```
1718  package com.test;
1719  import java.sql.Connection;
1720  import java.sql.DriverManager;
1721  public class DataUtil
1722  {
1723      private String dbUrl="jdbc:mysql://localhost:3306/smartshelf";
1724      private String dbUserName="root";
1725      private String dbPassWord="root";
1726      private String jdbcName="com.mysql.jdbc.Driver";
1727      public Connection getCon() throws Exception
1728      {
1729          Class.forName(jdbcName);
1730          Connection con=DriverManager.getConnection(dbUrl, dbUserName,dbPassWord);
1731          System.out.println("link database success");
1732          return con;
1733      }
1734      public void closeCon(Connection con) throws Exception
1735      {   if(con!=null)
1736          {
1737          con.close();
1738          }
1739      }
1740  }
1741  package com.test.view;
1742  import java.awt.Font;
1743  import java.awt.event.ActionEvent;
1744  import java.awt.event.ActionListener;
1745  import java.io.File;
1746  import java.io.FileNotFoundException;
1747  import java.io.IOException;
1748  import java.util.Collection;
1749  import java.util.HashMap;
1750  import java.util.LinkedHashSet;
1751  import java.util.Set;
1752  import javax.swing.BorderFactory;
1753  import javax.swing.JButton;
1754  import javax.swing.JLabel;
1755  import javax.swing.JPanel;
1756  import javax.swing.JScrollPane;
1757  import javax.swing.JTabbedPane;
1758  import javax.swing.JTextArea;
1759  import javax.swing.SwingConstants;
1760  import com.test.FileControl;
1761  import com.test.bean.Activity;
1762  import com.test.parse1.ParseBpel;
1763  public class tempPanel extends JPanel implements ActionListener{
1764      public static HashMap<String,Integer> FalseBranch=new HashMap<String,Integer>();
1765      public static HashMap<String,Integer> TrueBranch=new HashMap<String,Integer>();
1766      public static HashMap<String,Integer> TruePath=new HashMap<String,Integer>;
1767      public static HashMap<String,Integer> FalsePath=new HashMap<String,Integer>();
```

```
1768        public static LinkedHashSet<String> list=new LinkedHashSet<String>();
1769        static String bpelPath1 ="D:\\Workspace1\\DebugTest\\bpel\\QuoteProcess.bpel";
1770        static String filename="d:/Workspace1/Quote/path.txt";
1771        JTextArea nodeNameTextArea;
1772        private Start startFrame1;
1773        public tempPanel(Start startFrame) throws Exception {
1774            ParseBpel.readwsdlfile(bpelPath1);
1775            this.startFrame1=startFrame;
1776            setLayout(null);
1777            JLabel tipLabel = new JLabel();
1778            tipLabel.setVerticalAlignment(SwingConstants.TOP);
1779            tipLabel.setText("<html><font color=Blue size='4'>"+"Step2:Reduce Suspicious Set
1780        <br>" +"<br>"+
1781                        "(1)\"nodes\" shows the suspisious nodes<br>"
1782                        +"(2)press \"PreBranch\" button to reduce the number of nodes<br>"+
1783                        "(3)press \"Atom Activity\" button to reduce the number of nodes <br>"
1784                        );
1785            tipLabel.setBounds(0, 10, 168, 336);
1786            add(tipLabel);
1787            JPanel panel = new JPanel();
1788            panel.setBounds(170, 10, 600, 420);
1789            add(panel);
1790            panel.setLayout(null);
1791            startFrame.debugmu.setEnabled(false);
1792            startFrame.diffPathmu.setEnabled(true);
1793            startFrame.SOBERmu.setEnabled(false);
1794            JTabbedPane tabbedPaneSec = new JTabbedPane(SwingConstants.TOP);
1795            tabbedPaneSec.setBounds(0, 10, 310, 280);
1796            panel.add(tabbedPaneSec);
1797            JPanel panelPath = new JPanel();
1798            tabbedPaneSec.addTab("FalseRoutes", null, panelPath, null);
1799            panelPath.setLayout(null);
1800            JScrollPane scrollPanePath = new JScrollPane();
1801            panelPath.add(scrollPanePath);
1802            scrollPanePath.setBounds(0, 0, 310, 250);
1803        scrollPanePath.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_A
1804        S_NEEDED);
1805        scrollPanePath.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_AS_NE
1806        EDED);
1807            JTextArea falsePathTextArea = new JTextArea();
1808            scrollPanePath.setViewportView(falsePathTextArea);
1809            FileControl.readFileByLines(filename,FalsePath,"false",falsePathTextArea);
1810            for(Object o: FalsePath.entrySet()){
1811                System.out.println(o);
1812            }
1813            JButton showFalesPathButton = new JButton("PreBranch");
1814            showFalesPathButton.setBounds(115, 305, 98, 23);
1815            panel.add(showFalesPathButton);
1816            JPanel panelSec_1 = new JPanel();
1817            tabbedPaneSec.addTab("TrueRoutes", null, panelSec_1, null);
```

```
1818            panelSec_1.setLayout(null);
1819            JScrollPane scrollPaneOut= new JScrollPane();
1820            scrollPaneOut.setBounds(0, 0, 310, 250);
1821        scrollPaneOut.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_AS
1822        _NEEDED);
1823        scrollPaneOut.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_AS_NEE
1824        DED);
1825            panelSec_1.add(scrollPaneOut);
1826            JTextArea TruePathTextArea= new JTextArea();
1827            FileControl.readFileByLines(filename,TruePath,"true",TruePathTextArea);
1828            scrollPaneOut.setViewportView(TruePathTextArea);
1829            JButton showTruePathButton = new JButton("Atom Activity");
1830            showTruePathButton.setBounds(225, 305, 108, 23);
1831            panel.add(showTruePathButton);
1832            JLabel lblNewLabel3 = new JLabel("Nodes");
1833            lblNewLabel3.setBounds(330, 10, 180, 25);
1834            panel.add(lblNewLabel3);
1835            lblNewLabel3.setFont(new Font("瀹嬩綋", Font.BOLD, 18));
1836            JScrollPane scrollPane_3 = new JScrollPane();
1837            scrollPane_3.setBounds(330, 35, 120, 250);
1838            panel.add(scrollPane_3);
1839            nodeNameTextArea = new JTextArea();
1840            nodeNameTextArea.setBorder(BorderFactory.createLoweredSoftBevelBorder());
1841            nodeNameTextArea.setBounds(330, 35, 120, 250);
1842            panel.add(nodeNameTextArea);
1843            nodeNameTextArea.setColumns(50);
1844            scrollPane_3.setViewportView(nodeNameTextArea);
1845        scrollPane_3.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_AS_
1846        NEEDED);
1847        scrollPane_3.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_AS_NEED
1848        ED);
1849            addnodes();
1850            JButton next=new JButton("next");
1851            next.setBounds(350, 330, 88, 23);
1852            panel.add(next);
1853            next.addActionListener(
1854                new ActionListener() {
1855                    public void actionPerformed(ActionEvent arg0) {
1856                        shownodes();
1857                        SOBERPanel soberpanel;
1858                        try {
1859                            soberpanel = new
1860        SOBERPanel(startFrame1,list,TruePath,FalsePath,ParseBpel.If_);
1861                            startFrame1.showPanel(soberpanel);
1862                        } catch (Exception e) {
1863                            e.printStackTrace();
1864                        }
1865                    }
1866            });
1867            showFalesPathButton.addActionListener(
```

```
1868                    new ActionListener() {
1869                        public void actionPerformed(ActionEvent arg0) {
1870                            Collection<String> set=FalsePath.keySet();
1871                            Collection<String> branch=ParseBpel.If_.keySet();
1872                    for(Object o:ParseBpel.If_.entrySet()){
1873                                System.out.println(o);
1874                            }
1875                            for(String s:set){
1876                                System.out.println(s);
1877                                for(String s2:branch){
1878                                    if(s.contains(s2)){
1879                                        System.out.println("    "+s2);
1880                                        if(FalseBranch.containsKey(ParseBpel.If_.get(s2))){
1881                                            System.out.println(FalsePath.get(s)+"put1"+"
1882                                            FalseBranch.put(ParseBpel.If_.get(s2),
1883                                        }else{
1884                                            System.out.println("put2"+"       null");
1885                                            System.out.println(FalsePath.get(s)+"
1886                                            FalseBranch.put(ParseBpel.If_.get(s2),
1887                                        }
1888                                    }
1889                                }
1890                            }
1891                            int fsize=0;
1892                            for(int i:FalsePath.values()){
1893                                fsize+=i;
