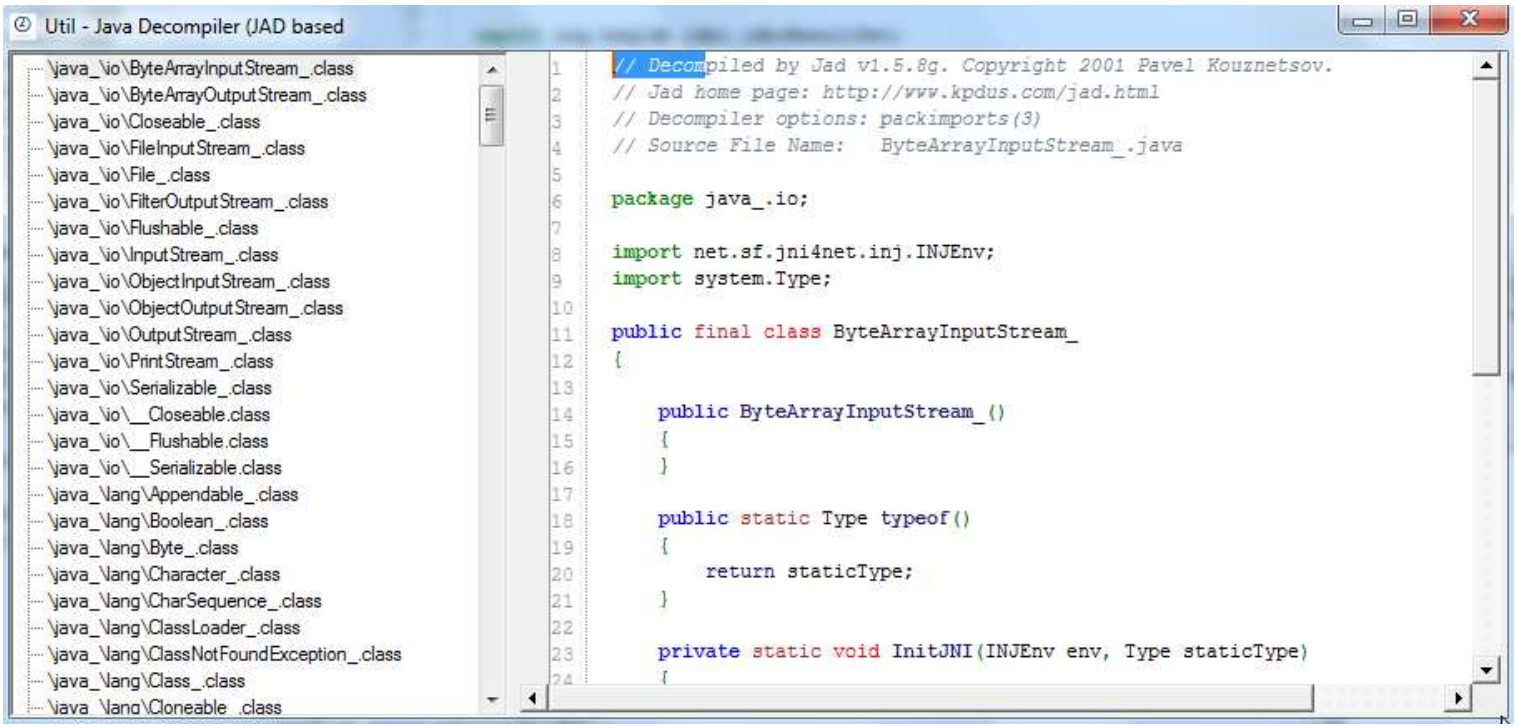
Creating a Java Decompiler (using Jad)

based on the version of Jad from http://www.varaneckas.com/jad/ , namely
http://www.varaneckas.com/jad/jad158g.win.zip

