Faculty of Engineering and Applied Science

University of Ontario Institute of Technology

SOFE 3980U

Winter 2018

Project Team 14

Software Quality Project

jEdit Bugs

Faculty Supervisor: Dr. Akramul Azim

April 6, 2018

Team Member	Student ID	Signature
Saleh Nawar	100536488	
Fawwaz Khayyat	100568635	
Anwar Almukhtar	100481975	
Christian MacLeod	100560504	

Introduction:

The objective of the project is to improve the quality of the jEdit text editor. jEdit is a mature programmer text editor written in java that contains source code editing, search and replace, file management , and customization. Some of the advantages of jEdit is that it is highly customizable and offers a variety of plugins. However, due to being developed using Java, it can be be demanding in terms of memory usage.

Project Details:

Findbugs which is an open source software for static code analysis in java program that scans for bugs / errors in the code was used .
Findbug reported 660 bugs in different categories , and around 85 bugs hits were addressed and fixed.

Statics of the bugs/errors:

Bad practice	234 bugs
Dodgy code	157 bugs
Malicious code vulnerability	94 bugs
Performance	101 bugs
Internationalization	35 bugs
Multithreaded correctness	25 bugs
Correctness	10 bugs
Experimental	4 bugs

Also, Junit was used in order not to break the relationship between the functions and classes by following these steps:

- Write test cases.
- Test.
- Modify.
- Test again.

Test Cases Design:

Due to the time constraint of the project, the team decided to focus their efforts on developing two test cases for the project. These test cases were designed with the purpose of ensuring that alterations to specific functions do not damage the code. Moreover, due to the nature of the test cases, the team decided not to use any tools to generate test data, instead, specific data were used as input for the the test cases.

Test Suite 1:

This test suite includes 10 test cases. The test cases were made to test the void function `attribute`. This function sets the values of local variables. However, the local variables could not be accessed from the test class because they are private. Moreover, this function is a void function that do not return a value, we added some getters to the class to test that the function is assigning the value to the correct local variable.

The main purpose of the test cases is to make sure the code did not break after the modification. Therefore, the test cases were written first and runned. After the test cases passed, the code got modified. Then, the test cases ran again. Finally, All the test cases passed.

gjt/sp/jedit/pluginmgr/PluginListHandler.java

Bug Details:

Comparison of String parameter using == or !=

Class:

PluginListHandler (org.gjt.sp.jedit.pluginmgr) line 72

Method:

attribute (org.gjt.sp.jedit.pluginmgr.PluginListHandler.attribute(String, String, boolean))

Priority:

High Confidence Bad practice

Figure 1: attribute() bug report

```
60      -           if(aname == "NAME")
    59  +           if(aname.equals("NAME"))
61  60                      name = value;
62      -           else if(aname == "JAR")
    61  +           else if(aname.equals("JAR"))
63  62                      jar = value;
64      -           else if(aname == "VERSION")
    63  +           else if(aname.equals("VERSION"))
65  64                      version = value;
66      -           else if(aname == "DATE")
    65  +           else if(aname.equals("DATE"))
67  66                      date = value;
68      -           else if(aname == "OBSOLETE")
    67  +           else if(aname.equals("OBSOLETE"))
69  68                      obsolete = ("TRUE".equals(value));
70      -           else if(aname == "WHAT")
    69  +           else if(aname.equals("WHAT"))
71  70                      depWhat = value;
72      -           else if(aname == "FROM")
    71  +           else if(aname.equals("FROM"))
73  72                      depFrom = value;
74      -           else if(aname == "TO")
    73  +           else if(aname.equals("TO"))
75  74                      depTo = value;
76      -           else if(aname == "PLUGIN")
    75  +           else if(aname.equals("PLUGIN"))
77  76                      depPlugin = value;
78      -           else if(aname == "SIZE")
    77  +           else if(aname.equals("SIZE"))
79  78              {
80  79                      size = Integer.parseInt(value);
81  80                      if(size == 0)
82  81                              Log.log(Log.WARNING,this,"SIZE = 0");
83  82              }
84  83      } //}}}
85  84
```

