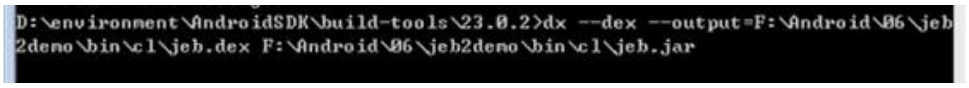
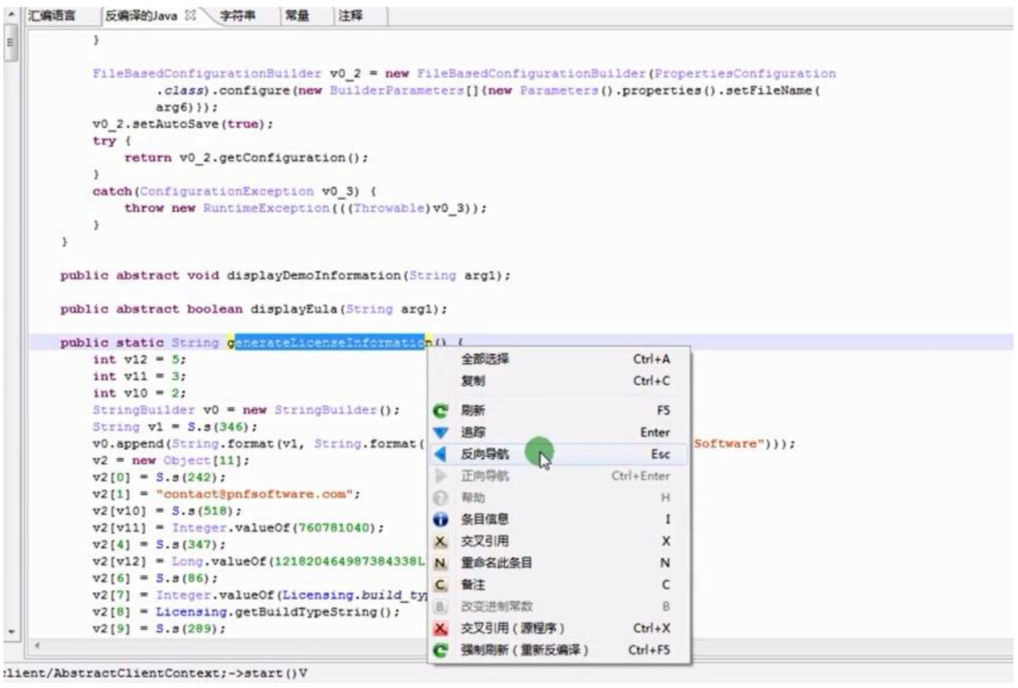
1. Android dx 工具:把 jar 代码转换为 dex 代码的工具
   可以手动转换出 dex 文件，然后利用 jeb 进行分析
   Dx --dex --output=输出 dex 路径 jar 文件路径



```
D:\environment\AndroidSDK\build-tools\23.0.2>dx --dex --output=F:\Android\06\jeb
2demo\bin\cl\jeb.dex F:\Android\06\jeb2demo\bin\cl\jeb.jar
```

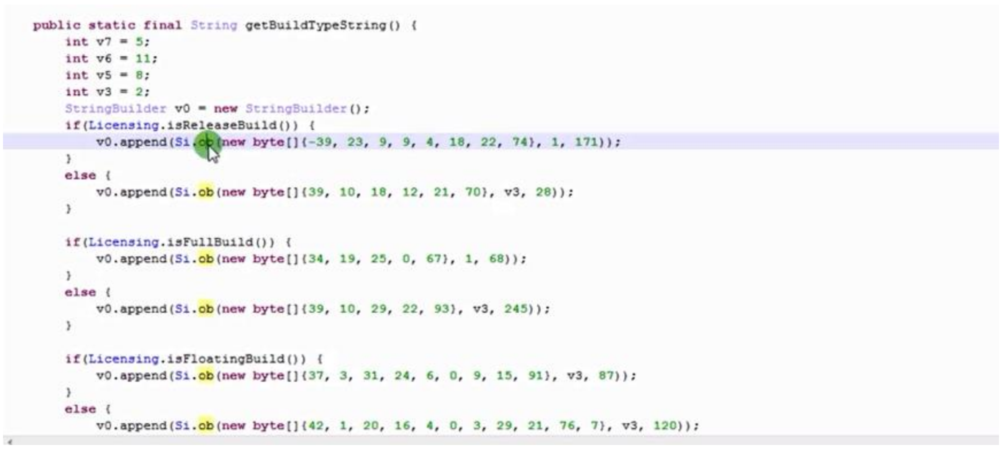
2. Jeb 的基本操作
   (1)重命名(n)
   (2)跟踪(Enter, 双击)
   (3)返回(Esc)
   (4)前进(Ctrl + Enter)
   (5)帮助(H)
   (6)条目信息(I)
   (7)交叉引用(X)，源码交叉引用(Ctrl + X)
   (8)注释(; or C)
   (9)改变进制数(B)
   (10)反编译(Tab)