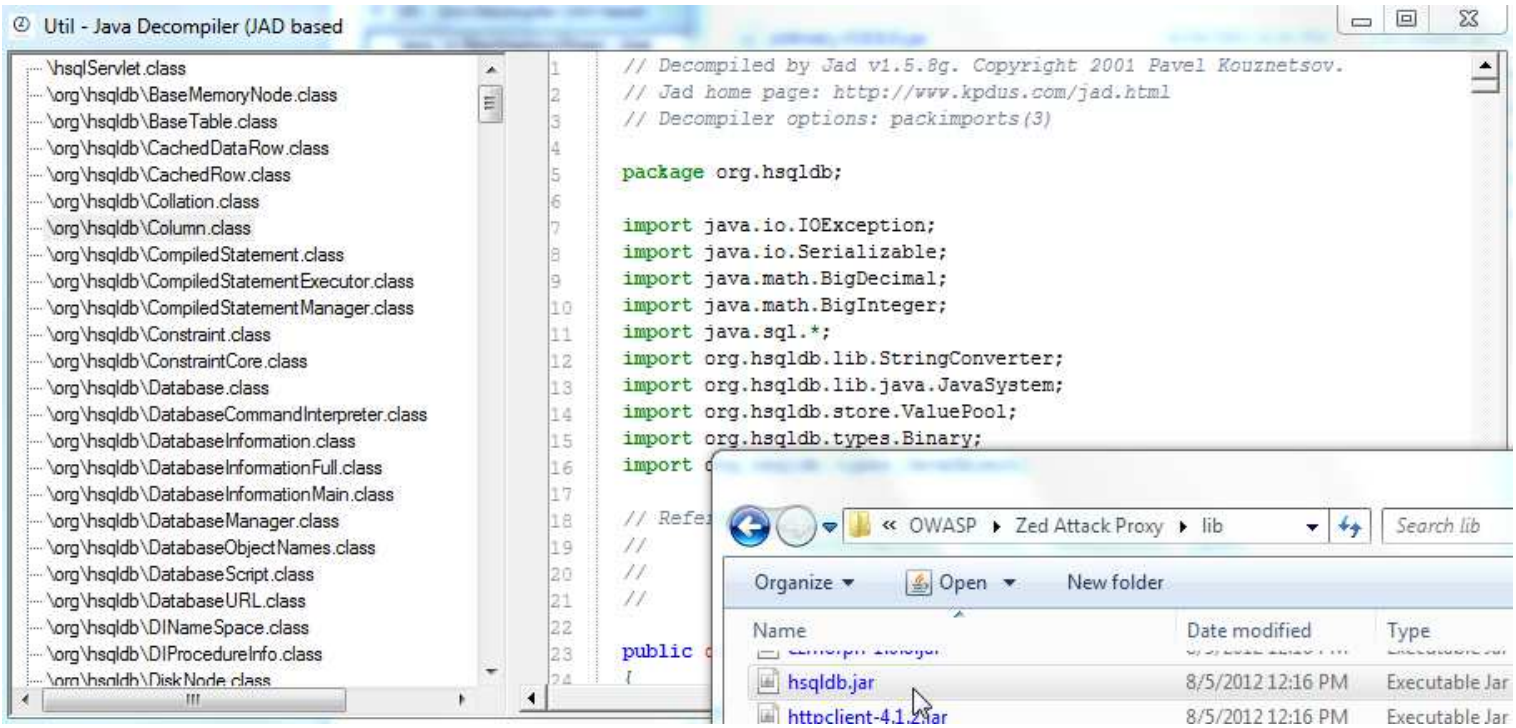
First working GUI

```
//var topPanel = panel.add_Panel(true);
var topPanel = "Util - Java Decompiler (JAD based".popupWindow(900,400);
var files = topPanel.insert_Left(300).add_TreeView();
var codeViewer = topPanel.add_SourceCodeViewer();

var targetFolder = "_JadDecompilations".tempDir(false);

Action<string> extractClassesFromJar =
       (jarFile)=>  {
                                "Extracting Classes from Jar: {0}".info(jarFile);
                                if (jarFile.extension(".jar"))
                                {
                                       files.azure();
                                       var extractFolder = targetFolder.pathCombine(jarFile.fileName
().safeFileName()).createDir();
                                       if (extractFolder.folderExists().isFalse() ||
extractFolder.files(true).empty())
                                              jarFile.unzip(extractFolder);
                                       var classFiles =extractFolder.files(true,"*.class");
                                       "found {0} *.class files in folder: {1}".info(classFiles.size
(),extractFolder);
                                       files.beginUpdate()
                                                    .clear()
                                                    .add_Nodes(classFiles, (file)=> file.remove
(extractFolder))
                                              .endUpdate();
                                       files.white();
                                       files.selectFirst();
                                }
                           };

files.afterSelect<string>(
       (file)=>{
                           var jad = @"C:\__Tests\_J2EE\jad.exe";
                           var parameters = "-p \"{0}\"".format(file);
                           var javaCode = jad.startProcess_getConsoleOut(parameters);

                           codeViewer.set_Text(javaCode,".java");
                     });
files.onDrop(extractClassesFromJar);
extractClassesFromJar(@"C:\__Tests\JNI_4_net\jni4net-0.8.6.0-bin\lib\jni4net.j-0.8.6.0.jar");
```