Figure 2: attribute() bug fixes

```
PluginListHandlerTest.java ×

public class PluginListHandlerTest {

    private Task task = null;
    private PluginList pluginList;
    PluginListHandler obj = new PluginListHandler(pluginList, path: "");
    @Test
    public void attribute_name() throws Exception {
        obj.attribute( aname: "NAME", value: "Name_value", isSpecified: true);
        assertEquals( expected: "Name_value",obj.getName());
    }
    @Test
    public void attribute_jar() throws Exception {
        obj.attribute( aname: "JAR", value: "JAR_value", isSpecified: true);
        assertEquals( expected: "JAR_value",obj.getJar());
    }
    @Test
    public void attribute_version() throws Exception {
        obj.attribute( aname: "VERSION", value: "VERSION_value", isSpecified: true);
        assertEquals( expected: "VERSION_value",obj.getVersion());
    }
    @Test
```

Figure 3: PluginListHandlerTest class

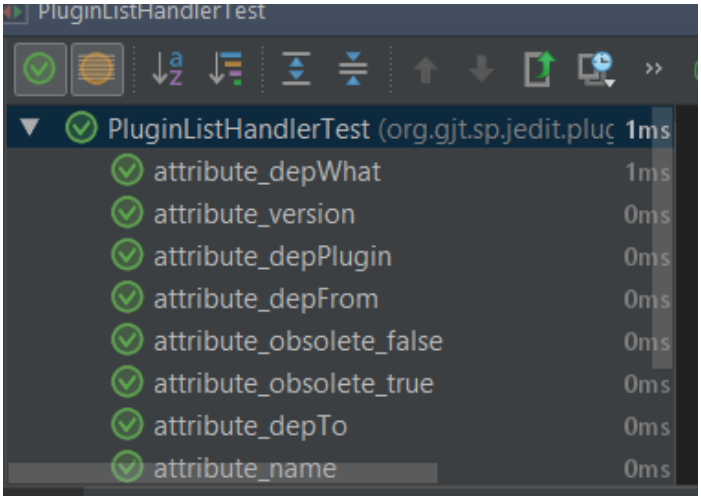


Figure 4: Test case results

Test case 2:

This test case deals with the getRegisters() function of the Registers.java class. Due to the fact the getter function exposed the array registers, a second function, getRegistersCpy(), that returns a copy of the array was made to address this security concern. Afterwards, a test case was made that insures that both functions return identical arrays of the same length.

org/gjt/sp/jedit/Registers.java

Bug Details:

Public static getRegisters() may expose internal representation by returning Registers.registers

Class:

Registers (org.gjt.sp.jedit) line 535

Method:

getRegisters (org.gjt.sp.jedit.Registers.getRegisters())

Priority:

Medium Confidence Dodgy code

Comments:

A public static method returns a reference to an array that is part of the static state of the class. Any code that calls this method can freely modify the underlying array. One fix is to return a copy of the array.

Figure 5: getRegisters bug report

```
public static Register[] getRegisters()
{
    if(!loaded)
        loadRegisters();
    return registers;

} //}}}
public static Register[] getRegistersCpy()
{
    if(!loaded)
        loadRegisters();
    Register[] registerCpy = new Register[registers.length];
    registerCpy= registers.clone();
    return registerCpy;

} //}}}
```

