K THE SAS CLASSES AND ITS UML DIAGRAMS

K.1 IWO.java Class

```
// The IWO.java class initialises the GUI features for entering/viewing IWO data
package yttjc.sas.blocks;

import java.awt.*;
import javax.swing.JPanel;
import javax.swing.*;
import javax.swing.border.*;
import java.awt.event.*;
import java.io.*;
import java.util.*;

/*      <p>Title: SAS</p>
        <p>Description: South African Sales</p>
        <p>Copyright: Copyright (c) 2005</p>
        <p>Company: University of the Witwatersrand, Johannesburg</p>
        @author I BOTEF
        @version 1.0
 */

public class IWO extends JPanel {
   // Instance variables are declared private to prevent other classes to manipulate
   // the GUI data. Then, the get and set methods are used to access them.
   private JPanel panel1 = new JPanel();
   ...
   private JTextField tfDescription = new JTextField();
   private JButton jButtonSave = new JButton();
```

```
/* The IWO class calls the JComboBox constructor and supplies an Object of
strings array to create its list of suggested input items. */
   Object[] dataObjectMaterial = {"Please Select", "Bar Stock", "Forging",
       "Casting", "Bought Out Component", "Material from Customer"};
   private JComboBox comboMaterial = new JComboBox(dataObjectMaterial);
   …
   public IWO() {
      try {
        jbInit();
      }
      catch(Exception ex) {
         ex.printStackTrace();
      }
   }

   private void jbInit() throws Exception { …
     jPanelIWO.add(tfJobNo, null);
     jPanelIWO.add(tfDescription, null);
     jPanelMaterial.add(comboMaterial, null);
     comboMaterial.setSelectedIndex(0);

/* When the user explicitly makes a selection in the JComboBox, its
getSelectedItem method returns the currently selected item; the JComboBox fires
an ActionEvent; and the actionPerformed method defines what should happen. */
     comboMaterial.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(ActionEvent e) {
           comboMaterial_actionPerformed(e);
        }
     });
     …
     initializeFields();
  }// End constructor. Calling the constructor from another class, builds the GUI.
```

```
/* To collect the GUI data, public methods are used to write/read the object
to/from a file. Their functionality is provided in the controller class. */
   public JButton getSave() {return jButtonSave;}
   public JButton getClose() {return jButtonClose;}

   public void cancel() {
      this.setVisible(false);
   }

/* The JComboBox's class getSelectedItem method returns the selected item as an
Object. To convert, or cast, the Object as a String object, so it could be compared
with other strings and later passed on, the getSelectedItem method is called with
the dot operator; then cast the Object to a String object type enclosed in
parentheses; and finally assign it to scomboMaterial String variable.*/
   void comboMaterial_actionPerformed(ActionEvent e) {
      scomboMaterial = (String) comboMaterial.getSelectedItem();
   }

/* To collect the GUI data, it would be too time consuming to call the getText
method, assign the result to a variable for each item, and then write it to a file.
Instead, the getText is called on each item and store the results for all in one
public Vector object variable which could: hold a variety of objects; be accessed
using its integer index; grow and shrink as elements are added or removed;
include primitive types using the class wrapper; receive new objects with the add
method by calling the getText method for each variable associated with text fields,
menus, and radio buttons, except the pull-down menu result already assigned to
"scomboMaterial"; and then use the getUIFields method to store the Vector.*/
   public Vector getUIFields() {
      Vector fieldValues = new Vector();
      fieldValues.add(tfDescription.getText());
      fieldValues.add(scomboMaterial);
      …
```

```
        return fieldValues; // returns fieldValues, the Vector object.
    } // closes getUIFields

/* The setFields method reads the object back into the GUI fields. Its functionality
is found in the IWOHandler class. */
    public void setFields(String string[])
        throws IllegalArgumentException {
          tfJobNo.setText(string[0]);
          tfCustomer.setText(string[1]);
          tfDescription.setText(string[2]);
          …
          comboMaterial.setSelectedItem(string[36]);
          …
    }//end of method setFields
}//end of class
```

K.2 IWOInfo.java Class

```
package yttjc.sas.blocks;

import java.io.*;
import java.io.Serializable;

public class IWOInfo implements Serializable {
    private String jobDescription, String jobcomboMaterial, … ;
    …
    public IWOInfo() {
        setIWO();
    }

    // setIWO method contains variables that do not take any arguments
    public void setIWO () {
```

```
        jobDescription = "";
        jobcomboMaterial = ""; …
    }

    // setIWO method takes the variables as arguments
    public void setIWO (String hDescription,   … String hcomboMaterial, …) {
        this.jobDescription = hDescription;
        this.jobcomboMaterial = hcomboMaterial; …
    }
    // Override the toString method of the String class so that toString
    // returns the data associated with the instance variables
    public String toString() {
        return jobDescription + … + jobcomboMaterial + …
    }

    // Provide a get method for each instance variable, e.g.
    public String getJobJobNo() {
        return jobJobNo;
    }
   …
}//End of class
```