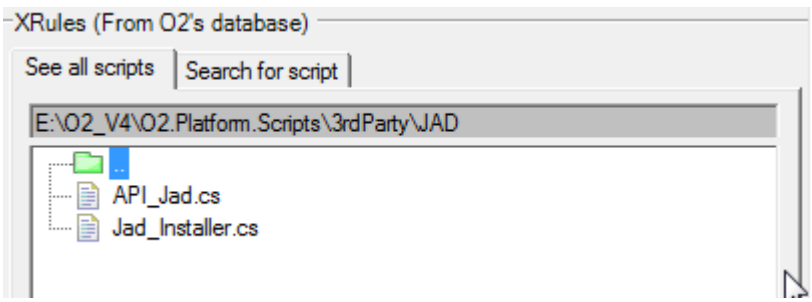
which looks like this:



and supports drag n'drop of jar files:



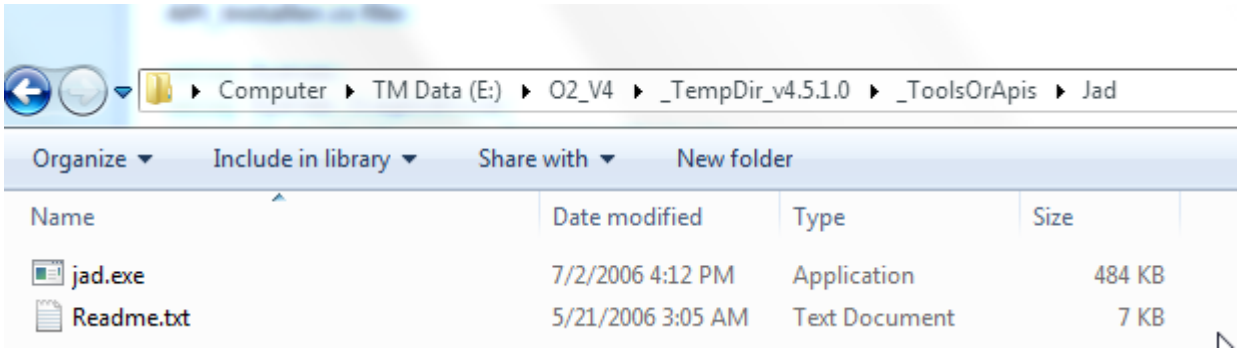
Next step is to refactor the code into code behind C# API files



API_Installer.cs file:

```
using System;
using System.Diagnostics;
using O2.DotNetWrappers.ExtensionMethods;
//O2File:Tool_API.cs

namespace O2.XRules.Database.APIs
{
	public class testInstall
	{
		public static void test()
		{
			new Jad_Install().start();
		}
	}

	public class Jad_Install : Tool_API
	{
		public Jad_Install()
		{
			config("Jad",
				"http://www.varaneckas.com/jad/jad158g.win.zip".uri(),
				@"jad.exe");
			installFromZip_Web();
		}
		//

		public Process start()
		{
			if (this.isInstalled())
				return this.Executable.startProcess();
			return null;
		}
	}
}
```

The API_Installer.cs when executed will download the Jad zip and unzip it into the ToolsOrApis folder:



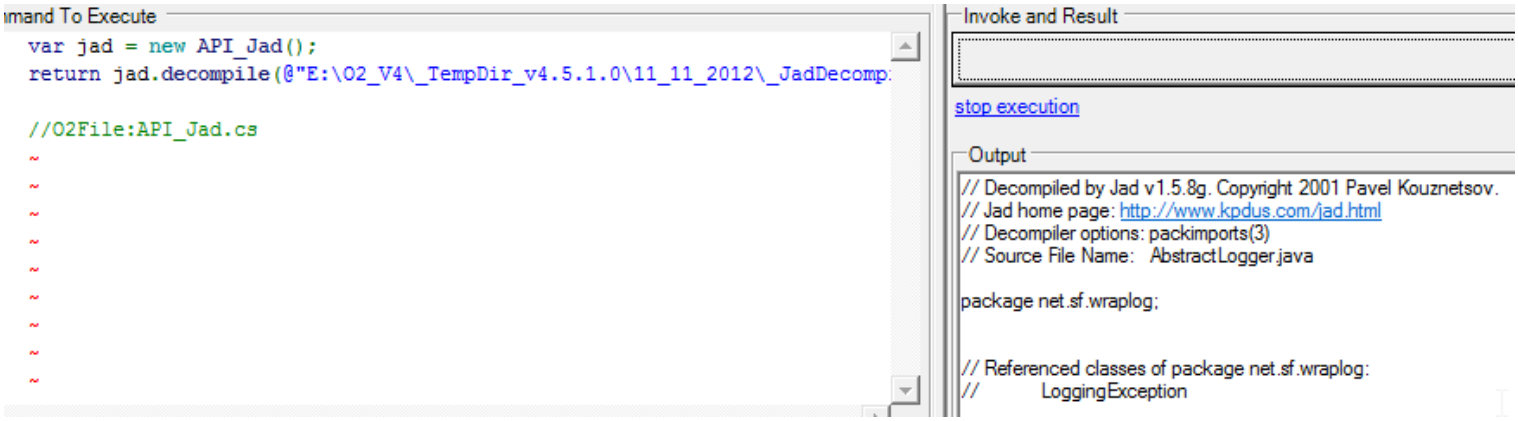
API_Jad.cs file:

```
using System;
using O2.Kernel;
using O2.DotNetWrappers.ExtensionMethods;
//Installer:Jad_Installer.cs!jad/jad.exe


namespace O2.XRules.Database.APIs
{
    public class API_Jad
    {
       public string Executable { get; set;}

       public API_Jad()
       {
            this.Executable = PublicDI.config.ToolsOrApis.pathCombine(@"jad/jad.exe");
       }
       public string execute(string commands)
       {
```

```
            return this.Executable.startProcess_getConsoleOut(commands);
        }
    }

    public static class API_Jad_ExtensionMethods
    {
        public static string help(this API_Jad jad)
        {
            return jad.execute("");
        }

        public static string decompile(this API_Jad jad, string classFile)
        {
            return       jad.execute("-p \"{0}\"".format(classFile));
        }
    }
}
```

The API_Jad.cs can be used like this:

```
var jad = new API_Jad();
return jad.decompile(@"E:\O2_V4\_TempDir_v4.5.1.0\11_11_2012\_JadDecompilations\BrowserLauncher2_1_3.jar
\net\sf\wraplog\AbstractLogger.class");

//O2File:API_Jad.cs
```



Going back to the Decompiler tools (shown in the beggining), we can refactor it using the API_Jad class:

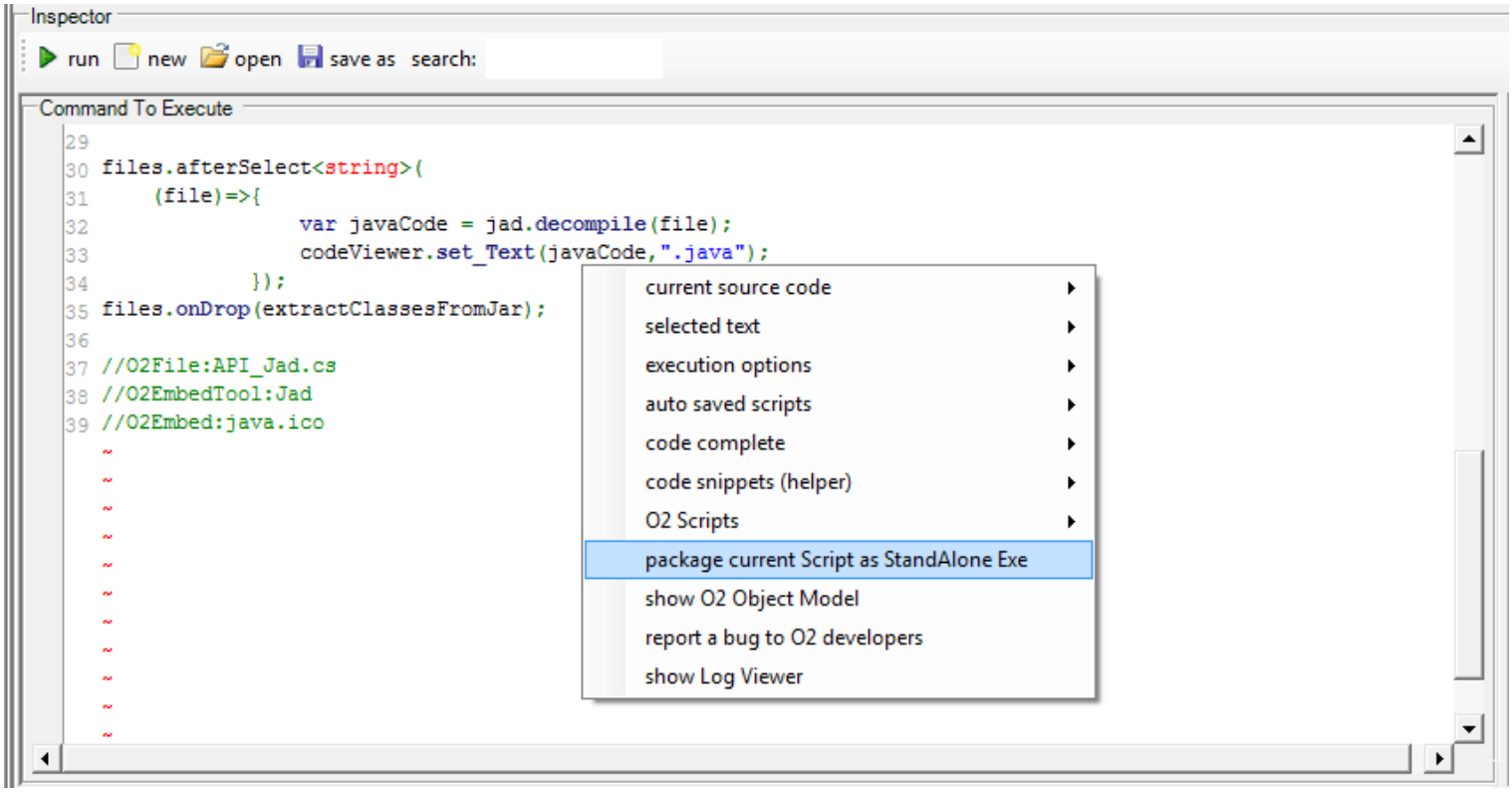
```
var jad = new API_Jad();

files.afterSelect<string>(
       (file)=>{
                        var javaCode = jad.decompile(file);
                        codeViewer.set_Text(javaCode,".java");
                    });
files.onDrop(extractClassesFromJar);



//O2File:API_Jad.cs
```