Figure 6: getRegisters and getRegistersCpy

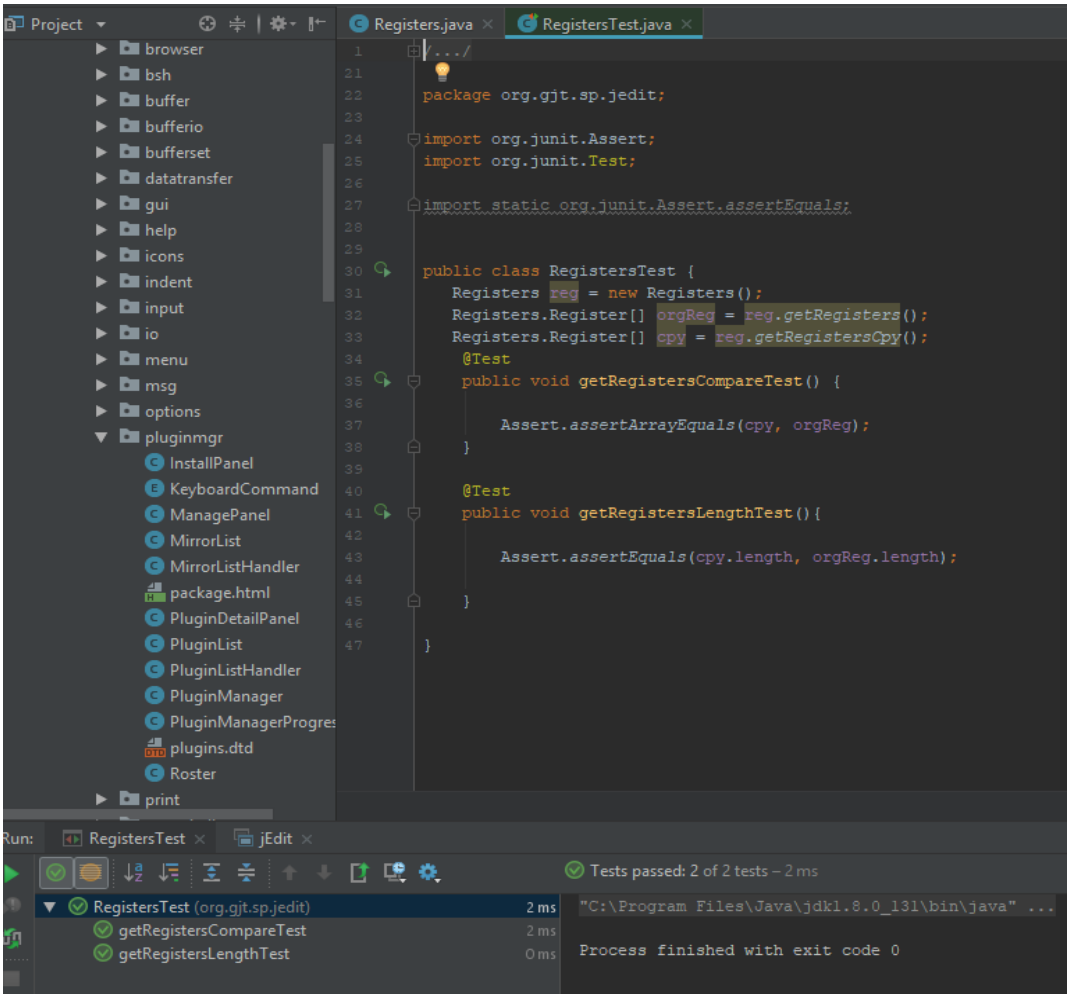


Figure 7: RegistersTest class and test case results

Challenges:

Many of the non-void methods in the application were declared private, making them difficult to test without exposing the internals of the system.
Additionally, the software is already highly stable. Version 4.3p9 launched in early 2007, and it has been receiving regular updates since. The majority of the open issues on their bugtracker are either unreproducible, or marked as corrected in an upcoming release.

Lessons Learned:

- Improve the quality of the code
- Unit tests should be developed before & during the early development of the code, as post release testing leads to a decrease in initial quality, and a reduced ability to obtain full test coverage.

Implementation:
The implementation of fixes to Jedit were made using a variety of tools:

- Findbugs was used for identifying potential bugs.
- intelliJ IDEA was used as IDE to run the project.
- Github was used as source control.

The following section will provide examples of bug fixes made by the team.