1894                            }
1895                            System.out.println("FalsePath.size()="+fsize);
1896                            for(String key:FalseBranch.keySet()){
1897                                int value=FalseBranch.get(key);
1898                                System.out.println(value);
1899                                if(value<fsize){
1900                                    System.out.println("delete:       "+key);
1901                                    deletlist(key);
1902                                }
1903                            }
1904                            shownodes();
1905                        }
1906                    });
1907            showTruePathButton.addActionListener(
1908                new ActionListener() {
1909                    public void actionPerformed(ActionEvent arg0) {
1910                        if(TruePath.size()>0){
1911                            for(Activity a: ParseBpel.activity.components){
1912                                if(a.getClass().getName().equals("com.test.bean.Atom"))
1913                                    list.remove(a.getName());
1914                            }
1915                        }
1916                        shownodes();
1917                    }
```

```
                });
        }
        private void addnodes() {
            Set<String> set=FalsePath.keySet();
            for(String s:set){
                String[] temp=s.split("#");
                for(String l:temp)
                    list.add(l);
            }
            list.remove("false");
            for(String s:list){
                nodeNameTextArea.append(s+"\n");
            }
        }
        private void shownodes(){
            nodeNameTextArea.setText("");
            for(String s:list){
                nodeNameTextArea.append(s+"\n");
            }
        }
        private void deletlist(String key) {
            String delete=ParseBpel.If_Branch.get(key);
            String[] branch=delete.split("#");
            for(String s:branch)
                list.remove(s);
        }
        public void actionPerformed(ActionEvent e) {
        }
}
package com.test.view;
import java.awt.EventQueue;
import javax.swing.JFrame;
import javax.swing.JMenuBar;
import javax.swing.JMenu;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.awt.BorderLayout;
import javax.swing.JPanel;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.util.Stack;
public class Start extends JFrame{
        private JFrame framDebug;
        private Stack stack=new Stack();
        public JMenu diffPathmu=null;
        public JMenu SOBERmu=null;
        public JMenu debugmu=null;
        public JMenu preparemu =null;
        public Stack getStack() {
            return stack;
```

```
1968        }
1969        public void setStack(Stack stack) {
1970            this.stack = stack;
1971        }
1972        public static void main(String[] args) {
1973            EventQueue.invokeLater(new Runnable() {
1974                @Override
1975                public void run() {
1976                    try {
1977                        Start window = new Start();
1978                        window.framDebug.setVisible(true);
1979                    } catch (Exception e) {
1980                        e.printStackTrace();
1981                    }
1982                }
1983            });
1984        }
1985        public Start() {
1986            initialize();
1987        }
1988        private void initialize() {
1989            framDebug = new JFrame();
1990            framDebug.setTitle("WS-BPEL_CSLocator");
1991            framDebug.setSize(660,450);
1992            framDebug.setResizable(true);
1993            framDebug.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
1994            JMenuBar menuBar = new JMenuBar();
1995            framDebug.setJMenuBar(menuBar);
1996            JMenu mnNewMenu_1 = new JMenu("Preparation");
1997            mnNewMenu_1.addMouseListener(new MouseAdapter() {
1998                public void mouseClicked(MouseEvent arg0) {
1999                    JPanel StarterPanel=new Preparation(Start.this);
2000                    showPanel(StarterPanel);
2001                    stack.push(StarterPanel);
2002                }
2003            });
2004            menuBar.add(mnNewMenu_1);
2005            diffPathmu = new JMenu("SusSetReduce");
2006            menuBar.add(diffPathmu);
2007            diffPathmu.setEnabled(false);
2008            diffPathmu.addActionListener(new ActionListener() {
2009                public void actionPerformed(ActionEvent e) {
2010                    if(stack!=null){
2011                        for (Object x : stack) {
2012                            if(x instanceof PathDiffPanel){
2013                                showPanel((PathDiffPanel)x);
2014                                SOBERmu.setEnabled(true);
2015                            }
2016                        }
2017                    }
```

```
                }
            });
            SOBERmu = new JMenu("Predicate-Ranking");
            menuBar.add(SOBERmu);
            SOBERmu.setEnabled(false);
            SOBERmu.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent e) {
                    if(stack!=null){
                        for (Object x : stack) {
                            if(x instanceof SOBERPanel){
                                showPanel((SOBERPanel)x);
                            }
                        }
                    }
                }
            });
            debugmu = new JMenu("Debug");
            debugmu.addActionListener(new ActionListener() {
                @Override
                public void actionPerformed(ActionEvent e) {
                    if(stack!=null){
                        for (Object x : stack) {
                            if(x instanceof DebugPanel){
                                showPanel((DebugPanel)x);
                            }
                        }
                    }
                }
            });
            menuBar.add(debugmu);
            debugmu.setEnabled(false);
            JMenu mnAbout = new JMenu("About");
            menuBar.add(mnAbout);
            framDebug.getContentPane().setLayout(new BorderLayout(0, 0));
        }
        public void showPanel(JPanel panel) {
            framDebug.getContentPane().removeAll();
            framDebug.getContentPane().add(panel,BorderLayout.CENTER);
            framDebug.getContentPane().validate();
            framDebug.getContentPane().repaint();
        }
    }
    package com.test.view;
    import java.awt.Font;
    import java.awt.event.ActionEvent;
    import java.awt.event.ActionListener;
    import java.text.DecimalFormat;
    import java.util.Collection;
    import java.util.HashMap;
    import java.util.LinkedHashSet;
```

```
import java.util.Map;
import java.util.TreeMap;
import javax.swing.JButton;
import javax.swing.JFileChooser;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTable;
import javax.swing.JTextArea;
import javax.swing.JTextField;
import javax.swing.SwingConstants;
import javax.swing.table.DefaultTableCellRenderer;
public class SOBERPanel extends JPanel{
	private JTextField bpelText;
	private JTextField testField;
	private JFileChooser jfc;
	private JTextArea currentArea ;
	private Start startFrame1;
	private String bpelpath;
	private String exportfile="";
	private JTextArea falsePathTextField;
	private JTextField TruePathTextField;
	private JButton showFalesPathButton ;
	private JButton showTruePathButton ;
	private JTextArea falsePathTextArea;
	private JScrollPane scrollPane;
	private JTextArea TruePathTextArea;
	private JScrollPane scrollPane_3;
	private JTextArea nodeNameTextArea;
	private JTable table;
	public SOBERPanel(final Start startFrame,final LinkedHashSet<String>
	list,HashMap<String,Integer> TruePath,
			HashMap<String,Integer> FalsePath,HashMap<String,String> If_) throws
	Exception{
		this.startFrame1=startFrame;
		setLayout(null);
		startFrame.debugmu.setEnabled(true);
		startFrame.diffPathmu.setEnabled(true);
		startFrame.SOBERmu.setEnabled(true);
		JLabel tipLabel = new JLabel();
		tipLabel.setVerticalAlignment(SwingConstants.TOP);
		tipLabel.setText("<html><font color=Blue size='4'>"+"Sort the Predicates :<br>"
	+"<br>"+
				"(1)column\"predicate\" shows the suspicious predicates"+"<br>"
				+"(2)column\"suspicion\" shows the suspicion of the corresponding
	predicates"+"<br>"
				 );
		tipLabel.setBounds(0, 10, 155, 336);
		add(tipLabel);
		JLabel label = new JLabel("Sorted Predicates");
```

```
2118            label.setBounds(190, 35, 220, 25);
2119            add(label);
2120            label.setFont(new Font("瀹嬩綋", Font.BOLD, 18));
2121            String[] n={"predicate","suspiciousness"};
2122            String[][]  data  =  new  String[30][2];
2123            for(int  i=0;  i <data.length;  i++)
2124            {
2125                for(int  j=0;  j <data[i].length  ;  j++)
2126                    data[i][j]  = "";
2127            }
2128            table = new JTable(data,n);
2129            table.setAutoResizeMode(JTable.AUTO_RESIZE_ALL_COLUMNS);
