International Journal of Current Advanced Research
ISSN: O: 2319-6475, ISSN: P: 2319-6505, Impact Factor: 6.614
Available Online at www.journalijcar.org
Volume 11; Issue 08 (B); August 2022; Page No.1434-1438
DOI: http://dx.doi.org/10.24327/ijcar.2022.1438.0318

Research Article

DEVELOPMENT OF THE RECOGNITION SOFTWARE FOR THE DIESEL ENGINE CRANKSHAFT AXIS STATUS BASED ON JAVA LANGUAGE

GuoJunwu and Kong Degang

Shanghai Maritime University, Marine Merchant College, Shanghai, China, No.1550 Haigang Avenue

A R T I C L E   I N F O

Article History:

Received 13th May, 2022
Received in revised form 11th June, 2022
Accepted 8th July, 2022
Published online 28th August, 2022

Keywords:

Java language, diesel engine crankshaft, crankweb deflection, crankshaft axis status

A B S T R A C T

The paper uses Java language of the computer to develop the recognition software for the diesel engine crankshaft axis state. The software adopts four modules of DrawAxes, Axes, DrawBtnListener and Reset Btn Listener to realize the three functions such as input, output and drawing, according to the measured data of the diesel engine crankweb deflection, these functions can realize the roles of computing the crankweb inflection and drawing the crankshaft axis status, finally recognizes the diesel engine crankshaft axis state.

INTRODUCTION

The Java language is a simple, object-oriented, distributed, interpreted, robust, secure, structure-neutral, portable, high-performance, multithreaded, dynamic language.

When SUN introduced the Java language in 1995, the world's attention was drawn to this magical language. The Java language was originally called OAK in 1991, and was designed by SUN as a general-purpose environment for some consumer electronics products. Scholars at home and abroad have studied the characteristics of JAVA language.

Ancona, D etc. [1] showed that it was possible to define type systems for Java-like languages, which, in contrast to those used by standard compilers, had principal typings, hence could be used as a basis for selective recompilation.

Tan Gang etc. [2] proposed ILEA (stands for Inter-LanguagE Analysis), which was a framework that enabled existing Java analyses to understand the behavior of C code.

Freund SN etc. [3] developed a precise specification of statically correct Java bytecode, in the form of a type system. Their focus was a subset of the bytecode language dealing with object creation and initialization.

AssiriFatmahYousef etc. [4] developed Java Exceptions in the NL (JENL) tool to process Java exceptions using NLG techniques. Empirical evaluation has shown extreme satisfaction among Java programming subjects (including student novice programmers) with the capabilities of JENL as well as its usability.

Bettini Lorenzo etc. [5] gave a formal account of our proposal through a core calculus, FDTJ (FEATHERWEIGHT DYNAMIC TRAIT JAVA), equipped With a static type system guaranteeing that in a well-typed program no runtime type error would take place.

Though the Java programming language was designed with extreme care, there were still a few ambiguities and irregularities left in the language. The ambiguities were those issues that were not defined clearly in the Java language specification. The problems of ambiguity, irregularity, and dependence on implementations frequently trapped an incautious Java programmer. Some suggestions and solutions for the problems were provided by Chan JT etc. [6]

Dmitriev M [7] described a make technology for the Java programming language, that was based on smart dependency checking, guarantees consistency of the project code, and at the same time reduced the number of source code recompilations to the minimum.

Cohen Tal etc. [8] presented an overview of JTL (the Java Tools Language, pronounced "Gee-tel"), a novel language for querying JAVA programs. JTL was designed to serve the development of source code software tools for JAVA, and as a small language to aid programming language extensions to JAVA.

Servetto Marco etc. [9] proposed a Java-like language where class definitions were first class values and new classes could be derived from existing ones by exploiting the full power of

Corresponding author:* **GuoJunwu
Shanghai Maritime University, Marine Merchant College, Shanghai, China, No.1550 Haigang Avenue

the language itself, used on top of a small set of primitive composition operators, instead of using a fixed mechanism like inheritance.

Cordoba-Sanchez Irene [10] described a new modelling language for the effective design and validation of Java annotations.

League C etc. [11] present an efficient encoding of core Java constructs in a simple, implementable typed intermediate language.