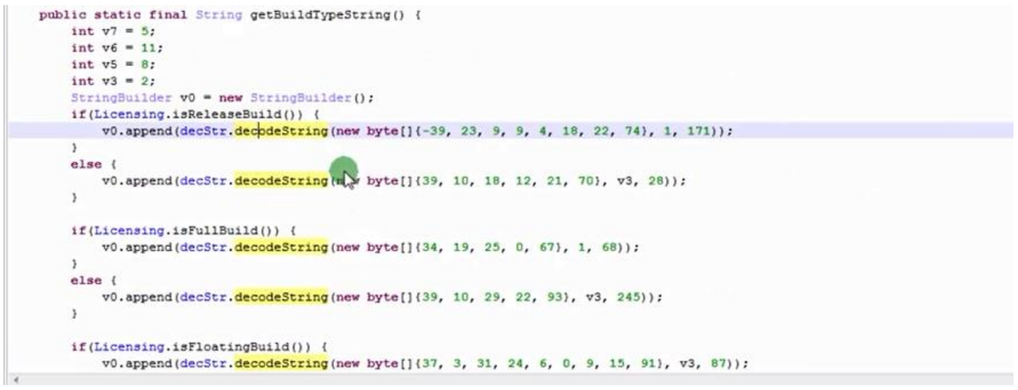
```
        }

        FileBasedConfigurationBuilder v0_2 = new FileBasedConfigurationBuilder(PropertiesConfiguration
                .class).configure(new BuilderParameters[]{new Parameters().properties().setFileName(
                arg6)});
        v0_2.setAutoSave(true);
        try {
            return v0_2.getConfiguration();
        }
        catch(ConfigurationException v0_3) {
            throw new RuntimeException(((Throwable)v0_3));
        }
    }

    public abstract void displayDemoInformation(String arg1);

    public abstract boolean displayEula(String arg1);

    public static String generateLicenseInformation() {
        int v12 = 5;
        int v11 = 3;
        int v10 = 2;
        StringBuilder v0 = new StringBuilder();
        String v1 = S.s(346);
        v0.append(String.format(v1, String.format(                          Software")));
        v2 = new Object[11];
        v2[0] = S.s(242);
        v2[1] = "contact@pnfsoftware.com";
        v2[v10] = S.s(518);
        v2[v11] = Integer.valueOf(760781040);
        v2[4] = S.s(347);
        v2[v12] = Long.valueOf(1218204649873843338L
        v2[6] = S.s(86);
        v2[7] = Integer.valueOf(Licensing.build_ty
        v2[8] = Licensing.getBuildTypeString();
        v2[9] = S.s(289);
```

```
:lient/AbstractClientContext;->start()V
```

3. 破解 Jeb 注册码
   找到生成注册码字符串的函数，并对函数和类进行重命名标记

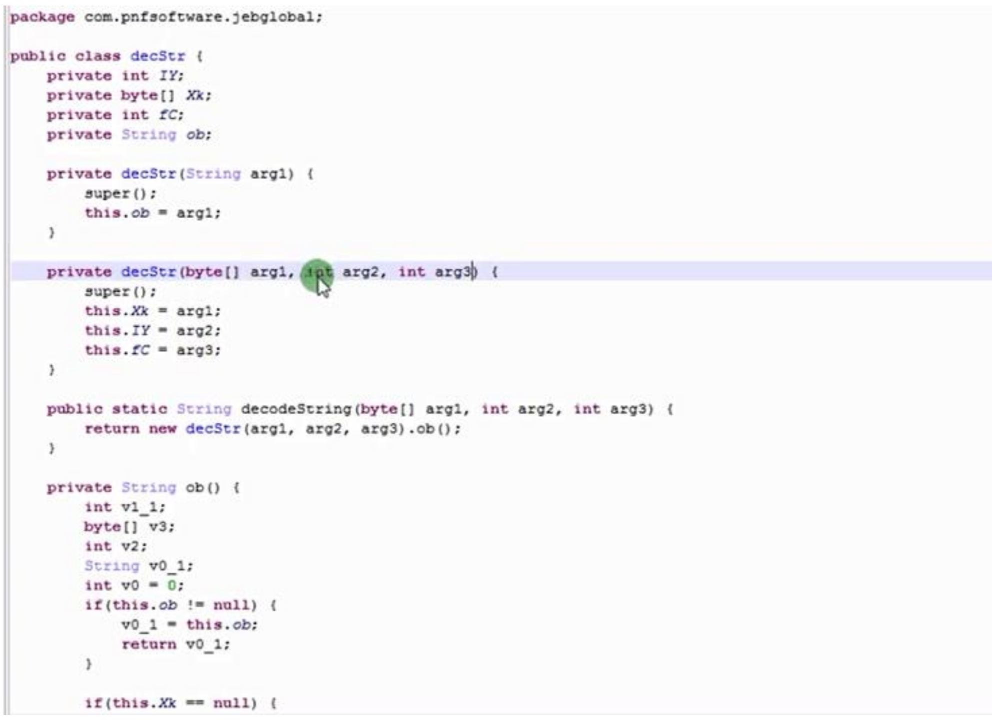


```java
public static final String getBuildTypeString() {
    int v7 = 5;
    int v6 = 11;
    int v5 = 8;
    int v3 = 2;
    StringBuilder v0 = new StringBuilder();
    if(Licensing.isReleaseBuild()) {
        v0.append(Si.ob(new byte[]{-39, 23, 9, 9, 4, 18, 22, 74}, 1, 171));
    }
    else {
        v0.append(Si.ob(new byte[]{39, 10, 18, 12, 21, 70}, v3, 28));
    }

    if(Licensing.isFullBuild()) {
        v0.append(Si.ob(new byte[]{34, 19, 25, 0, 67}, 1, 68));
    }
    else {
        v0.append(Si.ob(new byte[]{39, 10, 29, 22, 93}, v3, 245));
    }

    if(Licensing.isFloatingBuild()) {
        v0.append(Si.ob(new byte[]{37, 3, 31, 24, 6, 0, 9, 15, 91}, v3, 87));
    }
    else {
        v0.append(Si.ob(new byte[]{42, 1, 20, 16, 4, 0, 3, 29, 21, 76, 7}, v3, 120));
```