2130            scrollPane = new JScrollPane(table);
2131            scrollPane.setBounds(190, 80, 350, 250);
2132            add(scrollPane);
2133            DefaultTableCellRenderer tcr = new DefaultTableCellRenderer();
2134            tcr.setHorizontalAlignment(JLabel.CENTER);
2135            table.setDefaultRenderer(Object.class, tcr);
2136        scrollPane.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_AS_N
2137        EEDED);
2138        scrollPane.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDE
2139        D);
2140            table.setFillsViewportHeight(true);
2141            final TreeMap<Double, String> map=SOBER(list,TruePath,FalsePath,If_);
2142            int i=0;
2143            System.out.println(0);
2144            for(Map.Entry entry : map.descendingMap().entrySet()){
2145                System.out.println(entry.getValue()+""+entry.getKey());
2146                table.setValueAt(entry.getValue(), i, 0);
2147                table.setValueAt(entry.getKey().toString(), i++, 1);
2148            }
2149            JButton next = new JButton("next");
2150            next.setBounds(543, 349, 88, 23);
2151            add(next);
2152            next.addActionListener(new ActionListener() {
2153                @Override
2154                public void actionPerformed(ActionEvent arg0) {
2155                    DebugPanel debugpanel;
2156                    try {
2157                        debugpanel = new DebugPanel(startFrame1,map,list);
2158                        startFrame1.showPanel(debugpanel);
2159                        if(startFrame1!=null){
2160                            startFrame1.getStack().add(SOBERPanel.this);
2161                            startFrame1.SOBERmu.setEnabled(true);
2162                            }
2163                    } catch (Exception e) {
2164                        e.printStackTrace();
2165                    }
2166                }
2167            });
```

```
2168        }
2169        public static TreeMap<Double, String> SOBER(LinkedHashSet<String>
2170        list,HashMap<String,Integer> TruePath,
2171                HashMap<String,Integer> FalsePath,HashMap<String,String> If_) throws
2172        Exception{
2173            TreeMap<Double, String> tmap=new TreeMap<Double,String>();
2174            Collection<String> set=If_.values();
2175            int x=0;
2176            int y=0;
2177            int n=0;
2178            int m=0;
2179            String temp=null;
2180            for(Integer i:FalsePath.values()){
2181                m+=i;
2182            }
2183            for(Integer i:TruePath.values()){
2184                n+=i;
2185            }
2186            System.out.println(m+" "+n);
2187            for(String s: set){
2188                if(list.contains(s)){
2189                    x=0;
2190                    y=0;
2191                    temp=null;
2192                    System.out.println("SOBER  "+s);
2193                    for(Map.Entry<String,Integer> map: FalsePath.entrySet()){
2194                        for(Map.Entry<String,String> map2: If_.entrySet()){
2195                            if(map2.getValue().equals(s))
2196                                temp=map2.getKey();
2197                        }
2198                        if(map.getKey().contains(temp)){
2199                            y+=map.getValue();
2200                        }
2201                    }
2202                    for(Map.Entry<String,Integer> map: TruePath.entrySet()){
2203                        for(Map.Entry<String,String> map2: If_.entrySet()){
2204                            if(map2.getValue().equals(s))
2205                                temp=map2.getKey();
2206                        }
2207                        if(map.getKey().contains(temp)){
2208                            x+=map.getValue();
2209                        }
2210                    }
2211                    System.out.println("X：  "+x);
2212                    System.out.println("Y:"+y);
2213                    if((x+y==m+n)||(x+y==0)){
2214                        double P=Double.POSITIVE_INFINITY;
2215                        if(P==Double.NEGATIVE_INFINITY)
2216                            P=-1000;
2217                        else if(P==Double.POSITIVE_INFINITY)
```

```
                        P=1000;
                    while(tmap.containsKey(P)){
                        P=P-0.001;
                    }
                    DecimalFormat    df    = new DecimalFormat("######0.00");
                    df.format(P);
                    tmap.put(Double.valueOf(df.format(P)), s);
                    continue;
                }
                else{
                    double ux=(double)x/(double)n;
                    double fangcha=ux*(1-ux);
                    fangcha=(x*(1-ux)*(1-ux)+(n-x)*ux*ux)/(n-1);
                    double uy=(double)y/m;
                    double z=(uy-ux)/(Math.sqrt(fangcha)/Math.sqrt(m));
                    double P=Math.log(ux/(Math.sqrt(m)*Math.pow(Math.E, -z*z/2)));
                    if(Double.isNaN(P)){
                        P=0.5;
                        while(tmap.containsKey(P)){
                            P=P-0.001;
                        }
                        DecimalFormat    df    = new DecimalFormat("######0.00");
                        df.format(P);
                        tmap.put(Double.valueOf(df.format(P)), s);
                    }else{
                        if(P==Double.NEGATIVE_INFINITY){
                            P=-1000;
                        }
                        else if(P==Double.POSITIVE_INFINITY){
                            P=1000;
                        }
                        while(tmap.containsKey(P)){
                            P=P-0.001;
                            System.out.println("same"+P+"\n");
                        }
                        DecimalFormat    df    = new DecimalFormat("######0.00");
                        df.format(P);
                        tmap.put(Double.valueOf(df.format(P)), s);
                    }
                }
            }
        }
        System.out.println("result:"+"\n");
        Collection<String> c=tmap.descendingMap().values();
        for(String  s: c){
            System.out.println(s);
        }
        Object[] array=list.toArray();
        for(int i=array.length-1;i>=0;i--){
            String s=(String)array[i];
```

```
                    if(!c.contains(s)){
                        System.out.println(s);
                    }
                }
                return tmap;
            }
            public void actionPerformed(ActionEvent e) {
            }
    }
    package com.test.view;
    import javax.swing.JFileChooser;
    import javax.swing.JPanel;
    import javax.swing.JTextField;
    import javax.swing.JButton;
    import javax.swing.JLabel;
    import java.awt.BorderLayout;
    import java.awt.Font;
    import javax.swing.JScrollPane;
    import javax.swing.JTextArea;
    import org.apache.axiom.om.OMElement;
    import com.test.bpelbean.TestcaseNode;
    import com.test.update.*;
    import com.test.EngineImpl;
    import com.test.FileControl;
    import com.test.OmelementParse;
    import com.test.XMLHelper_Ran;
    import java.awt.event.ActionListener;
    import java.awt.event.ActionEvent;
    import java.io.File;
    import java.util.ArrayList;
    import java.util.List;
    import javax.swing.JTabbedPane;
    import javax.swing.SwingConstants;
    public class Preparation extends JPanel implements ActionListener{
        static String bpelPath1 ="D:\\Workspace1\\DebugTest\\bpel\\QuoteProcess.bpel";
        private JTextField bpelText;
        private JTextField testField;
        private JFileChooser jfc;
        private JTextArea expectArea;
        private Start startFrame1;
        private JTextArea testArea;
        private String bpelpath;
        private String exportfile="";
        private JTextField expecttextField;
        private JButton OpenBpelButton ;
        private JButton parseButton;
        private JButton expectButton ;
        private JButton TestCaseButton ;
        public Preparation(Start startFrame) {
            this.startFrame1=startFrame;
```

```
2318            setLayout(null);
2319            jfc=new JFileChooser(".");
2320            jfc.setCurrentDirectory(new File("d:/Workspace1/Quote/bpelContent"));
2321            final JPanel panel = new JPanel();
2322            panel.setBounds(167, 10, 475, 336);
2323            add(panel);
2324            panel.setLayout(null);
2325            startFrame.debugmu.setEnabled(false);
2326            startFrame.diffPathmu.setEnabled(false);
2327            startFrame.SOBERmu.setEnabled(false);
2328            expectButton = new JButton("Expect Button");
2329            expectButton.setBounds(365, 70, 98, 23);
2330            panel.add(expectButton);
2331            parseButton = new JButton("Parse ");
2332            parseButton.setBounds(365, 103, 98, 23);
2333            panel.add(parseButton);
2334            TestCaseButton = new JButton("Choose Testcases");
2335            TestCaseButton.setBounds(365, 37, 98, 23);
2336            panel.add(TestCaseButton);