Many contemporary object-oriented programming languages supported first-class queries or comprehensions. These JAVA language extensions made it easier for programmers to write queries, but were generally implemented no more efficiently than the code using collections, iterators, and loops that they replaced. [12]

The JastAdd Extensible Java Compiler was a high quality Java compiler that was easy to extend in order to build static analysis tools for Java, and to extend Java with new language constructs. It was built modularly, with a Java 1.4 compiler that was extended to a Java 5 compiler. [13]

Parnin Chris etc. [14] reported on the first empirical investigation into how Java generics had been integrated into open source software by automatically mining the history of 40 popular open source Java programs, traversing more than 650 million lines of code in the process, and evaluated five hypotheses and researched questions about how Java developers used generics.

No formal specification of the bytecode verifier existed in the Java Virtual Machine Specification published by Sun. Freund SN etc. [15] developed such a specification in the form of a type system for a subset of the bytecode language.

According to the research results of domestic and foreign experts, the excellent features of the Java language makes Java applications extremely robust and reliable, and also reduces the maintenance cost of the application system. Java's full support for object technology and the API embedded in the Java platform can shorten the development time and reduce the cost of application systems. Java's compile once, runnable nature allows it to provide an open architecture that can be used anywhere and a low-cost way to transfer information between multiple platforms. Therefore, a set of diesel engine crankshaft axis status recognition software can be designed for marine engineers by using advanced Java language, so as to judge the diesel engine axis status quickly, accurately and intuitively in real ship work and ensure the normal operation of diesel engine.

Install the development environment

The software can be developedon an ordinary computer, the steps are as follows:

1. Install the jdk-1_5_0_05-windows-i586-p.exe package on aordinary computer first.
2. And then jfreechart - 0.9.21.Jar/gnujaxp jar/jcommon - 0.9.6. Jar file copy to C: \ Program Files \ Java \ jdk1.5.0 _05 \ lib directory.
3. Configure environment variables:

My computer -> Properties -> Advanced -> Environment Variables -> System Variables -> Find the path item -> add "to the variable value"; C: Program Files\Java\jdk1.5.0_05\bin "-> Click OK
My Computer -> Properties -> Advanced -> Environment Variables -> System Variables -> New CLASSPath entry -> Enter variable values.
".;C:\ProgramFiles\Java\jdk1.5.0_05\lib\dt.jar;C:\ProgramFiles\Java\jdk1.5.0_05\lib\tools.jar;C:\ProgramFiles\Java\jdk1.5.0_05\lib\gnujaxp.jar;C:\ProgramFiles\Java\jdk1.5.0_05\lib\jcommon-0.9.6.jar;C:\ProgramFiles\Java\jdk1.5.0_05\lib\jfreechart-0.9.21.Jar"－>Click OK
At this point, the development environment has been installed and configured successfully.

Identification software design and description

In order to realize the three functions of input, output and drawing, the following four modules of DrawAxes, Axes, DrawBtnListener and ResetBtnListener are needed in this software.

DrawAxes module

Through DrawAxes, the function of data input, calculation and display can be realized. The specific process is as follows:

Step 1: Define various arrays to store data and computed results, as well as various visual objects such as labels, text boxes, and so on.
Step 2: Add the various visual objects to the top-level container of the JApplet to display the various objects.
Step 3: Design the input function and read the measurement data of the input text box.
Step 4: According to the calculation formula of arm distance difference, design the calculation function, calculate the arm distance difference and store it in the result array.
Step 5: Design the result display function to display the calculation results and use Axes module to draw the coordinate graph.