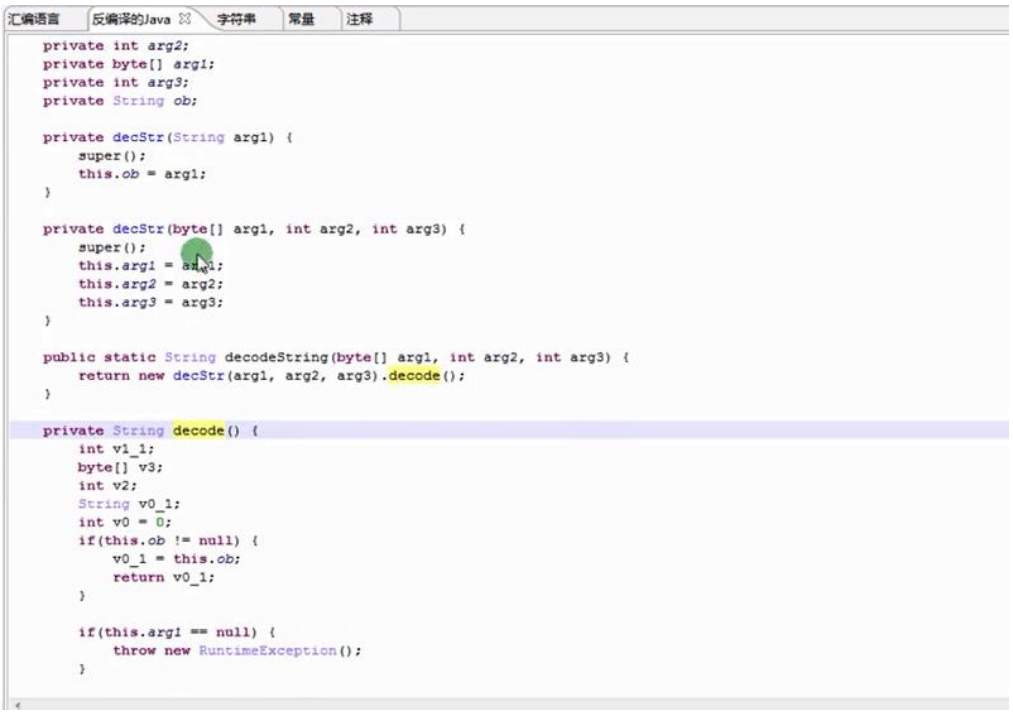


```java
public static final String getBuildTypeString() {
    int v7 = 5;
    int v6 = 11;
    int v5 = 8;
    int v3 = 2;
    StringBuilder v0 = new StringBuilder();
    if(Licensing.isReleaseBuild()) {
        v0.append(decStr.decodeString(new byte[]{-39, 23, 9, 9, 4, 18, 22, 74}, 1, 171));
    }
    else {
        v0.append(decStr.decodeString(new byte[]{39, 10, 18, 12, 21, 70}, v3, 28));
    }

    if(Licensing.isFullBuild()) {
        v0.append(decStr.decodeString(new byte[]{34, 19, 25, 0, 67}, 1, 68));
    }
    else {
        v0.append(decStr.decodeString(new byte[]{39, 10, 29, 22, 93}, v3, 245));
    }

    if(Licensing.isFloatingBuild()) {
        v0.append(decStr.decodeString(new byte[]{37, 3, 31, 24, 6, 0, 9, 15, 91}, v3, 87));
```

跳转到解码函数中

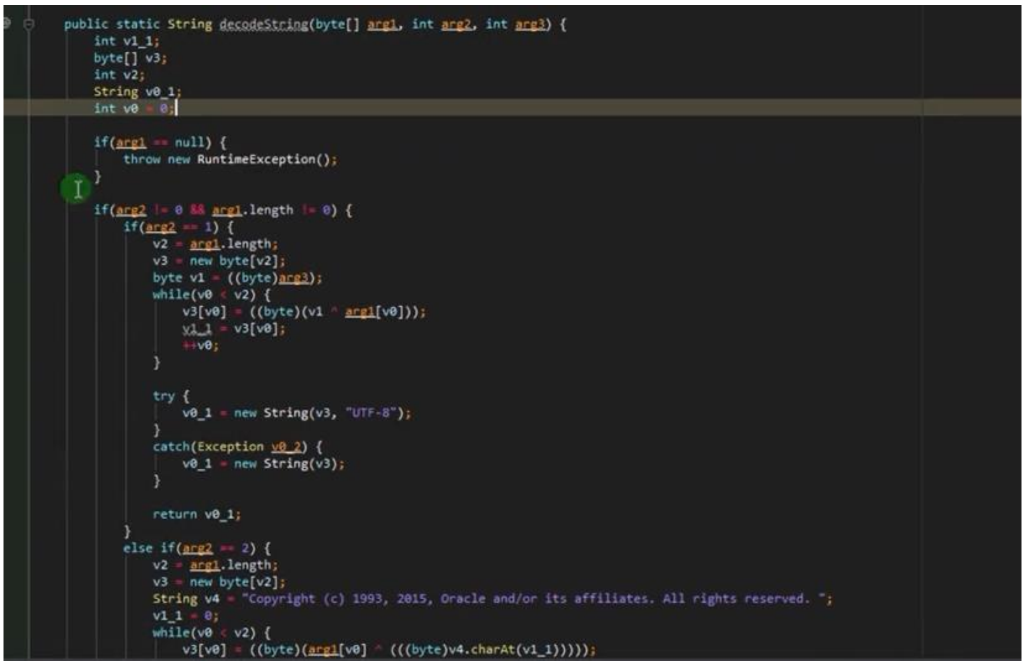


```java
package com.pnfsoftware.jebglobal;

public class decStr {
    private int IY;
    private byte[] Xk;
    private int fC;
    private String ob;

    private decStr(String arg1) {
        super();
        this.ob = arg1;
    }

    private decStr(byte[] arg1, int arg2, int arg3) {
        super();
        this.Xk = arg1;
        this.IY = arg2;
        this.fC = arg3;
    }

    public static String decodeString(byte[] arg1, int arg2, int arg3) {
        return new decStr(arg1, arg2, arg3).ob();
    }

    private String ob() {
        int v1_1;
        byte[] v3;
        int v2;
        String v0_1;
        int v0 = 0;
        if(this.ob != null) {
            v0_1 = this.ob;
            return v0_1;
        }

        if(this.Xk == null) {
```