2337            testField = new JTextField();
2338            testField.setBounds(10, 38, 345, 21);
2339            panel.add(testField);
2340            testField.setColumns(10);
2341            JTabbedPane tabbedPane = new JTabbedPane(SwingConstants.TOP);
2342            tabbedPane.setBounds(10, 136, 454, 194);
2343            panel.add(tabbedPane);
2344            JPanel panel_2 = new JPanel();
2345            tabbedPane.addTab("Testcases", null, panel_2, null);
2346            panel_2.setLayout(null);
2347            JScrollPane scrollPane = new JScrollPane();
2348            scrollPane.setBounds(0, 0, 546, 169);
2349            panel_2.add(scrollPane);
2350            testArea = new JTextArea();
2351            scrollPane.setViewportView(testArea);
2352            JPanel panel_1 = new JPanel();
2353            tabbedPane.addTab("Expect Output", null, panel_1, null);
2354            panel_1.setLayout(null);
2355            JScrollPane scrollPane_1 = new JScrollPane();
2356            scrollPane_1.setBounds(0, 0, 546, 169);
2357            panel_1.add(scrollPane_1);
2358            expectArea = new JTextArea();
2359            scrollPane_1.setViewportView(expectArea);
2360            expecttextField = new JTextField();
2361            expecttextField.setBounds(10, 70, 345, 23);
2362            panel.add(expecttextField);
2363            expecttextField.setColumns(10);
2364            OpenBpelButton = new JButton("Open WS-BPEL");
2365            OpenBpelButton.setBounds(365, 9, 98, 23);
2366            panel.add(OpenBpelButton);
2367            bpelText = new JTextField();
```

```
2368            bpelText.setBounds(10, 10, 345, 21);
2369            panel.add(bpelText);
2370            bpelText.setColumns(10);
2371            OpenBpelButton.addActionListener(new ActionListener() {
2372                @Override
2373                public void actionPerformed(ActionEvent arg0) {
2374                    jfc.setFileSelectionMode(0);
2375                    int state=jfc.showOpenDialog(null);
2376                    if(state==1){
2377                        return;
2378                    }
2379                    else{
2380                        File f=jfc.getSelectedFile();
2381                        System.out.println(f.getName());
2382                        bpelText.setText(f.getAbsolutePath());
2383                        bpelpath=bpelText.getText();
2384                            try {
2385                                FileControl.copyfile("E://"+f.getName(),bpelpath);
2386                            } catch (Exception e1) {
2387                                e1.printStackTrace();
2388                            }
2389
2390
2391                    }
2392                }
2393            });
2394            TestCaseButton.addActionListener(new ActionListener() {
2395                public void actionPerformed(ActionEvent e) {
2396                     jfc.setFileSelectionMode(0);
2397                        int state=jfc.showOpenDialog(null);
2398                        if(state==1){
2399                            return;
2400                        }
2401                        else{
2402                            File f=jfc.getSelectedFile();
2403                            testField.setText(f.getAbsolutePath());
2404                            FileControl.readFileByLines(f.getAbsolutePath(),testArea);
2405                        }
2406                }
2407            });
2408            expectButton.addActionListener(new ActionListener() {
2409                @Override
2410                public void actionPerformed(ActionEvent e) {
2411                     jfc.setFileSelectionMode(0);
2412                        int state=jfc.showOpenDialog(null);
2413                        if(state==1){
2414                            return;
2415                        }
2416                        else{
2417                            File f=jfc.getSelectedFile();
```

```
                                expecttextField.setText(f.getAbsolutePath());
                                FileControl.readFileByLines(f.getAbsolutePath(),expectArea);
                            }
                        }
                    });
                    final JButton runButton = new JButton("Run");
                    runButton.setBounds(543, 349, 88, 23);
                    add(runButton);
                    JLabel tipLabel = new JLabel();
                    tipLabel.setVerticalAlignment(SwingConstants.TOP);
                    tipLabel.setText("<html><font color=Blue size='4'>"+"Step1:Debug Preparation<br>"
                            "(1)choose the WS-BPEL file which you want to test<br>"
                            +"(2)choose the Testcases you want to test<br>"+
                            "(3)choose the Expect Output file you want to compare<br>"+
                            "(4)deploy the WS-BPEl and send the Testcases Message<br>" );
                    tipLabel.setBounds(0, 10, 168, 336);
                    add(tipLabel);
                    runButton.addActionListener(new ActionListener() {
                        @Override
                        public void actionPerformed(ActionEvent arg0) {
                            boolean flag = true;
                            String cur=testArea.getText();
                            String exp=expectArea.getText();
                            OutputPanel diffpanel=new OutputPanel(startFrame1);
                            startFrame1.showPanel(diffpanel);
                            if(startFrame1!=null){
                                startFrame1.getStack().add(Preparation.this);
                                startFrame1.diffPathmu.setEnabled(true);
                                }
                        }
                    });
                    parseButton.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent arg0) {
                            BPELFrame    bpelFrame = new BPELFrame(bpelPath1);
                            bpelFrame.setVisible(true);
                            bpelFrame.pack();
                        }
                    });
                }
                @Override
                public void actionPerformed(ActionEvent e) {
                }
            }
            package com.test.view;
            import java.awt.Font;
            import java.awt.event.ActionEvent;
            import java.awt.event.ActionListener;
            import java.io.BufferedReader;
            import java.io.File;
            import java.io.FileNotFoundException;
```

```
2468  import java.io.FileReader;
2469  import java.io.IOException;
2470  import java.util.Collection;
2471  import java.util.HashMap;
2472  import java.util.LinkedHashSet;
2473  import java.util.Set;
2474  import javax.swing.BorderFactory;
2475  import javax.swing.JButton;
2476  import javax.swing.JFileChooser;
2477  import javax.swing.JLabel;
2478  import javax.swing.JPanel;
2479  import javax.swing.JScrollPane;
2480  import javax.swing.JTextArea;
2481  import javax.swing.JTextField;
2482  import com.test.FileControl;
2483  import com.test.bean.Activity;
2484  import com.test.parse1.ParseBpel;
2485  public class PathDiffPanel extends JPanel implements ActionListener{
2486      public static HashMap<String,Integer> FalseBranch=new HashMap<String,Integer>();/
2487      public static HashMap<String,Integer> TrueBranch=new HashMap<String,Integer>();
2488      public static HashMap<String,Integer> TruePath=new HashMap<String,Integer>();
2489      public static HashMap<String,Integer> FalsePath=new HashMap<String,Integer>();
2490      public static LinkedHashSet<String> list=new LinkedHashSet<String>();
2491      static String bpelPath1 ="D:\\Workspace1\\DebugTest\\bpel\\QuoteProcess.bpel";
2492      static String filename="d:/Workspace1/Quote/path.txt";
2493      private Start startFrame1;
2494      private JButton showFalesPathButton ;
2495      private JButton showTruePathButton ;
2496      private JTextArea falsePathTextArea;
2497      private JScrollPane scrollPane_1;
2498      private JScrollPane scrollPane_2;
2499      private JTextArea TruePathTextArea;
2500      private JScrollPane scrollPane_3;
2501      private JTextArea nodeNameTextArea;
2502      public PathDiffPanel(Start startFrame) throws Exception{
2503          ParseBpel.readwsdlfile(bpelPath1);
2504          this.startFrame1=startFrame;
2505          setLayout(null);
2506          final JPanel panel = new JPanel();
2507          panel.setBounds(20, 20, 600, 420);
2508          add(panel);
2509          panel.setLayout(null);
2510          startFrame.debugmu.setEnabled(false);
2511          startFrame.diffPathmu.setEnabled(true);
2512          startFrame.SOBERmu.setEnabled(false);
2513          JLabel lblNewLabel = new JLabel("False-Path");
2514          lblNewLabel.setBounds(10, 10, 345, 15);
2515          panel.add(lblNewLabel);
2516          lblNewLabel.setFont(new Font("瀹嬩綋", Font.BOLD, 18));
2517
```

```
2518            scrollPane_1 = new JScrollPane();
2519            scrollPane_1.setBounds(10, 35, 345, 100);
2520            panel.add(scrollPane_1);
2521            falsePathTextArea = new JTextArea();
