The Program includes a framework and several algorithms.

Interface

The interface is finished using JAVA Swing. The main interface is a *JFrame*.

MainFrame

MainFrame is a subclass of *JFrame*. It shows some tab panels and a status bar as shown in Figure 1.

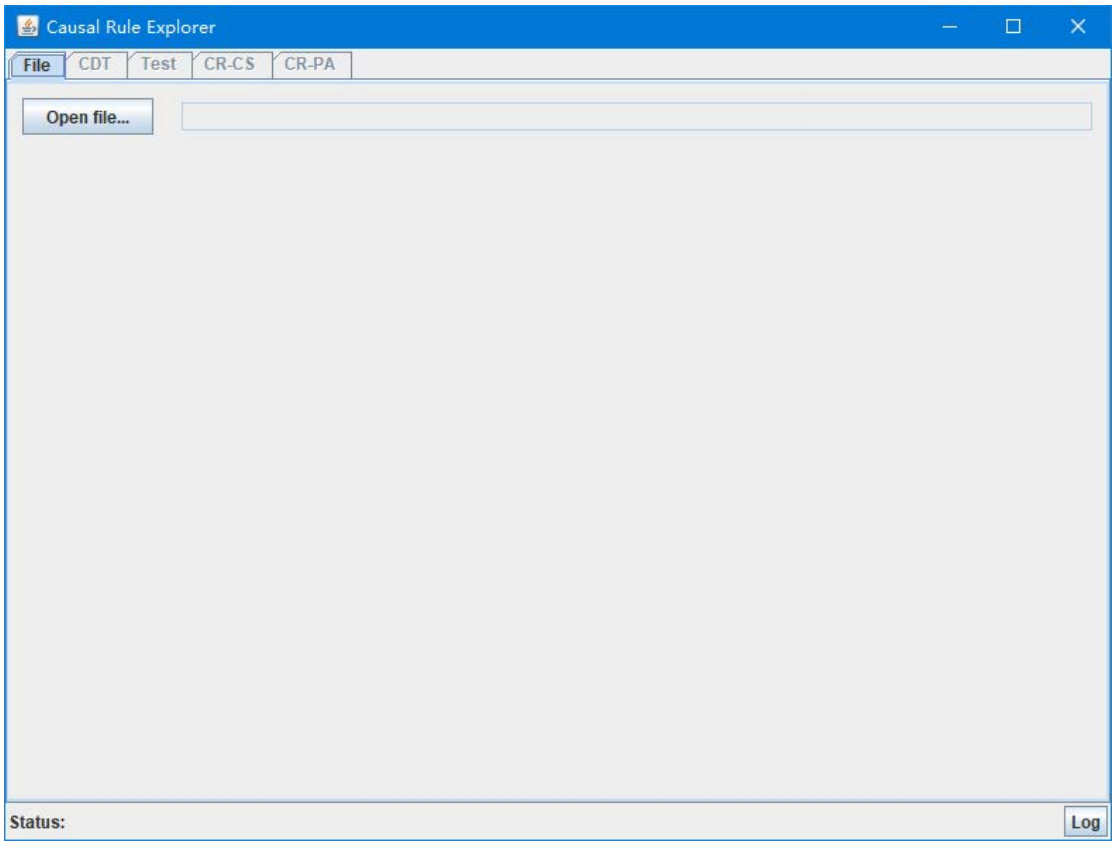

Figure 1 MainFrame

MainFrame implements some interfaces like *MainFrameEventHandler* and *CanShowStatus*. The former limits the access from other Class. The latter makes *MainFrame* to be a "Show Status Provider".

MainFrame reassign the standard output stream and the standard error output stream. Click "Log" button to access.

FilePanel

Until now, *FilePanel* has only one function, choose a file. When user chooses a file, this class inform the *MainFrameEventHandler*(actually *MainFrame*) by calling its function *selectANewFile(File)*.

Notice: user may choose a file repeatly even when one algorithm is calculating.