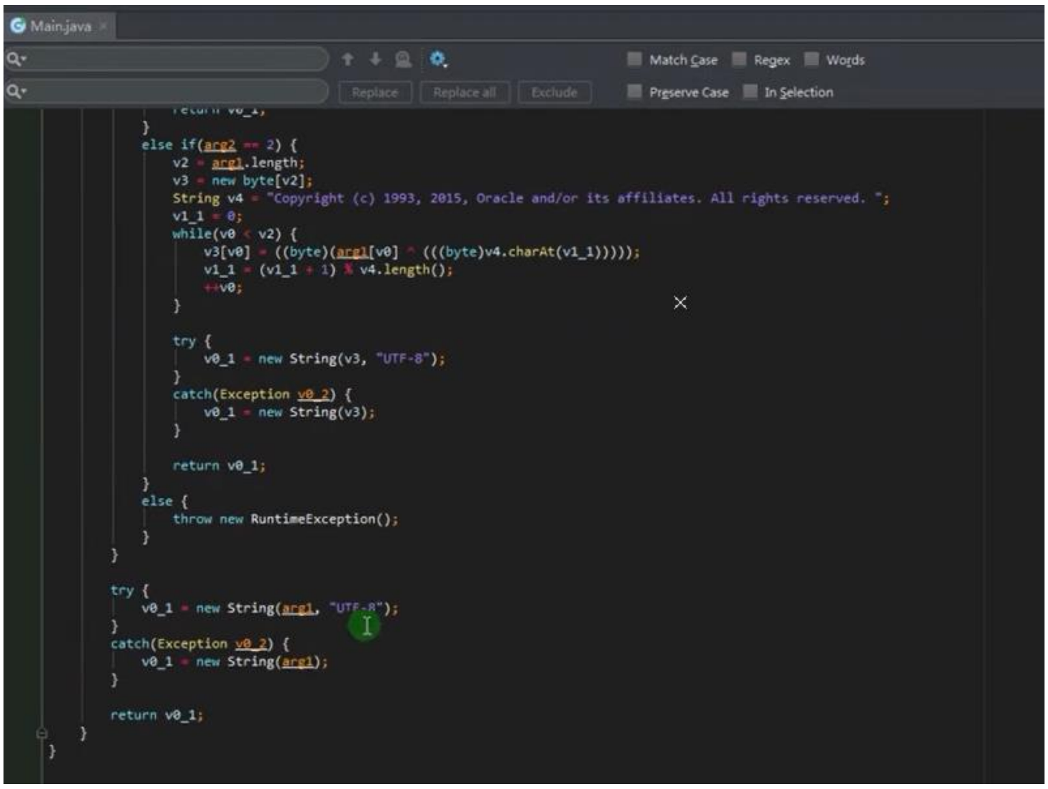


```java
    private int arg2;
    private byte[] arg1;
    private int arg3;
    private String ob;

    private decStr(String arg1) {
        super();
        this.ob = arg1;
    }

    private decStr(byte[] arg1, int arg2, int arg3) {
        super();
        this.arg1 = arg1;
        this.arg2 = arg2;
        this.arg3 = arg3;
    }

    public static String decodeString(byte[] arg1, int arg2, int arg3) {
        return new decStr(arg1, arg2, arg3).decode();
    }

    private String decode() {
        int v1_1;
        byte[] v3;
        int v2;
        String v0_1;
        int v0 = 0;
        if(this.ob != null) {
            v0_1 = this.ob;
            return v0_1;
        }

        if(this.arg1 == null) {
            throw new RuntimeException();
        }
```

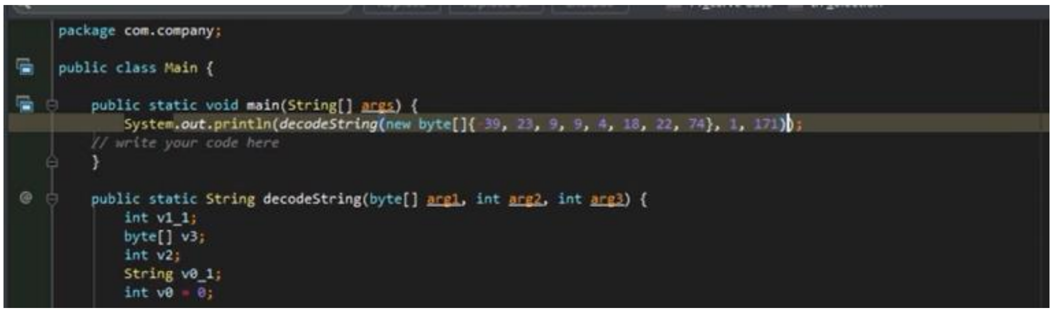
然后根据解密函数的代码编写一个相对应的程序



```java
    public static String decodeString(byte[] arg1, int arg2, int arg3) {
        int v1_1;
        byte[] v3;
        int v2;
        String v0_1;
        int v0 = 0;

        if(arg1 == null) {
            throw new RuntimeException();
        }

        if(arg2 != 0 && arg1.length != 0) {
            if(arg2 == 1) {
                v2 = arg1.length;
                v3 = new byte[v2];
                byte v1 = ((byte)arg3);
                while(v0 < v2) {
                    v3[v0] = ((byte)(v1 ^ arg1[v0]));
                    v1_1 = v3[v0];
                    ++v0;
                }

                try {
                    v0_1 = new String(v3, "UTF-8");
                }
                catch(Exception v0_2) {
                    v0_1 = new String(v3);
                }

                return v0_1;
            }
            else if(arg2 == 2) {
                v2 = arg1.length;
                v3 = new byte[v2];
                String v4 = "Copyright (c) 1993, 2015, Oracle and/or its affiliates. All rights reserved. ";
                v1_1 = 0;
                while(v0 < v2) {
                    v3[v0] = ((byte)(arg1[v0] ^ (((byte)v4.charAt(v1_1)))));
```