2522            falsePathTextArea.setBorder(BorderFactory.createLoweredSoftBevelBorder());
2523            falsePathTextArea.setBounds(10, 35, 145, 100);
2524            panel.add(falsePathTextArea);
2525            falsePathTextArea.setColumns(50);
2526            scrollPane_1.setViewportView(falsePathTextArea);
2527            scrollPane_1.setHorizontalScrollBarPolicy(JScrollPane. DED);
2528            scrollPane_1.setVerticalScrollBarPolicy(JScrollPane. AS_NEEDED);
2529            FileControl.readFileByLines(filename,FalsePath,"false",falsePathTextArea);
2530            showFalesPathButton = new JButton("False-Diff");
2531            showFalesPathButton.setBounds(235, 140, 98, 23);
2532            panel.add(showFalesPathButton);
2533            JLabel lblNewLabel2 = new JLabel("True-Path");
2534            lblNewLabel2.setBounds(10, 180, 345, 15);
2535            panel.add(lblNewLabel2);
2536            lblNewLabel2.setFont(new Font("瀹嬩綋", Font.BOLD, 18));
2537            scrollPane_2 = new JScrollPane();
2538            scrollPane_2.setBounds(10, 200, 345, 100);
2539            panel.add(scrollPane_2);
2540            TruePathTextArea = new JTextArea();
2541            TruePathTextArea.setBorder(BorderFactory.createLoweredSoftBevelBorder());
2542            TruePathTextArea.setBounds(10, 200, 145, 100);
2543            panel.add(TruePathTextArea);
2544            TruePathTextArea.setColumns(50);
2545            scrollPane_2.setViewportView(TruePathTextArea);
2546            scrollPane_2.setHorizontalScrollBarPolicy(JScrollPane. _AS_NEEDED);
2547            scrollPane_2.setVerticalScrollBarPolicy(JScrollPane. _AS_NEEDED);
2548            FileControl.readFileByLines(filename,TruePath,"true",TruePathTextArea);
2549            showTruePathButton = new JButton("True-Diff");
2550            showTruePathButton.setBounds(235, 305, 98, 23);
2551            panel.add(showTruePathButton);
2552            JLabel lblNewLabel3 = new JLabel("Nodes");
2553            lblNewLabel3.setBounds(400, 10, 345, 15);
2554            panel.add(lblNewLabel3);
2555            lblNewLabel3.setFont(new Font("瀹嬩綋", Font.BOLD, 18));
2556            scrollPane_3 = new JScrollPane();
2557            scrollPane_3.setBounds(400, 35, 180, 260);
2558            panel.add(scrollPane_3);
2559            nodeNameTextArea = new JTextArea();
2560            nodeNameTextArea.setBorder(BorderFactory.createLoweredSoftBevelBorder());
2561            nodeNameTextArea.setBounds(400, 35, 180, 260);
2562            panel.add(nodeNameTextArea);
2563            nodeNameTextArea.setColumns(50);
2564            scrollPane_3.setViewportView(nodeNameTextArea);
2565            scrollPane_3.setHorizontalScrollBarPolicy(JScrollPane. SCROLLBAR_NEVER);
2566            scrollPane_3.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_ S_NEEDED);
2567            addnodes();
```

```
2568            JButton next=new JButton("next");
2569            next.setBounds(500, 330, 88, 23);
2570            panel.add(next);
2571            next.addActionListener(
2572                  new ActionListener() {
2573                        public void actionPerformed(ActionEvent arg0) {
2574                              shownodes();
2575                              SOBERPanel soberpanel;
2576                              try {
2577                                    soberpanel = new
2578        SOBERPanel(startFrame1,list,TruePath,FalsePath,ParseBpel.If_);
2579                                    startFrame1.showPanel(soberpanel);
2580                              } catch (Exception e) {
2581                                    e.printStackTrace();
2582                              }
2583                        }
2584            });
2585            showFalesPathButton.addActionListener(
2586                        new ActionListener() {
2587                              public void actionPerformed(ActionEvent arg0) {
2588                                    Collection<String> set=FalsePath.keySet();
2589                                    Collection<String> branch=ParseBpel.If_.keySet();
2590                                    for(Object o:ParseBpel.If_.entrySet()){
2591                                          System.out.println(o);
2592                                    }
2593                                    for(String s:set){
2594                                          System.out.println(s);
2595                                          for(String s2:branch){
2596                                                if(s.contains(s2)){
2597                                                      System.out.println("    "+s2);
2598                                                      if(FalseBranch.containsKey(ParseBpel.If_.get(s2))){
2599                                                            System.out.println(FalsePath.get(s)+"put1"+"
2600                                                            FalseBranch.put(ParseBpel.If_.get(s2),
2601                                                      }else{
2602                                                            System.out.println("put2"+"       null");
2603                                                            System.out.println(FalsePath.get(s));
2604                                                            FalseBranch.put(ParseBpel.If_.get(s2));
2605                                                      }
2606                                                }
2607                                          }
2608                                    }
2609                                    int fsize=0;
2610                                    for(int i:FalsePath.values()){
2611                                          fsize+=i;
2612                                    }
2613                                    System.out.println("FalsePath.size()="+fsize);
2614                                    for(String key:FalseBranch.keySet()){
2615                                          int value=FalseBranch.get(key);
2616                                          System.out.println(value);
2617                                          if(value<fsize){
```

```
                                        System.out.println("delete:       "+key);
                                        deletlist(key);
                                    }
                                }
                                shownodes();
                            }
                        });
            showTruePathButton.addActionListener(
                    new ActionListener() {
                        @Override
                        public void actionPerformed(ActionEvent arg0) {
                            if(TruePath.size()>0){
                                for(Activity a: ParseBpel.activity.components){
                                    if(a.getClass().getName().equals("com.test.bean.Atom"))
                                        list.remove(a.getName());
                                }
                            }
                            shownodes();
                        }
                    });
        }
        private void addnodes() {
            Set<String> set=FalsePath.keySet();
            for(String s:set){
                String[] temp=s.split("#");
                for(String l:temp)
                    list.add(l);
            }
            list.remove("false");
            for(String s:list){
                nodeNameTextArea.append(s+"\n");
            }
        }
        private void shownodes(){
            nodeNameTextArea.setText("");
            for(String s:list){
                nodeNameTextArea.append(s+"\n");
            }
        }
        private void deletlist(String key) {
            String delete=ParseBpel.If_Branch.get(key);
            String[] branch=delete.split("#");
            for(String s:branch)
                list.remove(s);
        }
        public void actionPerformed(ActionEvent e) {
        }
    }
    package com.test.view;
    import java.awt.Button;
```

```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.File;
import javax.swing.JButton;
import javax.swing.JFileChooser;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTabbedPane;
import javax.swing.JTextArea;
import javax.swing.JTextField;
import javax.swing.SwingConstants;
import com.test.FileControl;
public class OutputPanel extends JPanel implements ActionListener{
    private JTextField bpelText;
    private JTextField testField;
    private JFileChooser jfc;
    private JTextArea TCArea ;
    private Start startFrame1;
    private String bpelpath;
    private String exportfile="";
    private JTextField expecttextField;
    private JButton OpenBpelButton ;
    private JButton deployButton;
    private JButton expectButton ;
    private JButton TestCaseButton ;
    public OutputPanel(Start startFrame) {
        File testcase=new File("d:/Workspace1/Quote/testcase.txt");
        File expectout= new File("d:/Workspace1/Quote/ExpectOutput_Quote.txt");
        File errorout =new File("d:/Workspace1/Quote/error.txt");
        File path=new File("d:/Workspace1/Quote/m11.txt");
        this.startFrame1=startFrame;