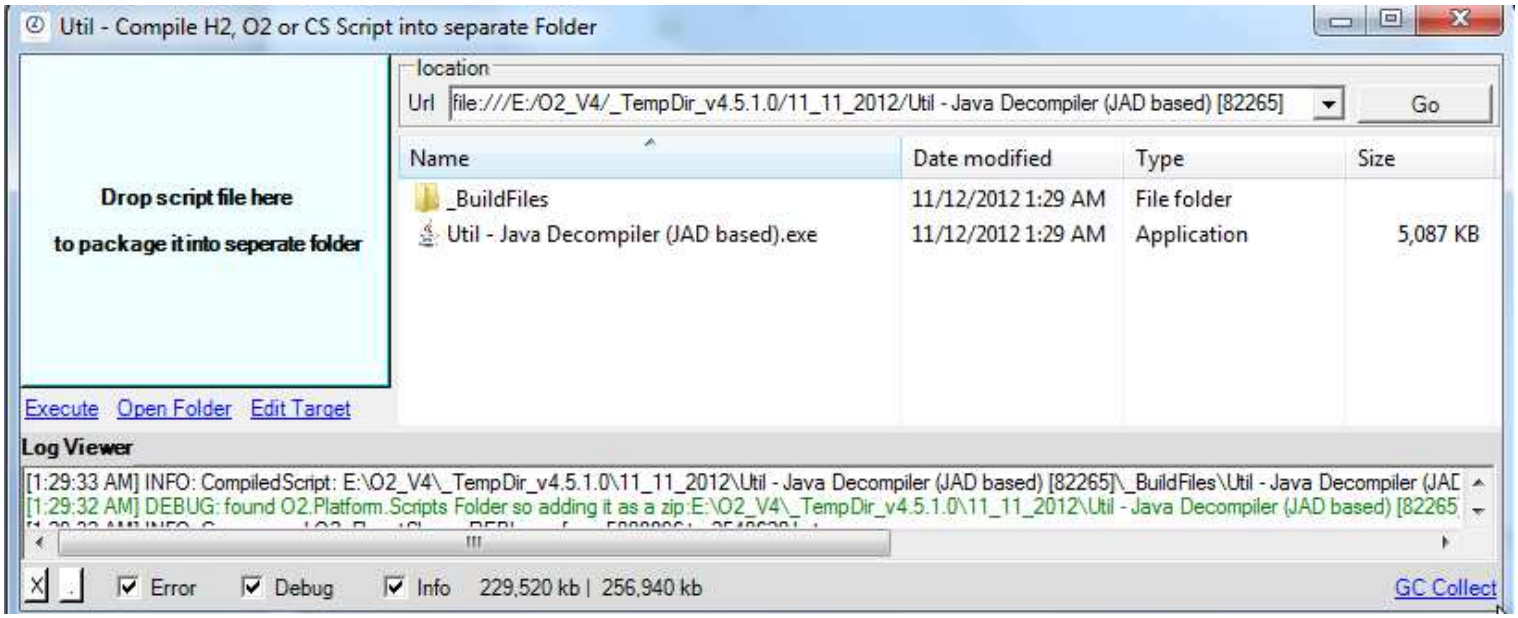
Here is the final script (with a couple extra settings to allow the creation of the stand alone tool)

```
O2Setup.extractEmbededConfigZips();
//var topPanel = panel.add_Panel(true);
var topPanel = "Util - Java Decompiler (JAD based)".popupWindow(900,400);
var files = topPanel.insert_Left(300,"Drop *.jar files here (to decompile them)").add_TreeView();
var codeViewer = topPanel.title("Decompiled selected *. Class file").add_SourceCodeViewer();

var targetFolder = "_JadDecompilations".tempDir(false);
```