gjt/sp/util/StringList.java

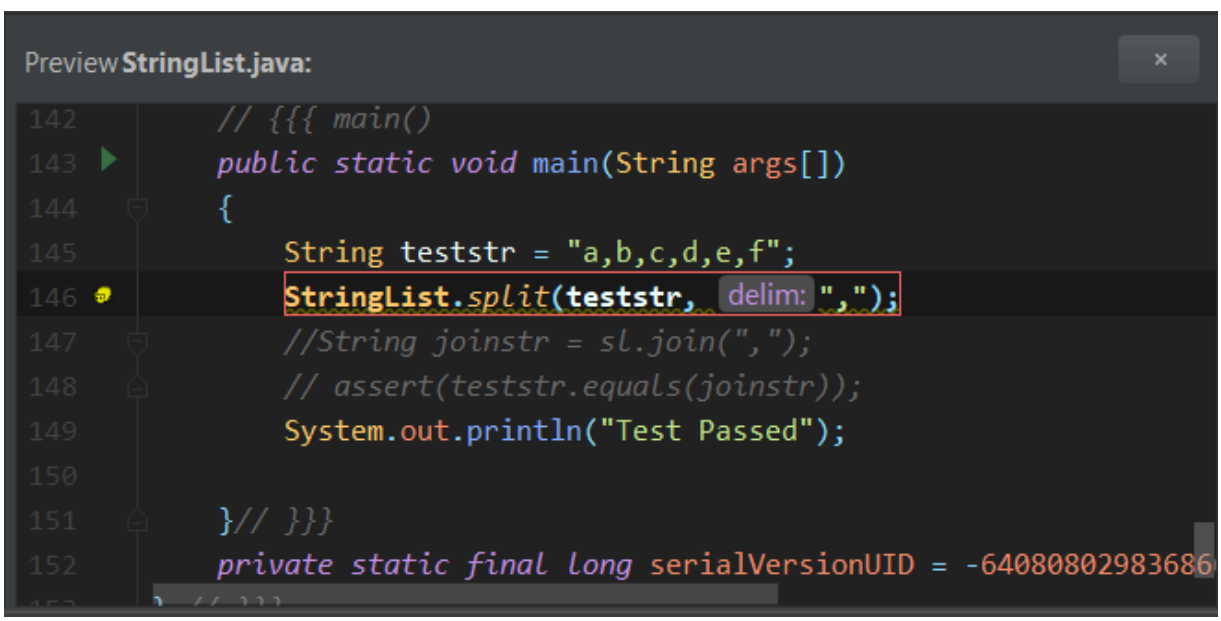
Bug Details:
Return value of split(String, Object) ignored, but method has no side effect

Class:
StringList (org.gjt.sp.util) line 146

Method:
main (org.gjt.sp.util.StringList.main(String[]))

Priority:
Medium Confidence Dodgy code

Comments:
The code in the *main* function is used to test *split* and *join* functions. The code run without error because part of the code is commented. If we uncomment the code, it will produce errors. That is because of missing sl variable storing the return value of the splitted string..



Original Code

```
"C:\Program Files\Java\jdk1.8.0_144\bin\java" ...

Test Passed


Process finished with exit code 0
```

Running Original Code

```
143     public static void main(String args[])
144     {
145         String teststr = "a,b,c,d,e,f";
146         StringList.split(teststr, delim: ",");
147         String joinstr = sl.join(",");
148         assert(teststr.equals(joinstr));
149         System.out.println("Test Passed");
150
151     }// }}}
152     private static final long serialVersionUID = -6408080298368668262L;
153 } // }}}
154
```

StringList › main()

```
Information: java: Errors occurred while compiling module 'jEdit'
Information: javac 1.8.0_144 was used to compile java sources
Information: 3/30/2018 12:02 AM - Compilation completed with 1 error and 0 warnings in 1s 209ms
C:\jEdit\org\gjt\sp\util\StringList.java
Error:(147, 34) java: cannot find symbol
                symbol:   variable sl
                location: class org.gjt.sp.util.StringList
```

Running Uncommented Code

```
144     public static void main(String args[])
145     {
146         String teststr = "a,b,c,d,e,f";
147         StringList sl = StringList.split(teststr, delim: ",");
148         String joinstr = sl.join( delim: "*");
149         assert(teststr.equals(joinstr));
150         System.out.println("Test Passed");
151
152     }// }}}
153     private static final long serialVersionUID = -6408080298368668262L;
154 } // }}}
155
```

```
"C:\Program Files\Java\jdk1.8.0_144\bin\java" ...

Test Passed


Process finished with exit code 0
```