        setLayout(null);
        JPanel panel = new JPanel();
        panel.setBounds(20, 20, 600, 420);
        add(panel);
        panel.setLayout(null);
        startFrame.debugmu.setEnabled(false);
        startFrame.diffPathmu.setEnabled(false);
        startFrame.SOBERmu.setEnabled(false);
        JTabbedPane tabbedPane = new JTabbedPane(SwingConstants.TOP);
        tabbedPane.setBounds(15, 15, 200, 300);
        panel.add(tabbedPane);
        JPanel panel_2 = new JPanel();
        tabbedPane.addTab("Testcases", null, panel_2, null);
        panel_2.setLayout(null);
        JScrollPane scrollPaneTC = new JScrollPane();
        scrollPaneTC.setBounds(0, 0, 200, 270);
        panel_2.add(scrollPaneTC);
        TCArea = new JTextArea();
        scrollPaneTC.setViewportView(TCArea);
```

```
2718            FileControl.readFileByLines(testcase.getAbsolutePath(),TCArea);
2719            JPanel panel_1 = new JPanel();
2720            tabbedPane.addTab("Expect Output", null, panel_1, null);
2721            panel_1.setLayout(null);
2722            JScrollPane scrollPaneEO = new JScrollPane();
2723            scrollPaneEO.setBounds(0, 0, 200, 270);
2724            panel_1.add(scrollPaneEO);
2725            JTextArea expectArea = new JTextArea();
2726            scrollPaneEO.setViewportView(expectArea);
2727            FileControl.readFileByLines(expectout.getAbsolutePath(), expectArea);
2728            JTabbedPane tabbedPaneSec = new JTabbedPane(SwingConstants.TOP);
2729            tabbedPaneSec.setBounds(270, 15, 310, 300);
2730            panel.add(tabbedPaneSec);
2731            JPanel panelPath = new JPanel();
2732            tabbedPaneSec.addTab("Execution Routes", null, panelPath, null);
2733            panelPath.setLayout(null);
2734            JScrollPane scrollPanePath = new JScrollPane(JScrollPane);
2735            panelPath.add(scrollPanePath);
2736            scrollPanePath.setBounds(0, 0, 310, 270);
2737            JTextArea PathArea = new JTextArea();
2738            scrollPanePath.setViewportView(PathArea);
2739            FileControl.readFileByLines(path.getAbsolutePath(), PathArea);
2740            JPanel panelSec_1 = new JPanel();
2741            tabbedPaneSec.addTab("Actual Output", null, panelSec_1, null);
2742            panelSec_1.setLayout(null);
2743            JScrollPane scrollPaneOut= new JScrollPane();
2744            scrollPaneOut.setBounds(0, 0, 310, 270);
2745            panelSec_1.add(scrollPaneOut);
2746            JTextArea OutArea= new JTextArea();
2747            scrollPaneOut.setViewportView(OutArea);
2748            FileControl.readFileByLines(errorout.getAbsolutePath(), OutArea);
2749            JButton next=new JButton("next");
2750            next.setBounds(490, 330, 88, 23);
2751            panel.add(next);
2752            next.addActionListener(
2753                    new ActionListener() {
2754                        @Override
2755                        public void actionPerformed(ActionEvent arg0) {
2756                            PathDiffPanel diffpanel;
2757                            try {
2758                                tempPanel temp =new tempPanel(startFrame1);
2759                                startFrame1.showPanel(temp);
2760                            } catch (Exception e) {
2761                                e.printStackTrace();
2762                            }
2763                        }
2764                });
2765        }
2766        public void actionPerformed(ActionEvent e) {
2767        }
```

```
2768    }
2769    package com.test.view;
2770    import java.io.File;
2771    import java.util.Calendar;
2772    import org.apache.axiom.om.OMAbstractFactory;
2773    import org.apache.axiom.om.OMElement;
2774    import org.apache.axiom.om.OMFactory;
2775    import org.apache.axiom.om.OMNamespace;
2776    import org.apache.axis2.AxisFault;
2777    import org.apache.axis2.addressing.EndpointReference;
2778    import org.apache.axis2.client.Options;
2779    import org.apache.axis2.client.ServiceClient;
2780    import org.dom4j.Attribute;
2781    import org.dom4j.Document;
2782    import org.dom4j.Element;
2783    import com.test.XMLHelper_Ran;
2784    public class MessageClient
2785    {
2786        public synchronized OMElement sendmessage(String name, int amount,String bpelpath)
2787        throws Exception
2788        {
2789            OMElement res=null;
2790              ServiceClient sc=null;
2791              try {
2792                    sc = new ServiceClient();
2793                    Options opts = new Options();
2794                    File file=new File(bpelpath);
2795                    String bpelname=file.getName();
2796                    String[] strs=bpelname.split("\\.");
2797                    opts.setTo(new EndpointReference(
2798                              "http://localhost:8080/ode/processes/"+strs[0]));
2799                    String actionName=processAction(bpelpath);
2800                    opts.setAction(actionName+"/process");
2801                    sc.setOptions(opts);
2802                    long startTime = Calendar.getInstance().getTimeInMillis();
2803                    System.out.println(actionName);
2804                    System.out.println(strs[0]);
2805                    res = sc.sendReceive(createPayLoad(name,amount,strs[0],actionName));
2806                    long endTime = Calendar.getInstance().getTimeInMillis();
2807                    System.out.println(name+"&&"+amount);
2808                   System.out.println("响应的信息： "+res);
2809                   System.out.println("引擎的执行时间(ms):"+(endTime-startTime));
2810                   sc.cleanupTransport();
2811              } catch (AxisFault e) {
2812              } finally{
2813              }
2814            return res;
2815        }
2816          private String processAction(String bpelpath) {
2817                Document doc = XMLHelper_Ran.openXMLFile(bpelpath);
```

```
2818            Element root = doc.getRootElement();
2819            Attribute attr=root.attribute("targetNamespace");
2820            return attr.getText();
2821        }
2822        public static OMElement createPayLoad(String parameter1,int parameter2,String
2823        bpelName,String namespace){
2824                    OMFactory fac = OMAbstractFactory.getOMFactory();
2825                   OMNamespace omNs = fac.createOMNamespace(namespace,"" );
2826                    OMNamespace omNs1 = fac.createOMNamespace(namespace,"");
2827                    OMElement method = fac.createOMElement(bpelName+"Request",omNs);
2828                    OMElement value1 = fac.createOMElement("name",omNs1);
2829                    OMElement value2 = fac.createOMElement("amount",omNs1);
2830                    value1.setText(parameter1);
2831                    value2.setText(parameter2+"");
2832                    method.addChild(value1) ;
2833                    method.addChild(value2) ;
2834                   System.out.println("发送的消息:"+method);
2835                    return method;
2836               } ;
2837           public static void main(String args[]) throws Exception{
2838                    MessageClient client=new MessageClient();
2839                  String string=
2840        client.processAction("D:\\Workspace1\\DebugTest\\bpel\\QuoteProcess.bpel");
2841                  System.out.println(string);
2842                 client.sendmessage("candy",
2843        200,"D:\\Workspace1\\DebugTest\\bpel\\QuoteProcess.bpel" );
2844              }
2845    }
2846    package com.test.view;
2847    import java.awt.Color;
2848    import java.awt.Font;
2849    import java.awt.event.ActionEvent;
2850    import java.util.Collection;
2851    import java.util.HashMap;
2852    import java.util.LinkedHashSet;
2853    import java.util.Map;
2854    import java.util.TreeMap;
2855    import javax.swing.BorderFactory;
2856    import javax.swing.JButton;
2857    import javax.swing.JFileChooser;
2858    import javax.swing.JLabel;
2859    import javax.swing.JPanel;
2860    import javax.swing.JScrollPane;
2861    import javax.swing.JTabbedPane;
2862    import javax.swing.JTable;
2863    import javax.swing.JTextArea;
2864    import javax.swing.JTextField;
2865    import javax.swing.SwingConstants;
2866    import javax.swing.border.LineBorder;
2867    import javax.swing.table.DefaultTableCellRenderer;
```

```
import javax.swing.table.DefaultTableModel;
import javax.swing.table.TableColumnModel;
import com.test.FileControl;
import com.test.XMLHelper_Ran;
import com.test.parse1.ParseBpel;
public class DebugPanel extends JPanel{
    private JTextField bpelText;
    private JTextField testField;
    private JFileChooser jfc;
    private JTextArea currentArea ;
    private Start startFrame1;
    private String bpelpath;
    private String exportfile="";
    private JTextArea falsePathTextField;
    private JTextField TruePathTextField;
    private JButton showFalesPathButton ;
    private JButton showTruePathButton ;
    private JTextArea falsePathTextArea;
    private JScrollPane scrollPane;
    private JTextArea TruePathTextArea;
    private JScrollPane scrollPane_3;
    private JTextArea nodeNameTextArea;