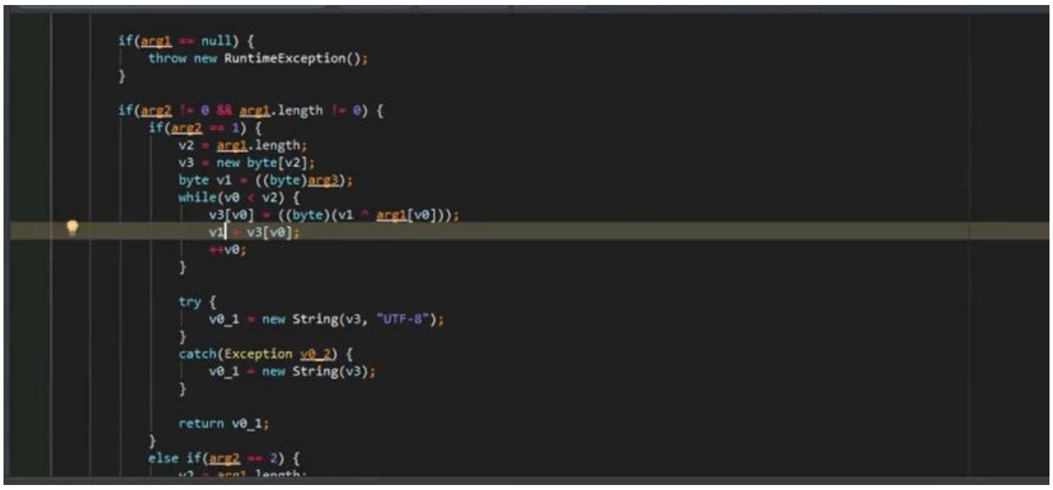


```java
            }
            else if(arg2 == 2) {
                v2 = arg1.length;
                v3 = new byte[v2];
                String v4 = "Copyright (c) 1993, 2015, Oracle and/or its affiliates. All rights reserved. ";
                v1_1 = 0;
                while(v0 < v2) {
                    v3[v0] = ((byte)(arg1[v0] ^ (((byte)v4.charAt(v1_1)))));
                    v1_1 = (v1_1 + 1) % v4.length();
                    ++v0;
                }

                try {
                    v0_1 = new String(v3, "UTF-8");
                }
                catch(Exception v0_2) {
                    v0_1 = new String(v3);
                }

                return v0_1;
            }
            else {
                throw new RuntimeException();
            }
        }

        try {
            v0_1 = new String(arg1, "UTF-8");
        }
        catch(Exception v0_2) {
            v0_1 = new String(arg1);
        }

        return v0_1;
    }
}
```



```java
package com.company;

public class Main {

    public static void main(String[] args) {
        System.out.println(decodeString(new byte[]{ 39, 23, 9, 9, 4, 18, 22, 74}, 1, 171));
    // write your code here
    }

    public static String decodeString(byte[] arg1, int arg2, int arg3) {
        int v1_1;
        byte[] v3;
        int v2;
        String v0_1;
        int v0 = 0;
```

将 v1_1 修改为 v1，二者使用的寄存器都相同，但是因为类型发生了变化，所以出现了 v1_1。同理，v0_1 也是如此。



```java
        if(arg1 == null) {
            throw new RuntimeException();
        }

        if(arg2 != 0 && arg1.length != 0) {
            if(arg2 == 1) {
                v2 = arg1.length;
                v3 = new byte[v2];
                byte v1 = ((byte)arg3);
                while(v0 < v2) {
                    v3[v0] = ((byte)(v1 ^ arg1[v0]));
                    v1 = v3[v0];
                    ++v0;
                }

                try {
                    v0_1 = new String(v3, "UTF-8");
                }
                catch(Exception v0_2) {
                    v0_1 = new String(v3);
                }

                return v0_1;
            }
            else if(arg2 == 2) {
```

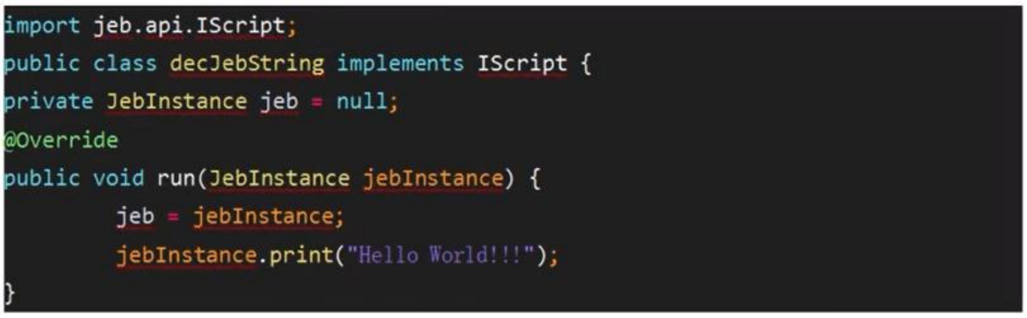
4. Jeb 高级技巧（插件扩展）

(1)插件帮助文档：Jeb/doc/apidoc

(2)插件编写：

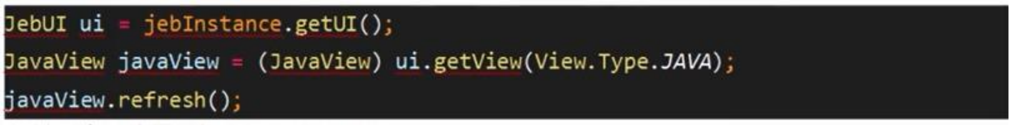
语言：Java or Python

入口：最简单的插件示例



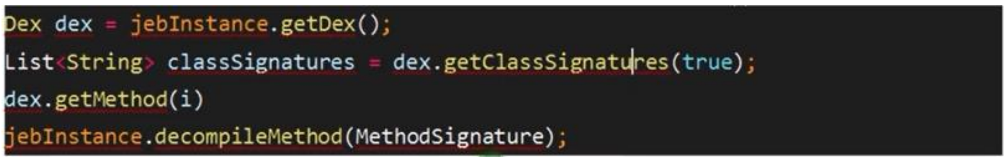
```java
import jeb.api.IScript;
public class decJebString implements IScript {
private JebInstance jeb = null;
@Override
public void run(JebInstance jebInstance) {
        jeb = jebInstance;
        jebInstance.print("Hello World!!!");
}
```