K.3 IWOHandler.java Class

```
package yttjc.sas.blocks;

import java.awt.*;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.*;
import java.awt.event.*;
import java.util.Vector;
```

```
import java.awt.geom.*;
import java.io.*;

public class IWOHandler extends JFrame {
   private IWO ui = new IWO();
   private IWOInfo iwor;
   private BorderLayout borderLayout1 = new BorderLayout();
   protected JButton saveButton, closeButton;
   protected String hJobNo, hCustomer, hDescription, …;
   private ObjectOutputStream output; //variable for output object stream
   private ObjectInputStream input; //variable for input object stream
   protected String filename;

   public IWOHandler() {
      this.setTitle("Sales Internal Works Order");
      this.getContentPane().add("Center",ui);

      saveButton = ui.getSave();
      closeButton = ui.getClose();

/* Anonymous inner class, without naming them, has the benefit of keeping the
action that handles the code in the same place with the GUI controller. */
      saveButton.addActionListener (
         new ActionListener () {
            public void actionPerformed (ActionEvent ae) {
               saveIWO();
            } //Close method
         } //Close inner class
      ); //Close argument and block

      filename = "database/sales/iwoex.dat";
      viewIWO();
```

```
}//End constructor

public void saveIWO() {
    Vector values = ui.getUIFields();
    //Gets each item and assigns it to a variable of the proper type.
    String hJobNo = (String)values.get(0);
    String hCustomer = (String)values.get(1);
    String hcomboMaterial = (String)values.get(36); …
   //Calls the setIWO method from IWOInfo to create the IWO object, and
   // passes the above assigned items
   try {
     iwor = new IWOInfo();
     iwor.setIWO( hJobNo, hCustomer, hDescription, …, hcomboMaterial, … );

/* The ObjectOutputStream writes objects/primitive data types to an
OutputStream only if the objects support the java.io.Serializable interface; then
each serializable object is encoded containing: the class name and signature,
object's fields and arrays values, and any other referenced objects. */
     FileOutputStream file = new FileOutputStream(filename);
     ObjectOutputStream output = new ObjectOutputStream(file);
     output.writeObject(iwor);
     output.flush();
     output.flush();
        if(input != null) {
          input.close();
          input = null;
        }
   }

//The ObjectInputStream object reads and deserialize objects/primitive data types
  public void viewIWO() {
    try {
```

```
        IWOInfo dr;
        if(input == null) {
          try{
            input = new ObjectInputStream(new FileInputStream(filename)); }
          catch(IOExceptionioex){JOptionPane.showMessageDialog(null, "Error
                during reading file", null, JOptionPane.ERROR_MESSAGE);}
        }

        iwor = (IWOInfo) input.readObject();
        String svalues[] = {
          String.valueOf(iwor.getJobJobNo()),
          String.valueOf(iwor.getJobcomboMaterial()),…
        };

        ui.setFields(svalues);
        return;
      }//close try then catch e.g. ClassNotFoundException
    }//end method viewIWO
}//end of class
```