The specific implementation of the key code is as follows:
// Variable definition section:
private static final double INCREMENT = -0.05;//Set the incremental
int symbol = 12;//Set flag bit
Container c = getContentPane();//The container
doublevalueTopPoint[]=new double[13];//The top dead center data
doublevalueLeftUnderPoint[]=new double[13];// The left bottom dead center data
doublevalueRightUnderPoint[]=new double[13];//The right bottom dead center data
doublevalueLeftPoint[]=new double[13];//The left point data
doublevalueRightPoint[]=new double[13];//The right point data
doublevalueResultTU[]=new double[13];//The result of ∆⊥ calculation
oublevalueResultLR[]=new double[13];//The result of ∆_ calculation
doublevalueAxes[]=new double[15];//Draw coordinate chart data
LineBorderlb = new LineBorder(Color.BLACK);//Creating a border Instance
JButtonbtnDraw=new JButton(" drawing "); // Draw button
JButtonbtnReset=new JButton(" reset "); // Reset button
JLabellblCylinderNum=new JLabel(" cylinder ", JLabel. CENTER); // Cylinder number label

```
JLabellblTopPoint=new JLabel(" TOP dead center
",JLabel.CENTER);
JLabellblLeftUnderPoint=new JLabel("Left bottom dead
center ",JLabel.CENTER);
JLabellblRightUnderPoint=new JLabel("Right bottom dead
center ",JLabel.CENTER);
JLabellblLeftPoint=new JLabel ("At "left
point",JLabel.CENTER);
JLabellblRightPoint=new JLabel("At "right
point",JLabel.CENTER);
JLabellblTopResultUnder=new
JLabel("△⊥",JLabel.CENTER);
JLabellblLeftResultRight=new
JLabel("△_",JLabel.CENTER);
Image img=null;//Create a storage image variable
ImageIconimgIcon;//Create icon
JLabeljLblExplain=new JLabel();//Create a display image
label
staticJPanelaxesGraph=new JPanel(null);//Coordinate line
chart
ResetBtnListenerrstBListener=new
ResetBtnListener(this);//Create a button listener
DrawBtnListenerdraBListener=new
DrawBtnListener(this);//Create a button listener
```

// Interface display part:

```
c.setLayout(null);//Set up the layout manager
c.add(btnDraw);//Add drawing button
c.add(btnReset);//Add reset button
c.add(lblCylinderNum);//Add a cylinder number label
c.add(lblTopPoint);
c.add(lblLeftUnderPoint);
c.add(lblRightUnderPoint);
c.add(lblLeftPoint);
c.add(lblRightPoint);
c.add(lblTopResultUnder);
c.add(lblLeftResultRight);
c.add(jLblExplain); // Add the display image tag
c.add(axesGraph); // Add coordinate line graph
btnDraw.setBounds(20, 474, 100, 30); // Set the button
position
btnReset.setBounds(140, 474, 100, 30);
lblCylinderNum.setBounds(0, 0, 69, 30);
lblTopPoint.setBounds(0,30,69,30);
lblLeftUnderPoint.setBounds(0,60,69,30);
lblRightUnderPoint.setBounds(0,90,69,30);
lblLeftPoint.setBounds(0,120,69,30);
lblRightPoint.setBounds(0,150,69,30);
lblTopResultUnder.setBounds(0,180,69,30);
lblLeftResultRight.setBounds(0,210,69,30);
lblResultTU1.setBounds(); // Set the result label position
lblResultLR1.setBounds(); // Set the result label position
txtTopPoint1.setBounds(); // Sets the position of the input text
box
txtLeftUnderPoint1.setBounds(); // Sets the position of the
input text box
txtRightUnderPoint1.setBounds(); // Sets the position of the
input text box
txtLeftPoint1.setBounds(); // Sets the position of the input text
box
txtRightPoint1.setBounds(); // Sets the position of the input
text box
btnReset.addActionListener(rstBListener); // Register reset
button
btnDraw.addActionListener(draBListener); // Register the
drawing button
JLblExplain.SetBounds (0240260234); // Set the image label
position
AxesGraph.SetBounds (260240385264); // Set the position of
the coordinate line chart
try{
img=getImage(new URL(getCodeBase(),"Explain.jpg")); //
Get the image
}
catch(Exception e){e.printStackTrace();}
imgIcon=new ImageIcon(img); // Generate an icon
jLblExplain.setIcon(imgIcon); // Display the image
jLblExplain.setHorizontalAlignment(SwingConstants.CENTE
R); // Set the image to display
c.setVisible(true); // Display all controls}
```