Running Fixed Code

gjt/sp/jedit/pluginmgr/PluginListHandler.java

Bug Details:
Comparison of String parameter using == or !=

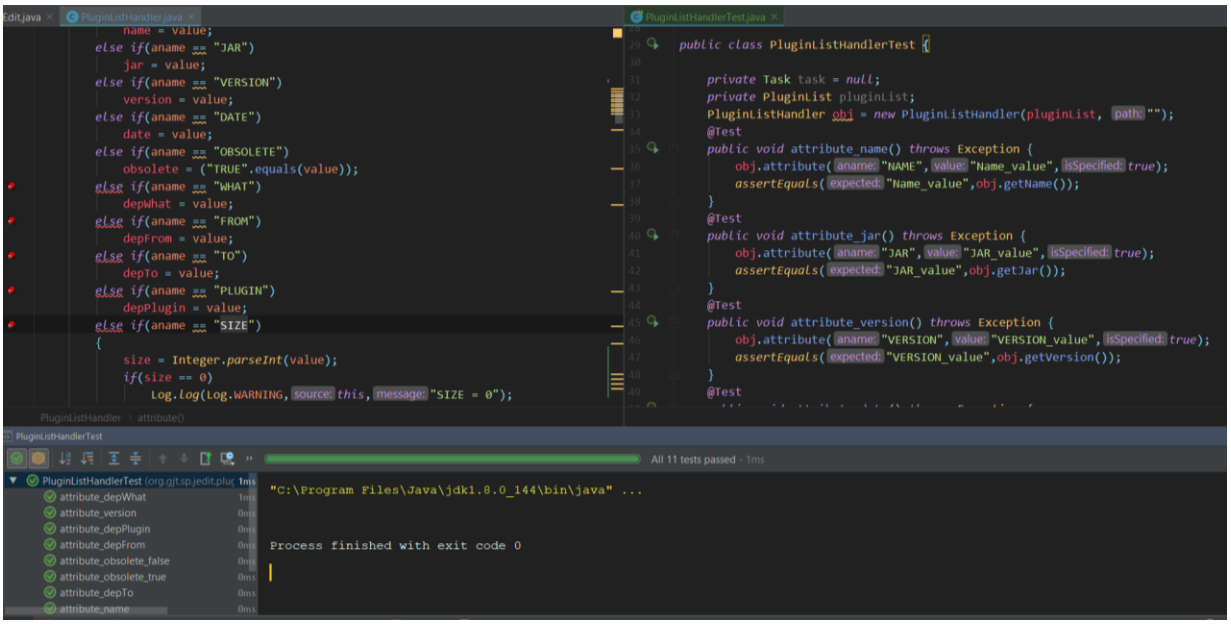
Class:
PluginListHandler (org.gjt.sp.jedit.pluginmgr) line 72

Method:
attribute (org.gjt.sp.jedit.pluginmgr.PluginListHandler.attribute(String, String, boolean))

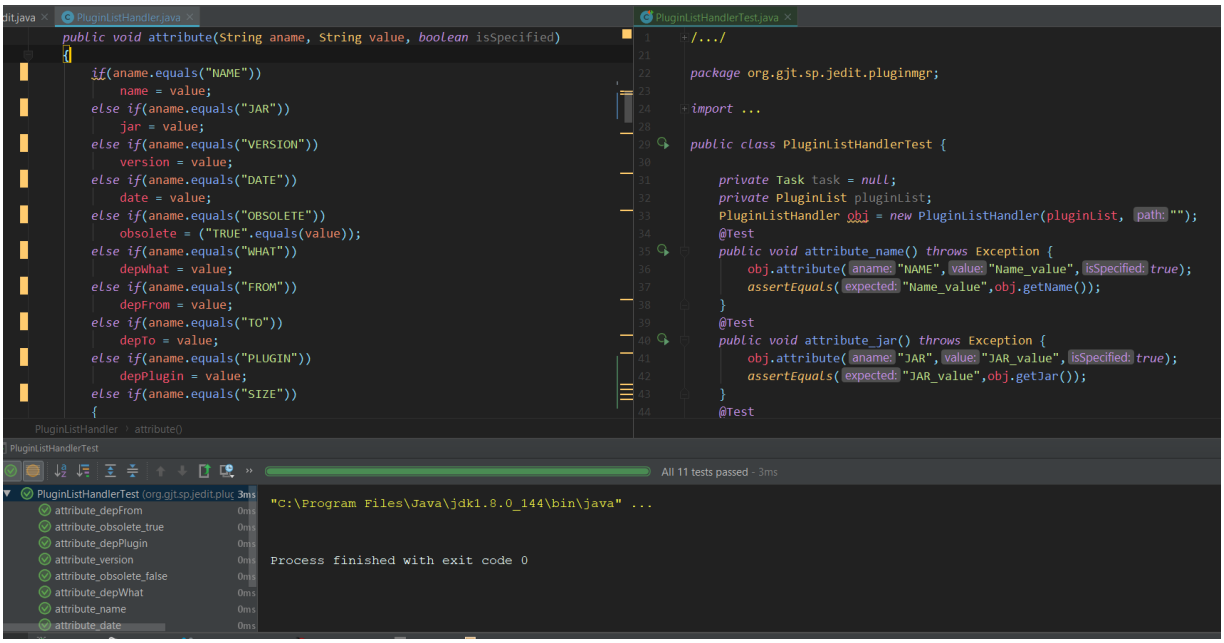
Priority:
High Confidence Bad practice

```java
//{{{ attribute() method
public void attribute(String aname, String value, boolean isSpecified)
{
    if(aname == "NAME")
        name = value;
    else if(aname == "JAR")
        jar = value;
    else if(aname == "VERSION")
        version = value;
    else if(aname == "DATE")
        date = value;
    else if(aname == "OBSOLETE")
        obsolete = ("TRUE".equals(value));
    else if(aname == "WHAT")
        depWhat = value;
    else if(aname == "FROM")
        depFrom = value;
    else if(aname == "TO")
        depTo = value;
    else if(aname == "PLUGIN")
        depPlugin = value;
    else if(aname == "SIZE")
    {
        size = Integer.parseInt(value);
        if(size == 0)
            Log.log(Log.WARNING, source: this, message: "SIZE = 0");
    }
} //}}}
```