强制反编译结果，相当于 Ctrl + F5



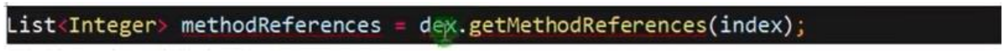
```java
JebUI ui = jebInstance.getUI();
JavaView javaView = (JavaView) ui.getView(View.Type.JAVA);
javaView.refresh();
```

获取所有类的签名，以及获取第 i 个方法的反编译方法（Tab）



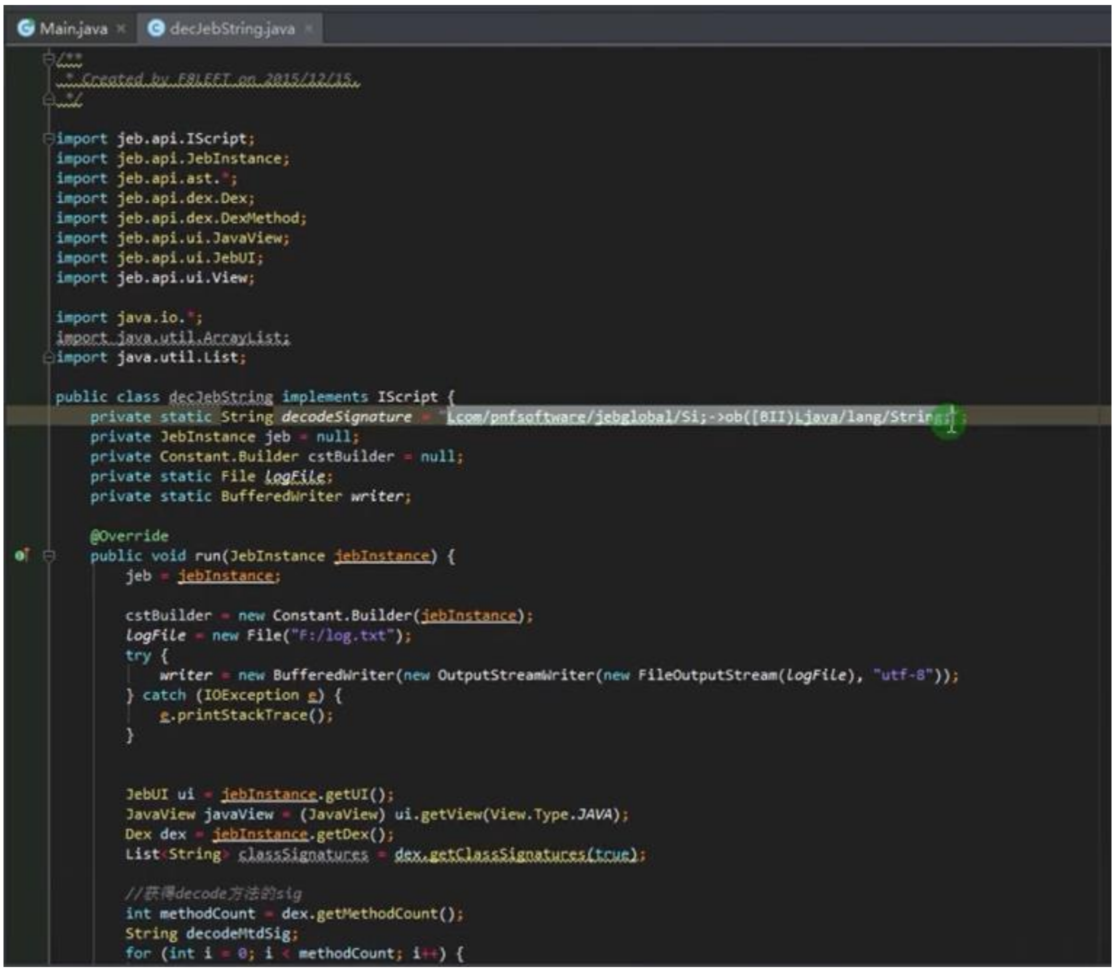
```java
Dex dex = jebInstance.getDex();
List<String> classSignatures = dex.getClassSignatures(true);
dex.getMethod(i)
jebInstance.decompileMethod(MethodSignature);
```