    private JTable table;
    public DebugPanel(final Start startFrame,TreeMap<Double, String>
    map,LinkedHashSet<String> list) throws Exception{
        this.startFrame1=startFrame;
        setLayout(null);
        startFrame.debugmu.setEnabled(true);
        startFrame.diffPathmu.setEnabled(true);
        startFrame.SOBERmu.setEnabled(true);
        JLabel tipLabel = new JLabel();
        tipLabel.setVerticalAlignment(SwingConstants.TOP);
        tipLabel.setText("<html><font color=Blue size='4'>"+"Result:<br>" +"<br>"+
                "(1)column\"Order\" shows the node's suspicious order"+"<br>"
                +"(2)column\"Node\" shows the node of the corresponding order"+"<br>"
                  );
        tipLabel.setBounds(0, 10, 155, 336);
        add(tipLabel);
        JLabel label = new JLabel("Result");
        label.setBounds(190, 35, 80, 15);
        add(label);
        label.setFont(new Font("宋体", Font.BOLD, 18));
        String[] n={"Order","Node"};
        String[][] data = new String[30][2];
        for(int i=0; i <data.length; i++)
        {
            for(int j=0; j <data[i].length ; j++)
                data[i][j] = "";
        }
        table = new JTable(data,n);
```

```
2918                table.setAutoResizeMode(JTable.AUTO_RESIZE_ALL_COLUMNS);
2919                scrollPane = new JScrollPane(table);
2920                scrollPane.setBounds(190, 80, 350, 250);
2921                add(scrollPane);
2922                DefaultTableCellRenderer tcr = new DefaultTableCellRenderer();
2923                tcr.setHorizontalAlignment(JLabel.CENTER);
2924                table.setDefaultRenderer(Object.class, tcr);
2925                scrollPane.setHorizontalScrollBarPolicy(JScrollPane. EEDED);
2926                scrollPane.setVerticalScrollBarPolicy(JScrollPane);
2927                table.setFillsViewportHeight(true);
2928                int i=0;
2929                for(Map.Entry entry : map.descendingMap().entrySet()){
2930                    System.out.println(entry.getValue()+""+entry.getKey());
2931                    table.setValueAt((i+1)+"", i, 0);
2932                    table.setValueAt(entry.getValue(), i++, 1);
2933                }
2934                Collection<String> c=map.descendingMap().values();
2935                Object[] array=list.toArray();
2936                for(int j=array.length-1;j>=0;j--){
2937                    String s=(String)array[j];
2938                    if(!c.contains(s)){
2939                        table.setValueAt((i+1)+"", i, 0);
2940                        table.setValueAt(s, i++, 1);
2941                    }
2942                }
2943            }
2944            public void actionPerformed(ActionEvent e) {
2945            }
2946    }
2947    package com.test.view;
2948    import java.awt.Dimension;
2949    import java.util.Iterator;
2950    import com.test.StringMapper;
2951    import javax.swing.JFrame;
2952    import javax.swing.JScrollPane;
2953    import javax.swing.JTree;
2954    import javax.swing.event.TreeSelectionEvent;
2955    import javax.swing.event.TreeSelectionListener;
2956    import javax.swing.tree.DefaultMutableTreeNode;
2957    import org.dom4j.Document;
2958    import org.dom4j.Element;
2959    import com.test.ElementWraper;
2960    import com.test.XMLHelper_Ran;
2961    public class BPELFrame extends JFrame {
2962        private static final long serialVersionUID = 1L;
2963        private JTree tree;
2964        private int j=1;
2965        private Document bpelDocument;
2966        public String s="";
2967        public BPELFrame( String bpelFilePath) {
```

```
2968            super();
2969            this.setTitle(StringMapper.get("BPELTitle"));
2970            bpelDocument = XMLHelper_Ran.openXMLFile(bpelFilePath);
2971            initControl();
2972            initLayout();
2973            this.setResizable(false);
2974        }
2975        private void initControl() {
2976            Element root=bpelDocument.getRootElement();
2977            tree = new JTree(build(bpelDocument.getRootElement(), ""));
2978            tree.addTreeSelectionListener(new TreeClickAction());
2979        }
2980        private void initLayout() {
2981            JScrollPane scrollPane = new JScrollPane(tree,
2982                    JScrollPane.VERTICAL_SCROLLBAR_ALWAYS,
2983                    JScrollPane.HORIZONTAL_SCROLLBAR_ALWAYS);
2984            scrollPane.setPreferredSize(new Dimension(800, 600));
2985            this.add(scrollPane);
2986        }
2987        private DefaultMutableTreeNode build(Element element, String parentXPath) {
2988            ElementWraper ew = new ElementWraper();
2989            ew.setElement(element);
2990            String xpath="";
2991            String xPath = "/" + element.getNamespacePrefix() + ":" + element.getName();
2992            if (element.attributeValue("name") != null) {
2993                xPath = xPath + "[@name='" + element.attributeValue("name")
2994                        + "']";
2995            }
2996        public String getS() {
2997            return s;
2998        }
2999        public String getS() {
3000            return s;
3001        }
3002        public void setS(String s) {
3003            this.s = s;
3004        }
3005        private class TreeClickAction implements TreeSelectionListener {
3006            public void valueChanged(TreeSelectionEvent e) {
3007                Object o = ((DefaultMutableTreeNode) e.getPath()
3008                        .getLastPathComponent()).getUserObject();
3009                if (o instanceof ElementWraper) {
3010                    ElementWraper ew = (ElementWraper) o;
3011                    Element element = (Element) ew.getElement();
3012                }
3013            }
3014        }
3015    }
3016    package com.test.update;
3017    import java.sql.Connection;
```

```
import java.sql.PreparedStatement;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.List;
import com.test.bpelbean.WareHouse;
import com.test.DataUtil;
public class WarehouseUpdate {
    private String date1 = "2014-06-09";
    private String date2 = "2014-10-09";
    private String date3 = "2014-05-01";
    private String date4 = "2016-08-15";
    private String sql1 = "update warehouse set Name=?,Amount=?,ProductionDate=? where
    id=1";
    private String sql2 = "update warehouse set Name=?,Amount=?,ProductionDate=? where
    id=2";
    private String sql3 = "update warehouse set Name=?,Amount=?,ProductionDate=? where
    id=3";
    private String sql4 = "update warehouse set Name=?,Amount=?,ProductionDate=? where
    id=4";
    public void warehouseupdate() throws Exception {
        int res = 0;
        SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd");
        java.util.Date date11 = format.parse(date1);
        java.util.Date date12 = format.parse(date2);
        java.util.Date date13 = format.parse(date3);
        java.util.Date date14 = format.parse(date4);
        java.sql.Date time1 = new java.sql.Date(date11.getTime());
        java.sql.Date time2 = new java.sql.Date(date12.getTime());
        java.sql.Date time3 = new java.sql.Date(date13.getTime());
        java.sql.Date time4 = new java.sql.Date(date14.getTime());
        WareHouse warehouse1 = new WareHouse("cookie", 400, time1);
        WareHouse warehouse2 = new WareHouse("milk", 500, time2);
        WareHouse warehouse3 = new WareHouse("coca-cola", 100, time3);
        WareHouse warehouse4 = new WareHouse("candy", 500, time4);
        List<WareHouse> list = new ArrayList<WareHouse>();
        List<String> list_sql = new ArrayList<String>();
        list.add(warehouse1);
        list.add(warehouse2);
        list.add(warehouse3);
        list.add(warehouse4);
        list_sql.add(sql1);
        list_sql.add(sql2);
        list_sql.add(sql3);
        list_sql.add(sql4);
        DataUtil datautil = new DataUtil();
        Connection conn = datautil.getCon();
        PreparedStatement pstmt=null;
        for (int i = 0; i < list.size(); i++) {
            pstmt=conn.prepareStatement(list_sql.get(i));
            pstmt.setString(1,list.get(i).getName());
```

```
3068                    pstmt.setInt(2,list.get(i).getAmount());
3069                    pstmt.setDate(3,list.get(i).getProductionDate());
3070                    System.out.println("正在更新 warehouse"+pstmt.executeUpdate());
3071                }
3072                if(pstmt!=null){
3073                    pstmt.close();
3074                }
3075                datautil.closeCon(conn);
3076            }
3077    }
3078    package com.test.update;
3079    import java.sql.Connection;
3080    import java.sql.PreparedStatement;
3081    import java.text.SimpleDateFormat;
3082    import java.util.ArrayList;
3083    import java.util.List;
3084    import com.test.bpelbean.Shelf;
3085    import com.test.DataUtil;
3086    public class ShelfUpdate {
3087        private String date1 = "2016-03-09";
3088        private String date2 = "2016-02-09";
3089        private String date3 = "2016-05-01";
3090        private String date4 = "2016-08-15";
3091        private String sql1 = "update shelf set Name=?,Amount=?,ProductionDate=?,ShelfNo=?