Original Code



Test Cases Passed Before Code Modification



Test Cases Passed After Modifying the Code

gjt/sp/jedit/browser/VFSDirectoryEntryTableModel.java

Bug Details:
VFSDirectoryEntryTableModel$EntryCompare implements Comparator but not Serializable

Class:
VFSDirectoryEntryTableModel$EntryCompare (org.gjt.sp.jedit.browser) lines 412-482

Method:
attribute (org.gjt.sp.jedit.pluginmgr.PluginListHandler.attribute(String, String, boolean))

Priority:
Medium Confidence Bad practice

Problem classification:
Bad practice (Incorrect definition of Serializable class)
SE_COMPARATOR_SHOULD_BE_SERIALIZABLE (Comparator doesn't implement Serializable)

Notes:
In class org.gjt.sp.jedit.browser.VFSDirectoryEntryTableModel$EntryCompare ComparatorIdiom (Se)

Preview VFSDirectoryEntryTableModel.java:

```
407         * Implementation of {@link Comparator}
408         * interface that compares {@link VFSDirectoryEntryTableModel.Entry} instances.
409         * For sorting columns in the VFS Browser.
410         * @since jEdit 4.3pre7
411         */
412         static class EntryCompare implements Comparator<Entry>
413         {
414             private boolean sortIgnoreCase, sortMixFilesAndDirs, sortAscending;
415             private String sortAttribute;
416             /**
417              * Creates a new <code>EntryCompare</code>
418              * Expanded branches are sorted, too, but keep with their parent entries
419              * @param sortBy The extended attribute by which to sort the entries.
420              * @param ascending If false, sort order is reversed.
421              */
```

gjt/sp/jedit/bufferset/BufferSet.java

Bug Details:
BufferSet$NameSorter implements Comparator but not Serializable

Class:
BufferSet$NameSorter (org.gjt.sp.jedit.bufferset) lines 364-374

Priority:
Medium Confidence Bad practice

Problem classification:
Bad practice (Incorrect definition of Serializable class)
SE_COMPARATOR_SHOULD_BE_SERIALIZABLE (Comparator doesn't implement Serializable)

Notes:
In class org.gjt.sp.jedit.bufferset.BufferSet$NameSorter
ComparatorIdiom (Se)