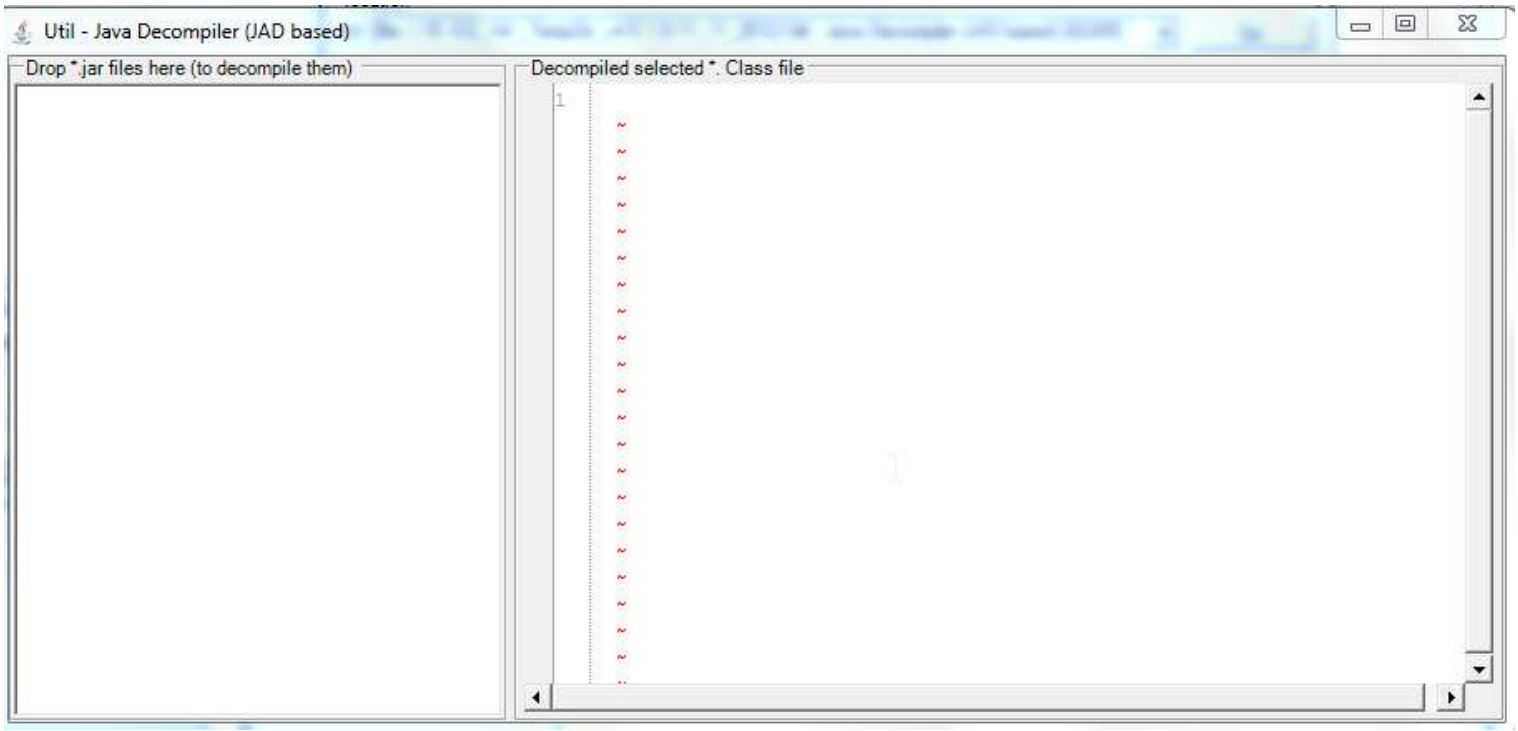
```
var jad = new API_Jad();

Action<string> extractClassesFromJar =
       (jarFile)=>   {
                                   "Extracting Classes from Jar: {0}".info(jarFile);
                                   if (jarFile.extension(".jar"))
                                   {
                                          files.azure();
                                          var extractFolder = targetFolder.pathCombine(jarFile.fileName
().safeFileName()).createDir();
                                          if (extractFolder.folderExists().isFalse() ||
extractFolder.files(true).empty())
                                                 jarFile.unzip(extractFolder);
                                          var classFiles =extractFolder.files(true,"*.class");
                                          "found {0} *.class files in folder: {1}".info(classFiles.size
(),extractFolder);
                                          var jarNode = files.add_Node(jarFile.fileName()).color
(Color.DarkRed);
                                          files.collapse();
                                          jarNode.add_Nodes(classFiles, (file)=> file.remove
(extractFolder));
                                          jarNode.selected().expand();
                                          jarNode.nodes().first().selected();
                                          files.white();
                                   }
                            };

files.afterSelect<string>(
       (file)=>{
                            var javaCode = jad.decompile(file);
                            codeViewer.set_Text(javaCode,".java");
                     });
files.onDrop(extractClassesFromJar);


//O2File:API_Jad.cs
//O2EmbedTool:Jad
//O2Embed:java.ico
```

The last step is to create the stand alone executable:

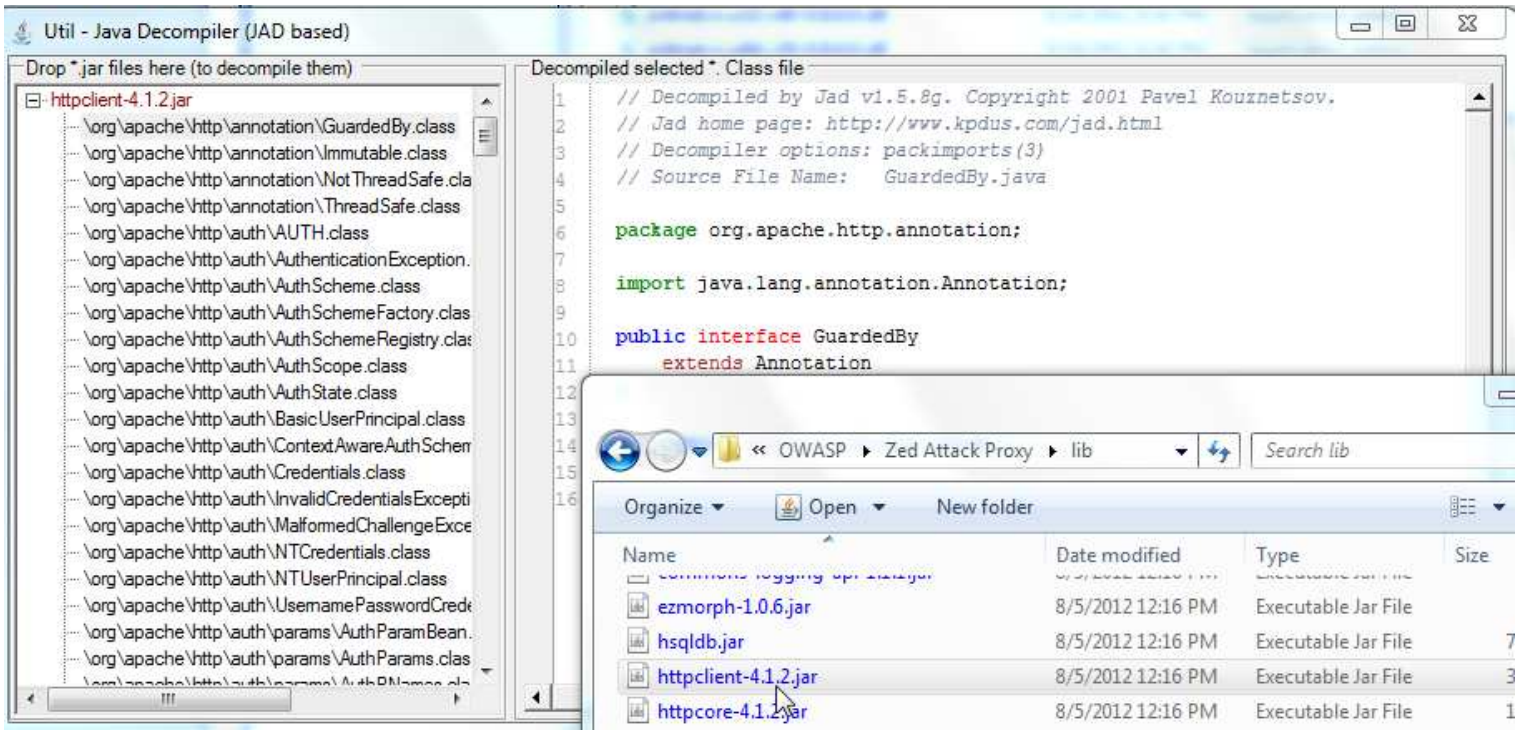




which when executed looks like this (by default):



and like this after dropping an Jar:


Util - Java Decompiler (JAD based)
Drop *.jar files here (to decompile them)
httpclient-4.1.2.jar
\org\apache\http\annotation\GuardedBy.class
\org\apache\http\annotation\Immutable.class
\org\apache\http\annotation\NotThreadSafe.cla
\org\apache\http\annotation\ThreadSafe.class
\org\apache\http\auth\AUTH.class
\org\apache\http\auth\AuthenticationException.
\org\apache\http\auth\AuthScheme.class
\org\apache\http\auth\AuthSchemeFactory.clas
\org\apache\http\auth\AuthSchemeRegistry.clas
\org\apache\http\auth\AuthScope.class
\org\apache\http\auth\AuthState.class
\org\apache\http\auth\BasicUserPrincipal.class
\org\apache\http\auth\ContextAwareAuthSchem
\org\apache\http\auth\Credentials.class
\org\apache\http\auth\InvalidCredentialsExcepti
\org\apache\http\auth\MalformedChallengeExce
\org\apache\http\auth\NTCredentials.class
\org\apache\http\auth\NTUserPrincipal.class
\org\apache\http\auth\UsernamePasswordCrede
\org\apache\http\auth\params\AuthParamBean.
\org\apache\http\auth\params\AuthParams.clas
Decompiled selected *.Class file
1 // Decompiled by Jad v1.5.8g. Copyright 2001 Pavel Kouznetsov.
2 // Jad home page: http://www.kpdus.com/jad.html
3 // Decompiler options: packimports(3)
4 // Source File Name: GuardedBy.java
5
6 package org.apache.http.annotation;
7
8 import java.lang.annotation.Annotation;
9
10 public interface GuardedBy
11 extends Annotation
12
13
14
15
16
« OWASP ▸ Zed Attack Proxy ▸ lib
Search lib
Organize
Open
New folder
Name
Date modified
Type
Size
ezmorph-1.0.6.jar
8/5/2012 12:16 PM
Executable Jar File
hsqldb.jar
8/5/2012 12:16 PM
Executable Jar File
httpclient-4.1.2.jar
8/5/2012 12:16 PM
Executable Jar File
httpcore-4.1.2.jar
8/5/2012 12:16 PM
Executable Jar File