// Function function:

```
Public void setSymbol()// Determines and sets the Symbol
value
{
inti=0;
for(i=1;i<=12;i++)
{
if(valueTopPoint[i]==0&&valueLeftUnderPoint[i]==0&&valu
eRightUnderPoint[i]==0&&valueLeftPoint[i]==0&&valueRig
htPoint[i]==0)      {symbol=i-1;break;}
else        symbol=i;
}
}
Public void getValuePoint()// Obtain measurement point data
Public void calculate()// Calculate the arm length difference
{
int counter=0;
int sign=symbol+1;
for(counter=1;counter<=symbol;counter++)
{
valueResultTU[counter]=valueTopPoint[counter]-
((valueLeftUnderPoint[counter]+valueRightUnderPoint[counte
r])/2.0);
valueResultLR[counter]=valueLeftPoint[counter]-
valueRightPoint[counter];
BigDecimalbTU = new
BigDecimal(Double.toString(valueResultTU[counter]));
valueResultTU[counter]=bTU.setScale(2,
BigDecimal.ROUND_HALF_UP).doubleValue(); //Round up
or down
BigDecimalbLR = new
BigDecimal(Double.toString(valueResultLR[counter]));
valueResultLR[counter]=bLR.setScale(2,
BigDecimal.ROUND_HALF_UP).doubleValue();///Round up
or down

}
for(counter=2;counter<=sign;counter++)
{valueAxes[counter]=valueResultTU[counter-1];}
valueAxes[1]=INCREMENT+valueResultTU[1];
valueAxes[sign+1]=INCREMENT+valueResultTU[symbol];
valueAxes[0]=sign+1;
}
Public void resetValuePoint()// Clears data
Public void resetTextLabel()// Clears text boxes and label
values
Public static JPanelgetAxesGraph(
```

Public void setLabelValue()// Sets the label to display the calculated result

Axes module

Axes class was used to realize the drawing function of Crankshaft Deflection coordinate figure and the specific process was as follows:

Step 1: Define the valuePoint array to hold the calculation results received from the DrawAxes module, that is, the raw data to draw the coordinate graph.

Step 2: Create xySeries data list with for(I =1; i<=valuePoint[0]; I ++)xyseries. Add (i-1,valuePoint[I]) statement, generate coordinate graph data point, and add xyseries data list. Next, create the XySeries Collection dataset, which is a coordinate data source recognized by the plotting function, and add the XYSeries data list to the dataset.

The third step: through jfreechart = ChartFactorycreateXYLineChart framework () method to generate the coordinate chart. The createChart() method is used to draw the coordinate graph from the xySeriesCollection data set as the data source.
Step 4: Add the ChartPanel graphics container containing the Jfreechart coordinates to the DrawAxes image display control axesGraph.
The key code for drawing the coordinate graph is as follows:

```
static double valuePoint[]=new double[15];
public Axes(String s,double temp[])
{   super(s);
valuePoint=temp;
DrawAxes.getAxesGraph().removeAll(); // Clear the original image in the display control
XYDatasetxydataset = createDataset();
JFreeChartjfreechart = createChart(xydataset);
ChartPanelchartpanel = new ChartPanel(jfreechart);
Chartpanel.SetPreferredSize (new Dimension (385264)); // Set the canvas size
setContentPane(chartpanel); // Display the coordinates
DrawAxes.getAxesGraph().add(chartpanel);
}
Private static XYDatasetcreateDataset()// Create coordinate data point
{   inti=0;
XYSeries = new XYSeries(" XYSeries "); // Create a list of data
for(i=1; i<=valuePoint[0]; i++)xyseries.add(i-1,valuePoint[i]);
XYSeriesCollectionxyseriescollection = new XYSeriesCollection(); // Create a data set
xyseriescollection.addSeries(xyseries);
returnxyseriescollection; }
Private static JFreeChartcreateChart(XYDatasetXYDataset)// Draw the coordinate diagram
{JFreeChartJFreeChart = ChartFactory. CreateXYLineChart (" Crankshaft Deflection coordinate figure ", "shaft arm is apart from the difference", "+ delta - delta, "xydataset, PlotOrientation. VERTICAL, true, true, false);
NumberAxisdomainaxis = (NumberAxis)xyplot.getDomainAxis(); // Coordinates are graduated in units spaced apart
domainaxis.setTickUnit(new NumberTickUnit(1D)); // Coordinates are graduated in units spaced apart
standardxyitemrenderer.setSeriesPaint(0,Color.BLACK); // Set the line color
returnjfreechart; }
```

DrawBtnListener module

Through the DrawBtnListener class, to achieve the response function of the drawing button, the specific implementation process is as follows:

Step 1: Define the DrawAxes variable x and set the object instances that the module will operate on.
Step 2: Get the input data from the input module using the getValuePoint() method.
Step 3: Calculate the arm distance difference using the Calculate () method.
Step 4: Display the calculated results through setLabelValue() method.
Step 5: Call Axes module to generate coordinate graph.
The specific implementation of the key code is as follows:

```
DrawAxes x;
    DrawBtnListener(DrawAxes temp)  {  x=temp;  }
    public void actionPerformed(ActionEvent e)
{ x.getValuePoint();
x.setSymbol(); // Set the number of cylinders
x.calculate();
x.setLabelValue();
Gra = new Axes("Crankshaft Deflection coordinate diagram", x.valueaxes);
gra.pack();
x.axesGraph.setVisible(true); // display coordinates}
```

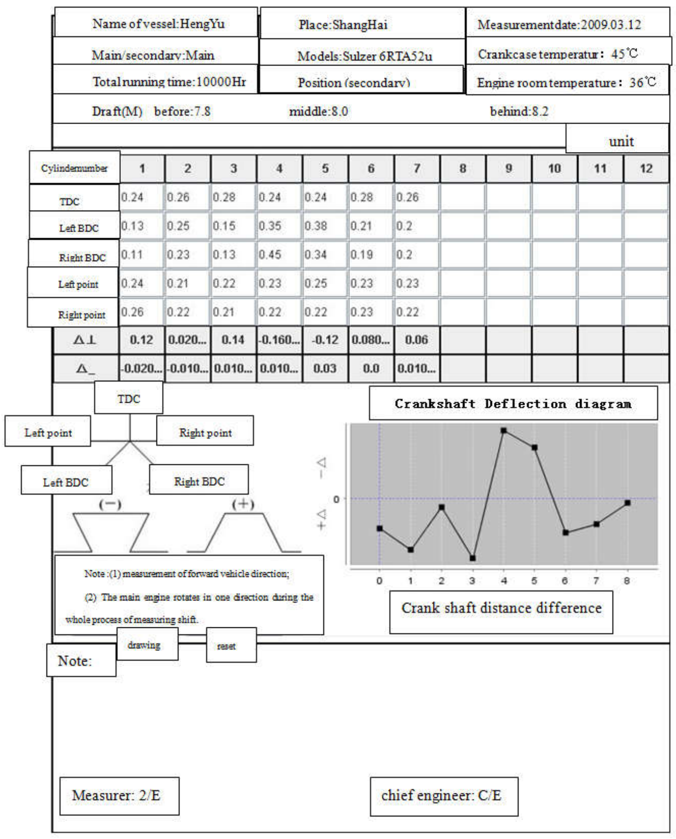
ResetBtnListener module

Through the class ResetBtnListener, to achieve the reset button response function, the specific implementation process is as follows:
Step 1: Define the DrawAxes variable x and set the object instances that the module will operate on.
Step 2: Clear the calculated results of the DrawAxes display area with resetTextLabel().
Step 3: Clear the data stored in the DrawAxes module using the resetValuePoint() method.
Step 4: Clear coordinate point data in Axes module by resetAxesValue() method.
The specific implementation of the key code is as follows:

```
DrawAxes x;
ResetBtnListener(DrawAxes temp)  {  x=temp;  }
public void actionPerformed(ActionEvent e)
   {  x.resetTextLabel();
                  x.resetValuePoint();
                  Axes.resetAxesValue();
                  x.axesGraph.setVisible(false);//Clear expired coordinates
}
```

Use of identification software

Run the program to get the measurement record and coordinate chart of the crankshaft of the main and secondary engines, directly input the actual measured arm distance difference data of each cylinder in the table, and finally output the crankshaft axis state curve of the diesel engine and the calculation results of the measurement data. See Figure 1.



Name of vessel:HengYu	Place:ShangHai	Measurementdate:2009.03.12
Main/secondary:Main	Models:Sulzer 6RTA52u	Crankcase temperatur：45℃
Total running time:10000Hr	Position (secondary)	Engine room temperature：36℃

Draft(M) before:7.8 middle:8.0 behind:8.2

unit

Cylindernumber	1	2	3	4	5	6	7	8	9	10	11	12
TDC	0.24	0.26	0.28	0.24	0.24	0.28	0.26					
Left BDC	0.13	0.25	0.15	0.35	0.38	0.21	0.2					
Right BDC	0.11	0.23	0.13	0.45	0.34	0.19	0.2					
Left point	0.24	0.21	0.22	0.23	0.25	0.23	0.23					
Right point	0.26	0.22	0.21	0.22	0.22	0.23	0.22					
Δ⊥	0.12	0.020...	0.14	-0.160...	-0.12	0.080...	0.06					
Δ_	-0.020...	-0.010...	0.010...	0.010...	0.03	0.0	0.010...					