Preview **BufferSet.java:**

```
359     private static final Comparator<Buffer> pathSorter = new PathSorter();
360     private Comparator<Buffer> sorter;
361     //}}}
362
363     //{{{ NameSorter class
364     public static class NameSorter implements Comparator<Buffer>
365     {
366         public int compare(Buffer o1, Buffer o2)
367         {
368
369             int ret = StandardUtilities.compareStrings(o1.getName(), o2.getName(),  igno
370             if (ret == 0)
371             {
372                 ret = StandardUtilities.compareStrings(o1.getPath(), o2.getPath(),  igno
373             }
```

gjt/sp/jedit/print/PrinterDialog.java

Bug Details:
PrinterDialog.ALL isn't final but should be

Class:
PrinterDialog (org.gjt.sp.jedit.print) line 78

Field:
ALL

Priority:
High Confidence Malicious code vulnerability

Problem classification:
Malicious code vulnerability (Mutable static field)
MS_SHOULD_BE_FINAL (Field isn't final but should be)

Notes:
Field org.gjt.sp.jedit.print.PrinterDialog.ALL
MutableStaticFields (MS)

Preview **PrinterDialog.java:**

```java
    private JComboBox<Finishings> finishing;
    private JComboBox<Sides> sides;
    private JComboBox<NumberUp> pagesPerSide;
    private JComboBox<PresentationDirection> pageOrdering;
    private JComboBox<MediaTray> paperSource;
    private JComboBox<OrientationRequested> orientation;
    private boolean pageSetupOnly;
    private boolean canceled = false;
    private Map<String, String> messageMap;
    private PageSetupPanel pageSetupPanel;
    public static int ALL = 0;
    public static int ODD = 1;
    public static int EVEN = 2;
    public static int RANGE = 3;
    public static int CURRENT_PAGE = 4;
    public static int SELECTION = 5;
    public static int onlyPrintPages = ALL;
    private int printRangeType = ALL;
    private DocFlavor DOC_FLAVOR = DocFlavor.SERVICE_FORMATTED.PRINT
    private boolean reversePrinting = false;
```

gjt/sp/jedit/jEdit.java

Bug Details:
initUserProperties() may fail to clean up java.io.InputStream

Class:
jEdit (org.gjt.sp.jedit) line 3787

Method:
initUserProperties (org.gjt.sp.jedit.jEdit.initUserProperties())

Priority:
Medium Confidence Experimental

Problem classification:
Experimental (Unsatisfied obligation to clean up stream or resource)
OBL_UNSATISFIED_OBLIGATION (Method may fail to clean up stream or resource)

Notes:
Obligation to clean up resource created at jEdit.java:[line 3787] is not discharged
Reference type java.io.InputStream
1 instances of obligation remaining
Path continues at jEdit.java:[line 3797]
Path continues at jEdit.java:[line 3799]
Remaining obligations: {InputStream x 1}
FindUnsatisfiedObligation (OBL)

Preview **jEdit.java:**

```
3783            propsModTime = file.lastModified();
3784
3785            try
3786            {
3787                propMgr.loadUserProps(
3788                    new FileInputStream(file));
3789            }
3790            catch(FileNotFoundException fnf)
3791            {
3792                //Log.log(Log.DEBUG,jEdit.class,fnf);
3793            }
3794            catch(Exception e)
```

gjt/sp/jedit/View.java

Bug Details:
Dead store to showToolbars

Class:
View (org.gjt.sp.jedit) line 1386

Method:
updateFullScreenProps (org.gjt.sp.jedit.View.updateFullScreenProps())

Preview **View.java:**

```
1382    {
1383        boolean alternateLayout = jEdit.getBooleanProperty(
1384            name: "view.toolbar.alternateLayout");
1385        boolean showMenu = jEdit.getBooleanProperty( name: "fullSc
1386        boolean showToolbars = jEdit.getBooleanProperty( name: "fu
1387        boolean showStatus = jEdit.getBooleanProperty( name: "full
1388        if (! showMenu)
1389        {
1390            menuBar = getJMenuBar();
1391            setJMenuBar(null);
1392        }
1393        else if (menuBar != null)
```

gjt/sp/jedit/SplitConfigParser.java

Bug Details:
Should SplitConfigParser$BufferSet be a _static_ inner class?