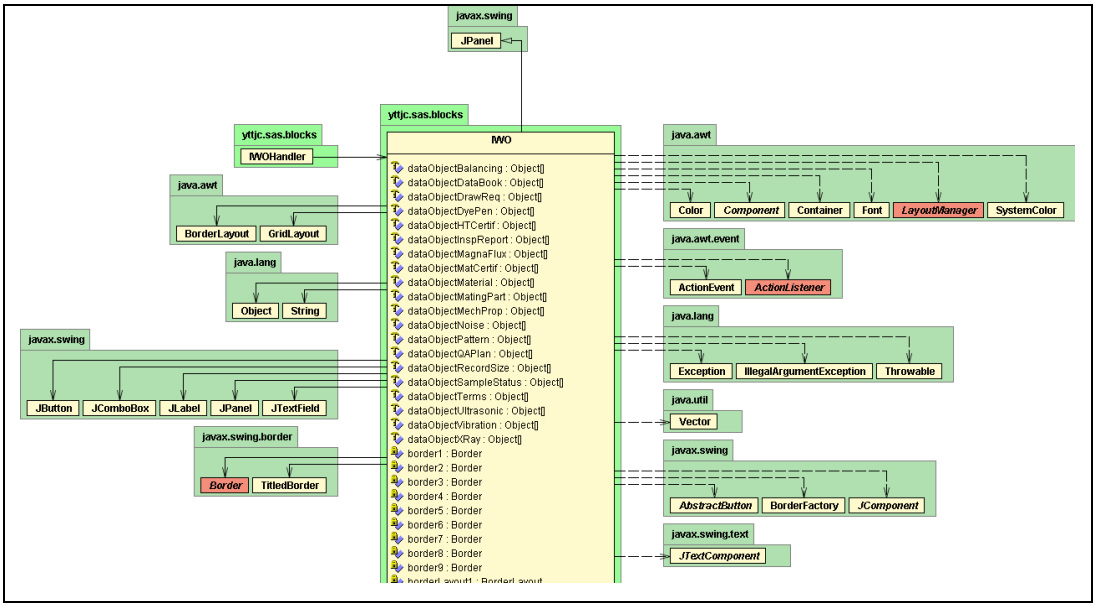

Figure K.1 IWO JBuilder UML diagram

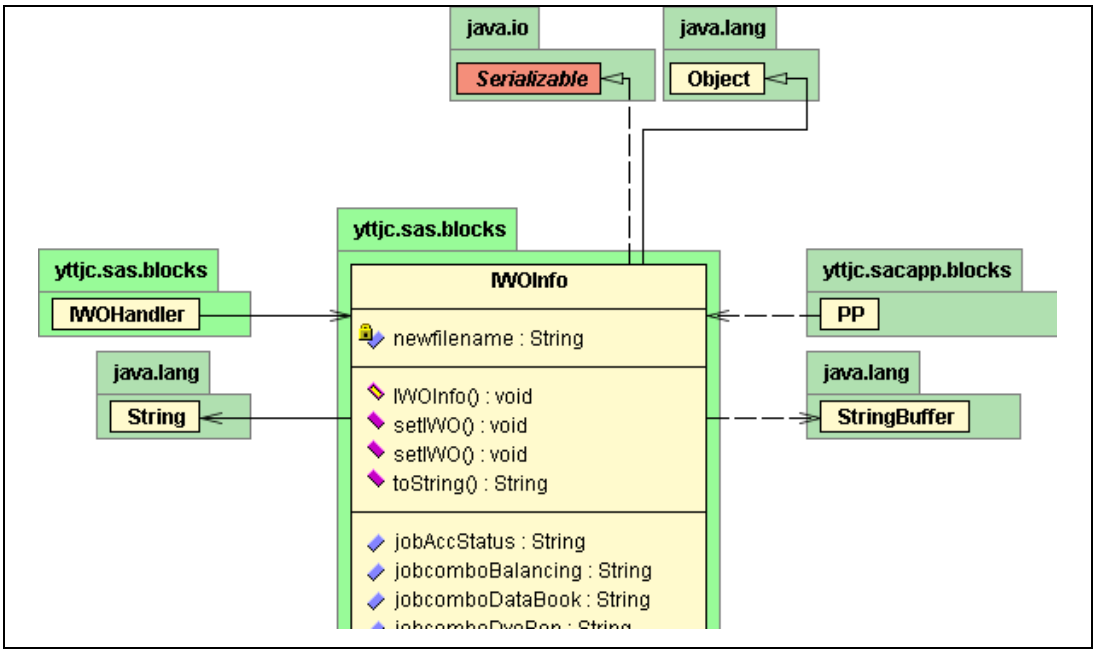

Figure K.2 IWOInfo JBuilder UML diagram

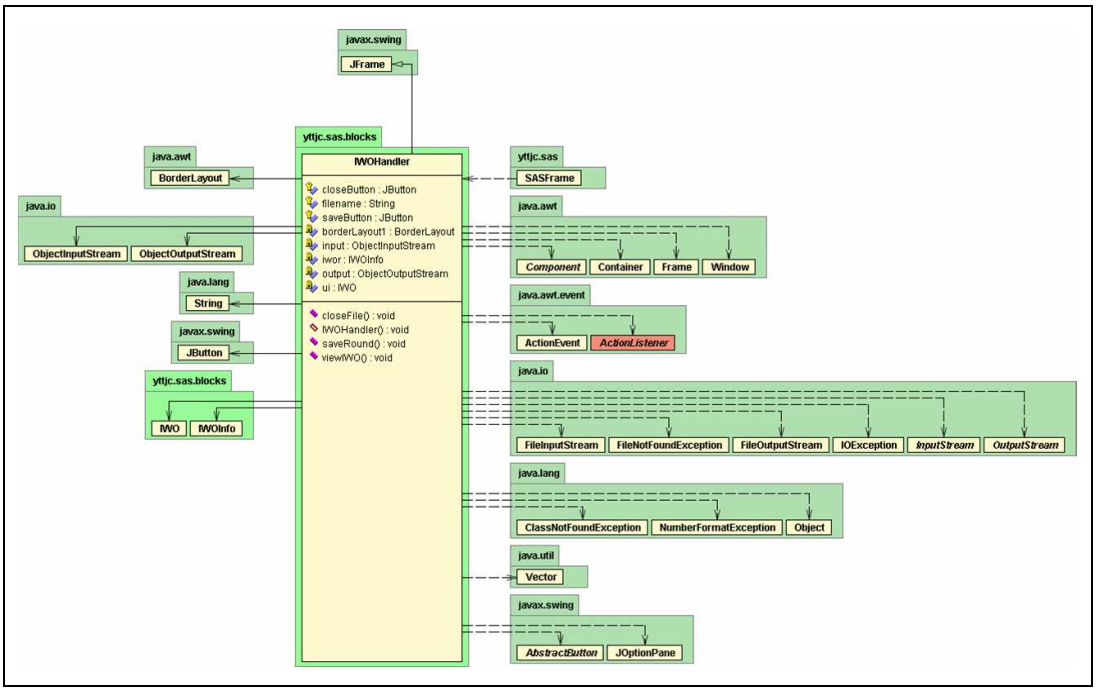

Figure K.3 IWOHandler JBuilder UML Diagram