorType(CursorType.STATIC);

        // Determines whether there are opened layers.
        int querycount = mapControl1.getMap().getLayers().getCount();
        if (querycount == 0) {
            JOptionPane.showMessageDialog(this, " Please open a vector
dataset before query!");
            return;
        }

        // Traverses each layer to query among multi-layers.
        int flag = 0;
        for (int i = 0; i < querycount; i++) {

        // Gets the image and vector dataset and convert it to the
DatasetVector type.
            Dataset dataset =
            mapControl1.getMap().getLayers().get(i).getDataset();
            if (dataset.getType() == DatasetType.IMAGE) {continue;}
            DatasetVector datasetvector=(DatasetVector)dataset;
            if (datasetvector == null) {
                continue;
             }
             // Queries the vector dataset and selects the attribute data.
             Recordset recordset = datasetvector.query(queryDef);

             // Assigns 1 if the record in the Recordset is not zero.
             if (recordset.getRecordCount() > 0) {
                flag = 1;
             }

             // Adds the queried data to the Selection, and highlights the
data.
```

```
            Selection selection =
mapControl1.getMap().getLayers().get(i).
                                getSelection();
            selection.fromRecordset(recordset);

            // Calls dispose method。
            recordset.dispose();
            recordset = null;
        }
        // Determines whether the query result is empty.
        if (flag == 0) {
            JOptionPane.showMessageDialog(this, " The Recordset is empty
or no record satisfying the query conditon!");
        }
        // Refreshes the map window to display the map.
        mapControl1.getMap().refresh();

        // Releases the occupied resource.
        queryDef.dispose();

    }
```

2. Add the following code after the event code:

```
class Frame1_jButtonSQLQuery_actionAdapter implements ActionListener
{
    private Frame1 adaptee;
    Frame1_jButtonSQLQuery_actionAdapter(Frame1 adaptee) {
        this.adaptee = adaptee;
    }

    public void actionPerformed(ActionEvent e) {
        adaptee.jButtonSQLQuery_actionPerformed(e);
    }
}
```

3. Add the following code before closing jbInit() of Frame1:

```
        jButtonSQLQuery.addActionListener(new
Frame1_jButtonSQLQuery_actionAdapter(this));
```

### 2.7.2. Running

The correct query format is: field name comparison operators compare value, such as SmID = 1. Type the query filter: SmID >150, and then click the button to perform SQL query. The highlighted part on the map is the query result.

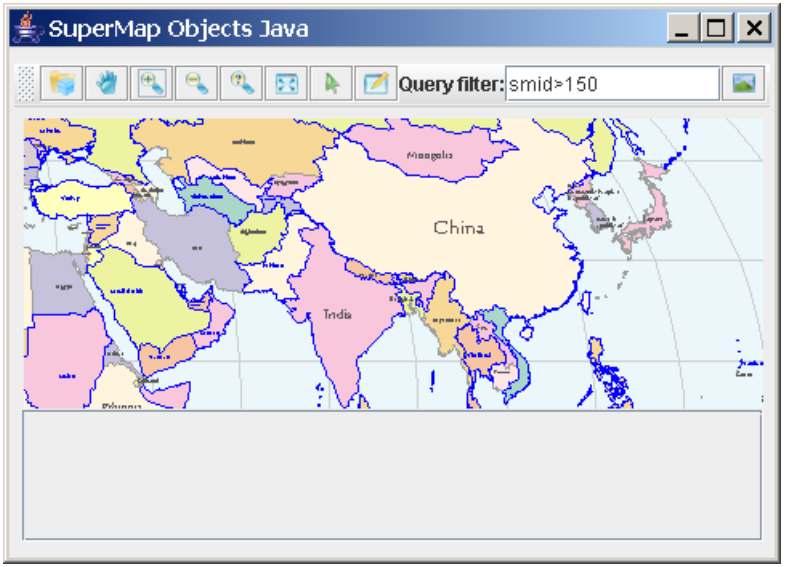



Illustration 18 Query features by SQL

# 3

# Remark

SuperMap Objects Java 6R provides you with the source code of the getting started program. The program is stored in the Getting Started folder under the installation directory. Generally, you can run the source program directly after installation. Otherwise, you can check the path of the JAR package (the correct path of the .jar files: the installation directory\Bin folder). The following part will introduce how to set the path of JAR package for the getting started program.

Run Eclipse, locate the workspace to the Eclipse folder in the GettingStarted under the installation directory.