AlgorithmPanel

When user chooses a file in *FilePanel*, several *AlgorithmPanel* become available. This kind of panel consisits of three parts. Left-top is the section in which user can configure the algorithm. Left-bottom is the section which has “start button” and “stop button” and shows history of each transaction. When we change the item in history, the center part, a *JTextArea* will show the outputs. The center part can also show diagrams.

How to integrate an algorithm

An algorithm must extend *AbstractAlgorithm*. An algorithm must have some configurations, like a threshold. We store these configuration in a class. And this class will be the member of the algorithm.

First, we need to create a class as the map of configuration.
Second, we need to create a class which extends *AbstractAlgorithm*.
Finally, modify some lines in *MainFrame.java*.

How to write the class as the map of configuration

The class should implement *Cloneable* interface. Because when user start a calculation, the algorithm which is used is only a copy of the original algorithm and the original algorithm may be modified when calculation is still going. The class as the map of configuration is part of the algorithm, as Figure 2.

```
@Override
public Object clone() {
    CDTConfig newC = null;
    try {
        newC = (CDTConfig) super.clone();
    } catch (CloneNotSupportedException e) {
        e.printStackTrace();
    }
    return newC;
}
```

Figure 2 the "configuration class"

The class should override *toString*. This function is called when show the configuration.

The types which configuration can use and extra functions are shown below. These attributes must have setters and getters. Assume the name of attribute is XXX. Each attributes may have function "*String getXXXComment()*" to provide the comment of the attribute and function "*String getXXXShownName()*" to provide the shown name.

Data Type	View (Corresponding to the attribute and created automatically)	functions	remark
int	JTextField	int getXXXMax()	return the maximal value of XXX
		int getXXXMin()	return the minimum value of XXX
double	JTextField	double getXXXMax()	Return the maximal value of XXX
		double getXXXMin()	return the minimum value of XXX
boolean	JCombobox(only true or false)		
String	JTextField or JCombobox	String[] getXXXList()	If this function exists, it means user need to pick a String from a list and there will be a JCombobox instead of a JTextField.
TreeMap<String, List<Integer>>	JTextField(User can not edit this JTextField. But A	String[] getXXXNames()	To divide "names" into several categories. For example. "Tina Bob

	dialog will appear when user clicks the JTextField)	String[] getXXXClasses()	Tom Helen" is the returned value from "getXXXNames" while "Male Female" is the returned value from "getXXXClasses". When user click the JTextField, a Dialog will show as shown in Figure 3. The value of TreeMap will be " ["Male":[1, 2], "Female": [0,3]]".

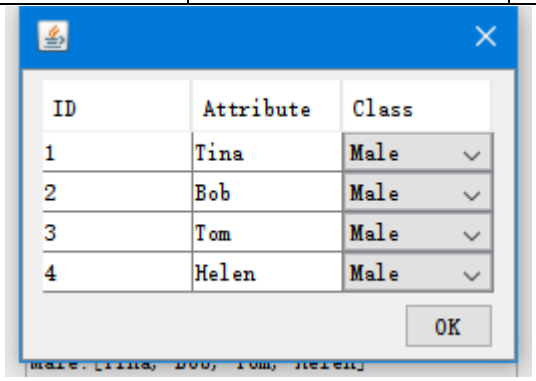

Figure 3

Class *ConfigSample* is a sample.

Some tips about AbstractAlgorithm

1. If constructor of the class throws exceptions, please override the *init()* and put these code in *init()*. *init()* will be called when constructor is called. But if you call constructor by yourself, such as, in *clone()*, please call *init()* explicitly.
2. In *doAlgorithm()*, if you want to make the algorithm stop when user press the "Stop" button, please call *isShouldStop()* to check.
3. *Clone()* will be called when user press the "Start" button.
4. *getCloneBecauseChangeOfFile()* will be called when user have chosen a new file.

How to add new algorithm in MainFrame

1. Find final filed *algorithmNames* in *MainFrame* and add the name of new algorithm.
2. Find function *selectANewFile()* in *MainFrame* and add a new case in switch part.
3. That's all! Enjoy!