获取第 i 个方法的交叉引用



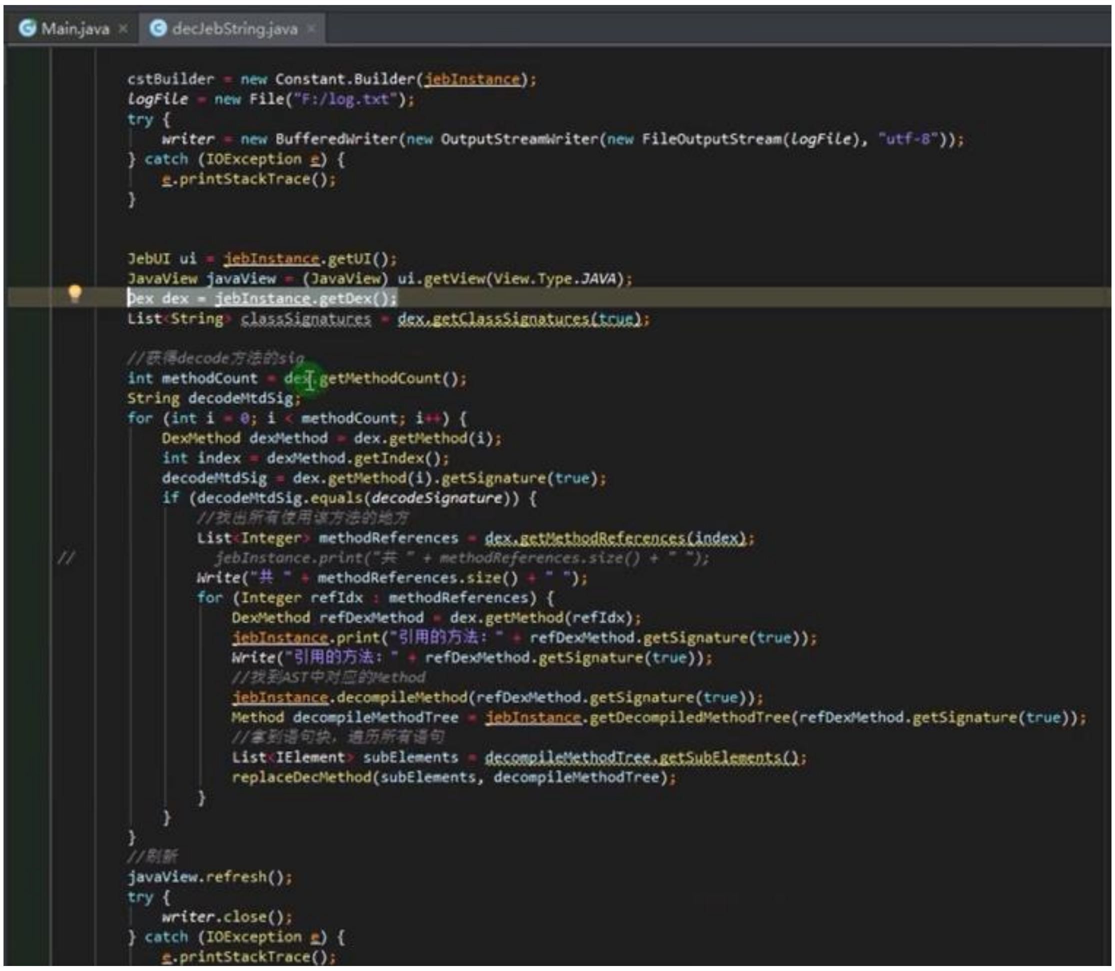
```java
List<Integer> methodReferences = dex.getMethodReferences(index);
```

Jeb 字符串的还原脚本



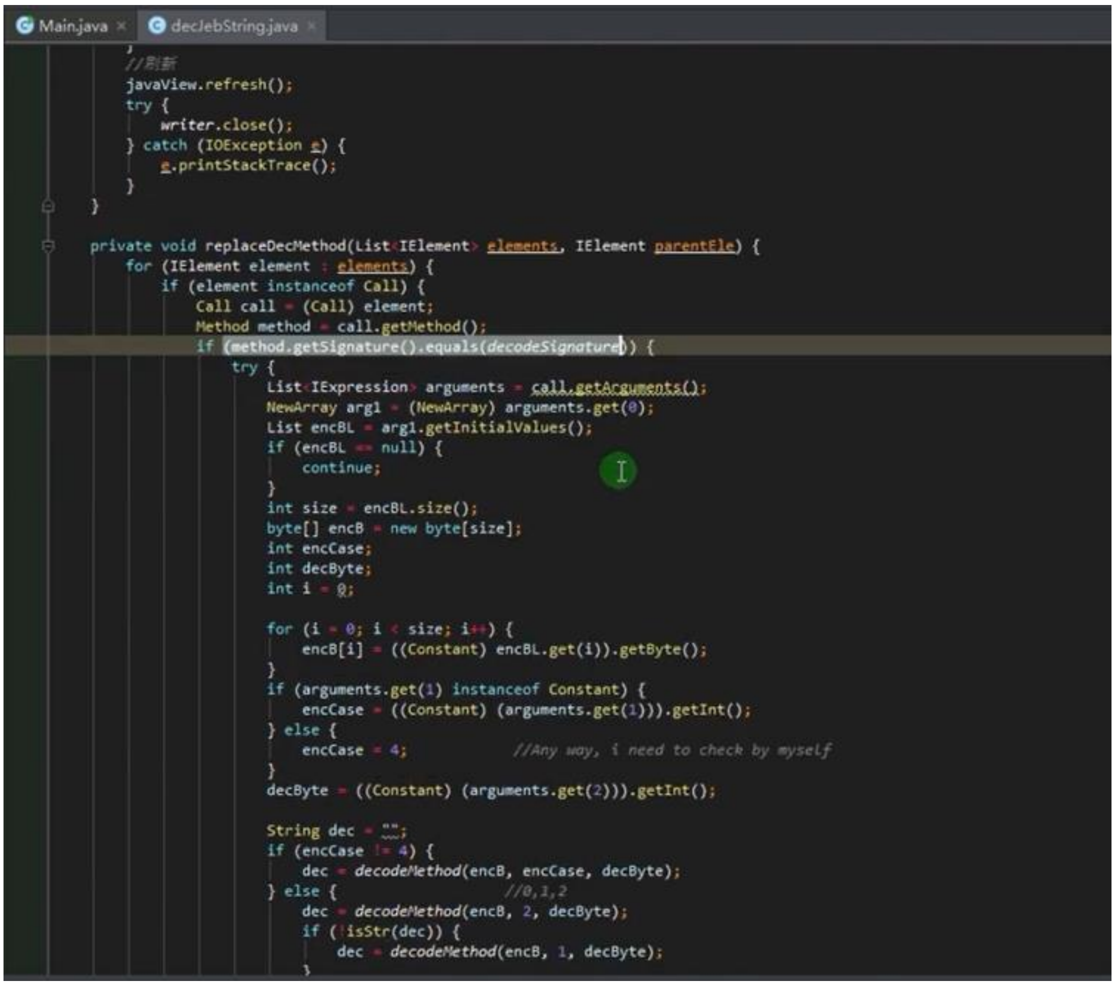
Main.java　decJebString.java

```java
/**
 * Created by F8LEFT on 2015/12/15.
 */

import jeb.api.IScript;
import jeb.api.JebInstance;
import jeb.api.ast.*;
import jeb.api.dex.Dex;
import jeb.api.dex.DexMethod;
import jeb.api.ui.JavaView;
import jeb.api.ui.JebUI;
import jeb.api.ui.View;

import java.io.*;
import java.util.ArrayList;
import java.util.List;

public class decJebString implements IScript {
    private static String decodeSignature = "Lcom/pnfsoftware/jebglobal/Si;->ob([BII)Ljava/lang/String;";
    private JebInstance jeb = null;
    private Constant.Builder cstBuilder = null;
    private static File logFile;
    private static BufferedWriter writer;

    @Override
    public void run(JebInstance jebInstance) {
        jeb = jebInstance;

        cstBuilder = new Constant.Builder(jebInstance);
        logFile = new File("F:/log.txt");
        try {
            writer = new BufferedWriter(new OutputStreamWriter(new FileOutputStream(logFile), "utf-8"));
        } catch (IOException e) {
            e.printStackTrace();
        }


        JebUI ui = jebInstance.getUI();
        JavaView javaView = (JavaView) ui.getView(View.Type.JAVA);
        Dex dex = jebInstance.getDex();
        List<String> classSignatures = dex.getClassSignatures(true);

        //获得decode方法的sig
        int methodCount = dex.getMethodCount();
        String decodeMtdSig;
        for (int i = 0; i < methodCount; i++) {
```