Select project, and click **File > Properties**. In the pop-up dialogue box, set the properties for the getting started source code. Select Java Build Path in the left index tree and Libraries on the right. Click the **Add External JARs**…as shown in Illustration 19.

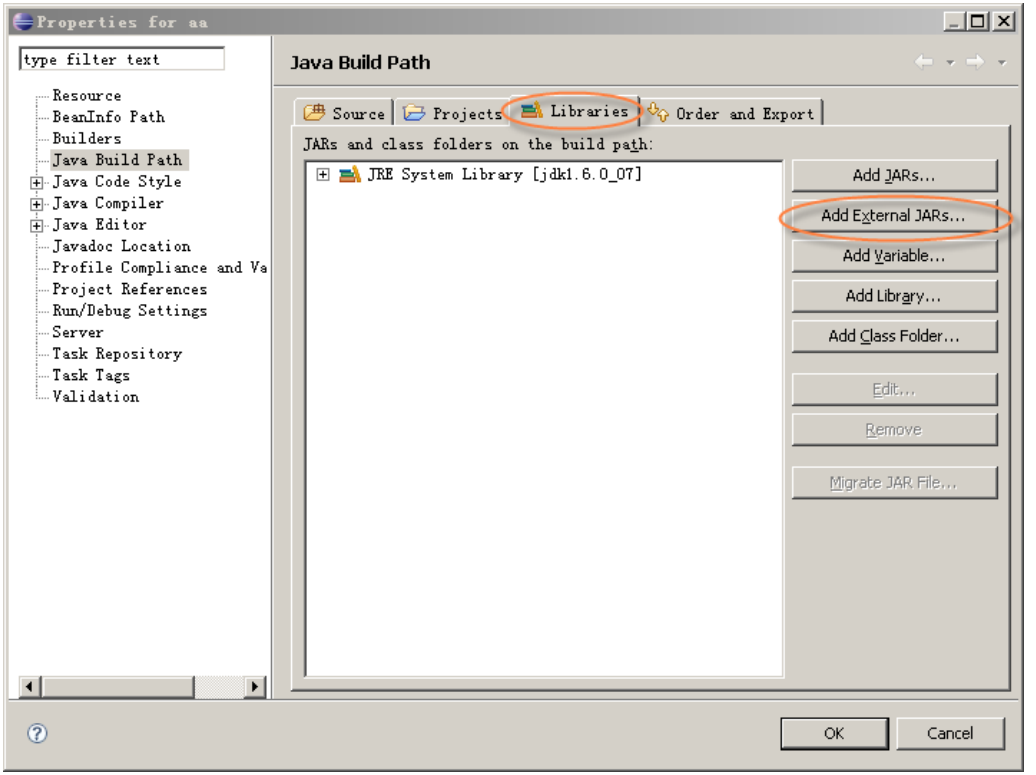



Illustration 19 Add external JAR package

The dialogue box for selecting JAR package pops up, as in Illustration 20. Select .jar files in Bin folder under the installation directory (when running the program, the false .jar files need to be removed by the **Remove** in Illustration 19).

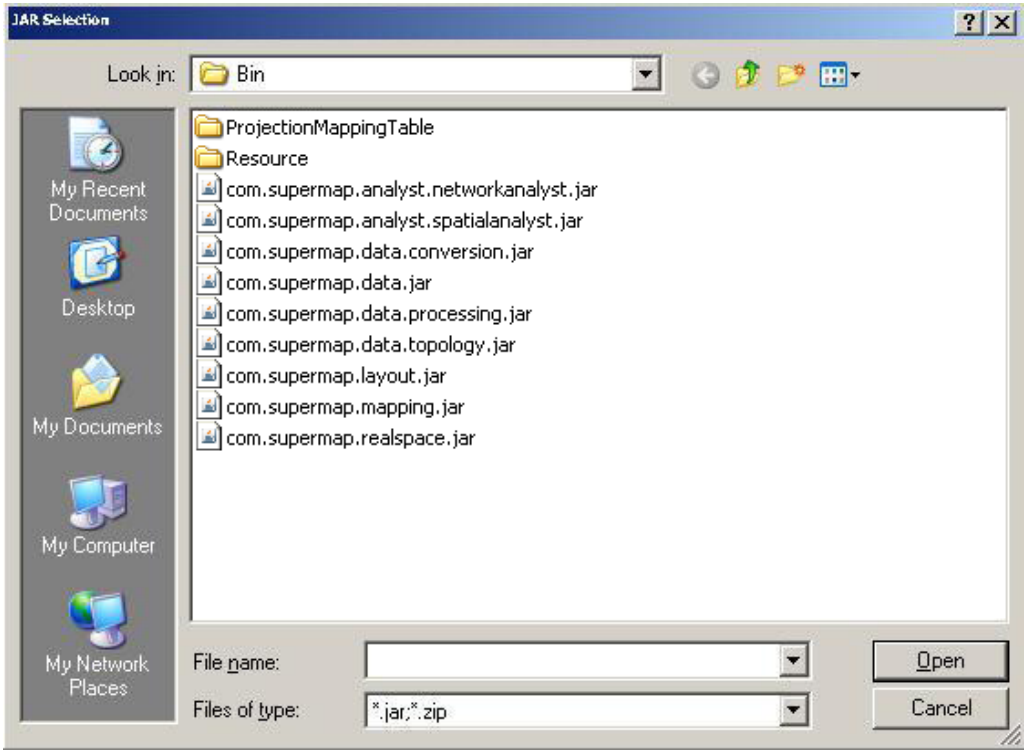



Illustration 20 Selecting the JAR package

After setting the libraries, click **Run > Run As > Java Application,** and Illustration 21 pops up:

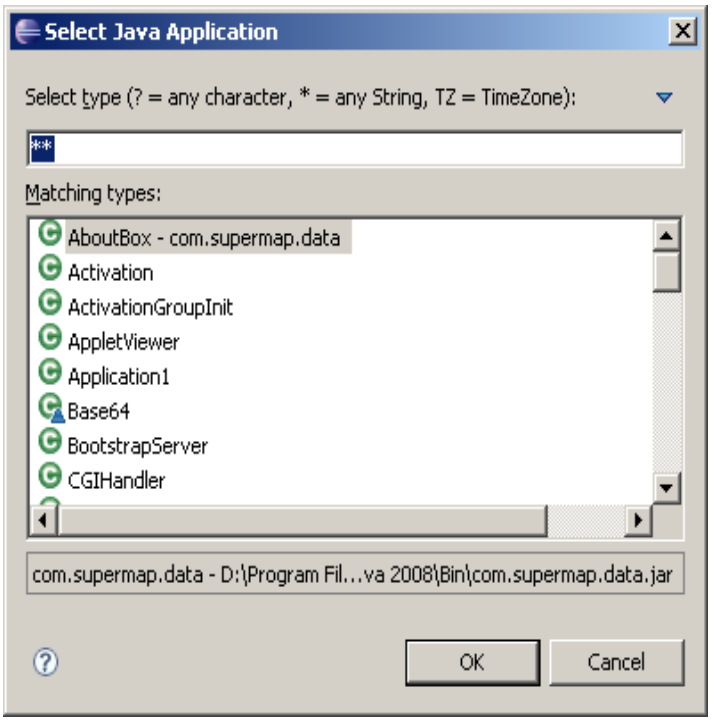



Illustration 21 Select Java application program

Search Application 1, as shown in Illustration 22, select the found file, and then click **OK** to run the getting started program.

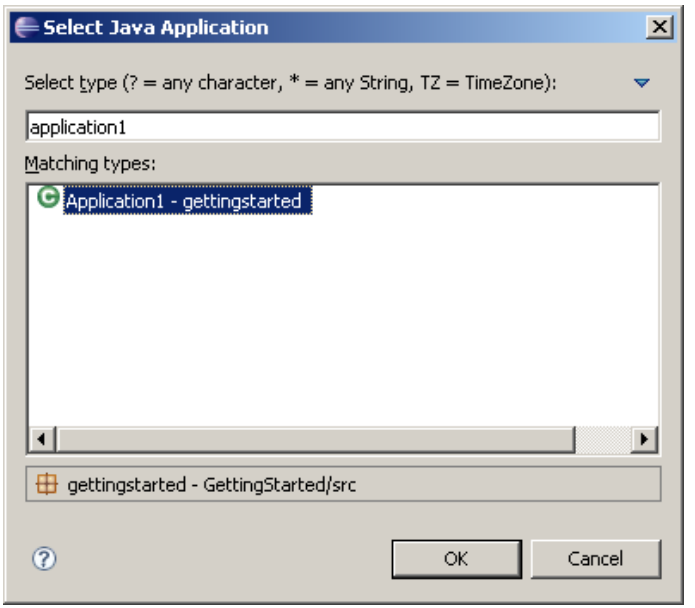



Illustration 22 Select Java application program 2

PS: If you can't run the program smoothly, please check the coding pattern of the program, and change it into **UTF-8 through Project > Properties > Info > Text file encoding > Other.**