3092        where id=1";
3093        private String sql2 = "update shelf set Name=?,Amount=?,ProductionDate=?,ShelfNo=?
3094        where id=2";
3095        private String sql3 = "update shelf set Name=?,Amount=?,ProductionDate=?,ShelfNo=?
3096        where id=3";
3097        private String sql4 = "update shelf set Name=?,Amount=?,ProductionDate=?,ShelfNo=?
3098        where id=4";
3099       public void shelfupdate() throws Exception {
3100            int res = 0;
3101            SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd");
3102            java.util.Date date11 = format.parse(date1);
3103            java.util.Date date12 = format.parse(date2);
3104            java.util.Date date13 = format.parse(date3);
3105            java.util.Date date14 = format.parse(date4);
3106            java.sql.Date time1 = new java.sql.Date(date11.getTime());
3107            java.sql.Date time2 = new java.sql.Date(date12.getTime());
3108            java.sql.Date time3 = new java.sql.Date(date13.getTime());
3109            java.sql.Date time4 = new java.sql.Date(date14.getTime());
3110            Shelf shelf1 = new Shelf("cookie", 300, time1, 1);
3111            Shelf shelf2 = new Shelf("milk", 300, time2, 2);
3112            Shelf shelf3 = new Shelf("coca-cola", 300, time3, 3);
3113            Shelf shelf4 = new Shelf("candy", 300, time4, 1);
3114            List<Shelf> list = new ArrayList<Shelf>();
3115            List<String> list_sql = new ArrayList<String>();
3116            list.add(shelf1);
3117            list.add(shelf2);
```

```
3118                list.add(shelf3);
3119                list.add(shelf4);
3120                list_sql.add(sql1);
3121                list_sql.add(sql2);
3122                list_sql.add(sql3);
3123                list_sql.add(sql4);
3124                DataUtil datautil = new DataUtil();
3125                Connection conn = datautil.getCon();
3126                PreparedStatement pstmt=null;
3127                for (int i = 0; i < list.size(); i++) {
3128                    pstmt=conn.prepareStatement(list_sql.get(i));
3129                    pstmt.setString(1,list.get(i).getName());
3130                    pstmt.setInt(2,list.get(i).getAmount());
3131                    pstmt.setDate(3,list.get(i).getProductionDate());
3132                    pstmt.setInt(4,list.get(i).getShelfNo());
3133                    System.out.println("正在执行 shelf"+pstmt.executeUpdate()+"更新");
3134                    System.out.println("正在执行第" + i + "条");
3135                }
3136                if(pstmt!=null){
3137                    pstmt.close();
3138                }
3139                datautil.closeCon(conn);
3140            }
3141        }
3142        package com.test.update;
3143        import java.sql.Connection;
3144        import java.sql.PreparedStatement;
3145        import java.sql.SQLException;
3146        import java.util.ArrayList;
3147        import java.util.List;
3148        import com.test.bpelbean.Product;
3149        import com.test.DataUtil;
3150        public class ProductUpdate {
3151            private Product product1 = new Product("cookie", 120, 1);
3152            private Product product2 = new Product("milk", 1, 2);
3153            private Product product3 = new Product("coca-cola", 60, 1);
3154            private Product product4 = new Product("candy", 120, 2);
3155            String sql1 = "update product set Name=?,ExpirationDate=?,ShelfNo=? where id=1";
3156            String sql2 = "update product set Name=?,ExpirationDate=?,ShelfNo=? where id=2";
3157            String sql3 = "update product set Name=?,ExpirationDate=?,ShelfNo=? where id=3";
3158            String sql4 = "update product set Name=?,ExpirationDate=?,ShelfNo=? where id=4";
3159            public void productupdate() {
3160                int res = 0;
3161                List<Product> list = new ArrayList<Product>();
3162                List<String> list_sql = new ArrayList<String>();
3163                list.add(product1);
3164                list.add(product2);
3165                list.add(product3);
3166                list.add(product4);
3167                list_sql.add(sql1);
```

```
3168            list_sql.add(sql2);
3169            list_sql.add(sql3);
3170            list_sql.add(sql4);
3171            PreparedStatement pstmt=null;
3172            Connection conn=null;
3173            try {
3174                DataUtil datautil = new DataUtil();
3175                 conn= datautil.getCon();
3176                for (int i = 0; i < list.size(); i++) {
3177                    System.out.println(list.size());
3178                    pstmt=conn.prepareStatement(list_sql.get(i));
3179                    pstmt.setString(1,list.get(i).getName());
3180                    pstmt.setInt(2,list.get(i).getExpirationDate());
3181                    pstmt.setInt(3,list.get(i).getShelfNo());
3182                    System.out.println("正在执行 product"+pstmt.executeUpdate()+"更新");
3183                    System.out.println("product" + "sql" + i + "----------"
3184                            + "success");
3185                }
3186            } catch (Exception e) {
3187                e.printStackTrace();
3188            }finally{
3189                try {
3190                    pstmt.close();
3191                    conn.close();
3192                } catch (SQLException e) {
3193                    e.printStackTrace();
3194                }
3195            }
3196        }
3197    }
3198    package com.test.update;
3199    import java.sql.Connection;
3200    import java.sql.PreparedStatement;
3201    import java.sql.SQLException;
3202    import java.util.ArrayList;
3203    import java.util.List;
3204    import com.test.bpelbean.Inventory;
3205    import com.test.DataUtil;
3206    public class InventoryUpdate {
3207        private Inventory inventory1 = new Inventory("cookie", 2, "Enough");
3208        private Inventory inventory2 = new Inventory("milk", 1.5, "Enough");
3209        private Inventory inventory3 = new Inventory("coca-cola", 3, "Enough");
3210        private Inventory inventory4 = new Inventory("candy", 0.5, "Enough");
3211        String sql1 = "update inventory set Name=?,Price=?,State=? where id=1";
3212        String sql2 = "update inventory set Name=?,Price=?,State=? where id=2";
3213        String sql3 = "update inventory set Name=?,Price=?,State=? where id=3";
3214        String sql4 = "update inventory set Name=?,Price=?,State=? where id=4";
3215        String sql5= "update inventory set Name=?,Price=?,State=? where id=5";
3216        }
3217    }
```