Class:
SplitConfigParser$BufferSet (org.gjt.sp.jedit) lines 204-288

Priority:
Medium Confidence Performance

Problem classification:
Performance (Inner class could be made static)
SIC_INNER_SHOULD_BE_STATIC (Should be a static inner class)

Notes:
In class org.gjt.sp.jedit.SplitConfigParser$BufferSet
UnreadFields (NP|SIC|SS|ST|UrF|UuF|UwF)

Preview **SplitConfigParser.java:**

```
200         //{{{ BufferSet
201         // Represents a set of file names for buffers.
202         private class BufferSet
203         {
204                 List<String> buffers = new ArrayList<String>();
205                 String scope = null;
206
207                 boolean includeFiles = true;
208                 boolean includeRemotes = false;
209
210                 public BufferSet() {}
211
```

gjt/sp/jedit/EditPane.java

Bug Details:
Switch statement found where default case is missing

Class:
EditPane$StatusHandler (org.gjt.sp.jedit) lines 1194-1207

Method:
statusChanged (org.gjt.sp.jedit.EditPane$StatusHandler.statusChanged(TextArea, int, boolean))

Priority:
Medium Confidence Dodgy code

Problem classification:
Dodgy code (Switch case falls through)
SF_SWITCH_NO_DEFAULT (Switch statement found where default case is missing)
SwitchFallthrough (SF)

Preview EditPane.java:

```
1190            StatusBar status = view.getStatus();
1191            if(status == null)
1192                return;
1193
1194            switch(flag)
1195            {
1196            case OVERWRITE_CHANGED:
1197                status.setMessageAndClear(
1198                    jEdit.getProperty( name: "view.status.overwrite-changed",
1199                    new Integer[] { value ? 1 : 0 }));
1200                break;
1201            case MULTI_SELECT_CHANGED:
1202                status.setMessageAndClear(
1203                    jEdit.getProperty( name: "view.status.multi-changed",
1204                    new Integer[] { value ? 1 : 0 }));
1205                break;
1206            case RECT_SELECT_CHANGED:
1207                status.setMessageAndClear(
1208                    jEdit.getProperty( name: "view.status.rect-select-changed",
1209                    new Integer[] { value ? 1 : 0 }));
1210                break;
1211            }
1212
```

org/gjt/sp/jedit/Registers.java

Bug Details:
Public static getRegisters() may expose internal representation by returning Registers.registers

Class:
Registers (org.gjt.sp.jedit) line 535

Method:
getRegisters (org.gjt.sp.jedit.Registers.getRegisters())

Priority:
Medium Confidence Dodgy code

Comments:
A public static method returns a reference to an array that is part of the static state of the class. Any code that calls this method can freely modify the underlying array. One fix is to return a copy of the array.

Preview Registers.java:

```
516                 register.setTransferable(new StringSelection( s, ""));
517             else
518             {
519                 registers[name] = null;
520                 modified = true;
521                 if (listener != null)
522                     listener.registerChanged(name);
523             }
524         } //}}}
525
526         //{{{ getRegisters() method
527         /**
528          * Returns an array of all available registers. Some of the elements
529          * of this array might be <code>null</code>.
530          */
531         public static Register[] getRegisters()
532         {
533             if(!loaded)
534                 loadRegisters();
535             return registers;
536
537         } //}}}
```

Bug:	Call to static DateFormat (Sun Bug #6231579 and Sun Bug #6178997)

Locations:	org.gjt.sp.jedit.io.FileVFS$LocalFile.getExtendedAttribute(String) org.gjt.sp.util.Log._log(int, String, String)
Classification:	High Confidence Multithreaded Correctness

Original Code:	`fetchAttrs();`
	`if (name.equals(EA_MODIFIED))`
	`{`
	`  return DATE_FORMAT.format(new Date(modified));`
	`}`
	`else`
	`{`
	`  return super.getExtendedAttribute(name);`
	`}`
	`String fullMessage = timeFormat.format(`
	`  new Date()) + " ["+Thread.currentThread().getName()+`
	`  "] [" + urgencyToString(urgency) + "] " + source`
	`  + ": " + message;`

New Code:	`String fullMessage;`
	
	`synchronized (timeFormat) {`
	`  fullMessage = timeFormat.format(new Date()) + " [" + Thread.currentThread().getName() + "] [" +`
	`  urgencyToString(urgency) + "] " + source + ": " + message;`
	
	`synchronized ((DATE_FORMAT)){`
	`  return DATE_FORMAT.format(new Date(modified));`
	`}`

Fix Description:	Encapsulated the DateFormat objects in java's Synchronized() block; which ensures only one thread can access the object at a time

FindBugs Results

Before fixes	After fixes