Note :(1) measurement of forward vehicle direction;

(2) The main engine rotates in one direction during the whole process of measuring shift.

Note:

Measurer: 2/E

chief engineer: C/E

Figure 1 Axis status diagram of marine engine

CONCLUSION

In this paper, Java language is used to develop the recognition software for the diesel engine crankshaft axis status, the system can quickly and accurately get the diesel engine crankshaft axis status recognition charts, engine management personnel can judge the crankshaft center line status by the crankshaft status recognition chart.

Acknowledgements

In writing this paper, I have benefited from the presence of my colleagues. They generously helped me collect the materials and offered many invaluable suggestions. I hereby extend my grateful thanks to them for their kind help, without which the paper would not have been what it is.

Conflicts of Interest

The author declares that there is no conflict of interest regarding the publication of this paper.

Data Availability Statement

The data used to support the findings of this study are included within the article.

Funding Statement

This work was supported by the National Natural Science Foundation of China. Grant number is 11272213.

References

1. Ancona, D; Zucca, E.Principaltypings for Java-like languages.ACM SIGPLAN NOTICES. 2004, 39(1): 306-317
2. Tan Gang; Morrisett Greg. ILEA: Inter-language analysis across Java and C. ACM SIGPLAN NOTICES. 2007, 42(10): 39-56
3. Freund SN; Mitchell JC.A type system for object initialization in the Java bytecode language.ACM Transactions On Programming Languages And Systems. 1999, 21(6): 1196-1250
4. AssiriFatmahYousef; ElazharyHanan.Automated Java e xceptions explanation using natural language generation techniques. Computer Applications in Engineering Education. 2020, 28(3): 626-644
5. Bettini Lorenzo; Capecchi Sara; DamianiFerruccio.On flexible dynamic trait replacement for JAVA-like languages. Science of Computer Programming. 2013, 78(7): 907-932
6. Chan JT; Yang W; Huang JW. Traps in Java. Journal of Systems and Software. 2004, 72(1): 33-47
7. DmitrievM. Language-specific make technology for the Java (TM) programming language. ACM Sigplan Notices. 2002, 37(11): 373-385
8. Cohen Tal; Gil Joseph; MamanItay. JTL - the Java tools language. ACM Sigplan Notices. 2006, 41(10): 89-108
9. Servetto Marco; Zucca Elena. Metafjig A Meta-Circular Composition Language for Java-like Classes. Acm Sigplan Notices. 2011, 45(10): 464-483
10. Cordoba-Sanchez Irene; de Lara Juan. Ann: A domain-specific language for the effective design and validation of Java annotations. Computer Languages Systems & Structures. 2016, 45: 164-190
11. League C; Shao Z; Trifonov V. Type-preserving compilation of featherweight Java.ACM Transactions On Programming Languages And Systems. 2002, 24(2): 112-152
12. Willis Darren; Pearce David J.; Noble James. Caching and Incrementalisation in the Java Query Language. Acm Sigplan Notices.2008, 43(10): 1-17
13. Ekman Torbjorn; Hedin Gorel.The JastAdd Extensible Java Compiler. ACM Sigplan Notices.2007, 42(10): 1-17
14. Parnin Chris; Bird Christian; Murphy-Hill Emerson. Adoption and use of Java generics. Empirical Software Engineering. 2013, 18(6): 1047-1089
15. Freund SN; Mitchell JC.A type system for the Java bytecode language and verifier. Journal Of Automated Reasoning. 2003, 30(3): 271-321

How to cite this article:

GuoJunwu and Kong Degang (2022) 'Development of The Recognition Software For The Diesel Engine Crankshaft Axis Status Based on Java Language', *International Journal of Current Advanced Research,* 11(08), pp. 1434-1438.
DOI: http://dx.doi.org/10.24327/ijcar.2022.1438.0318

****