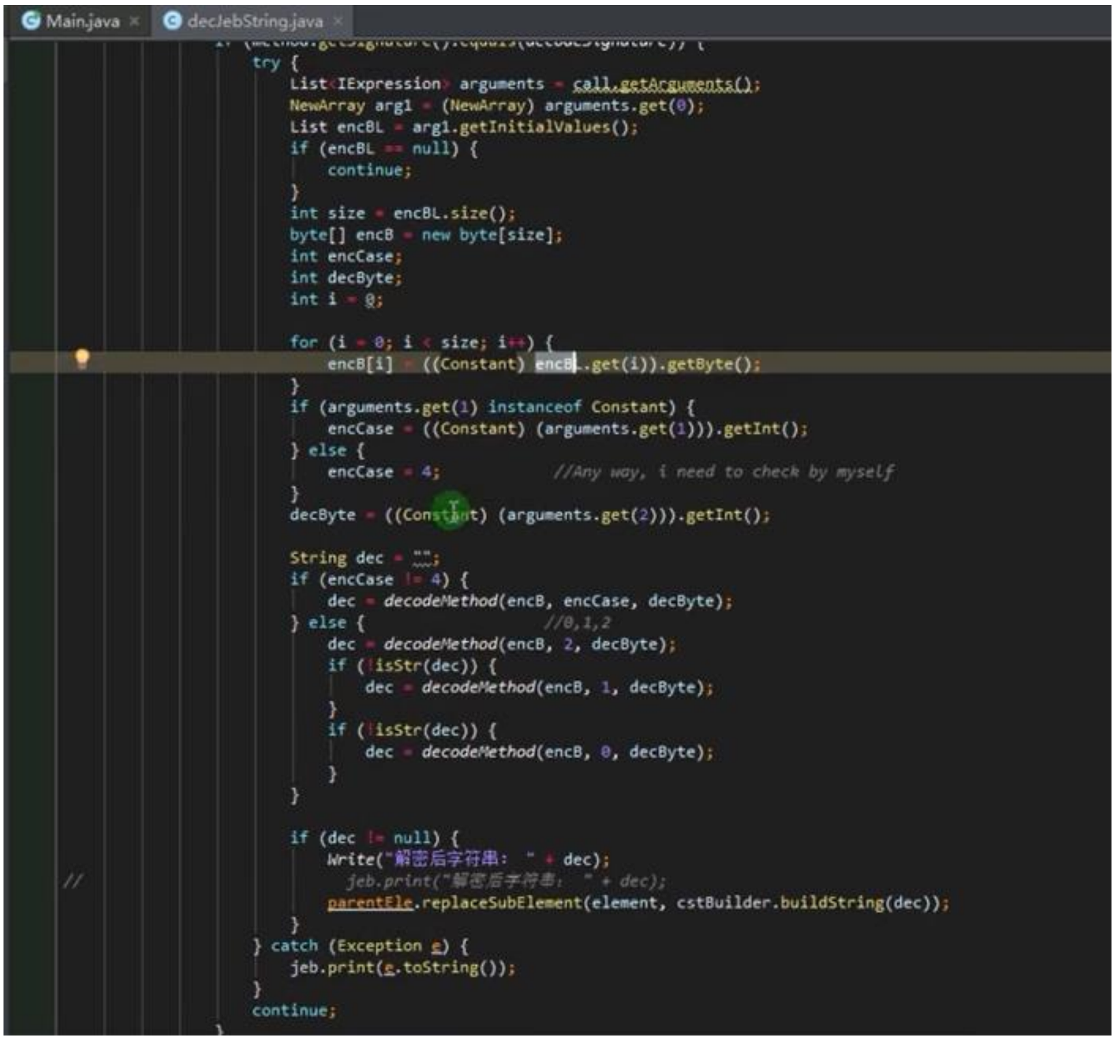


Main.java × decJebString.java ×

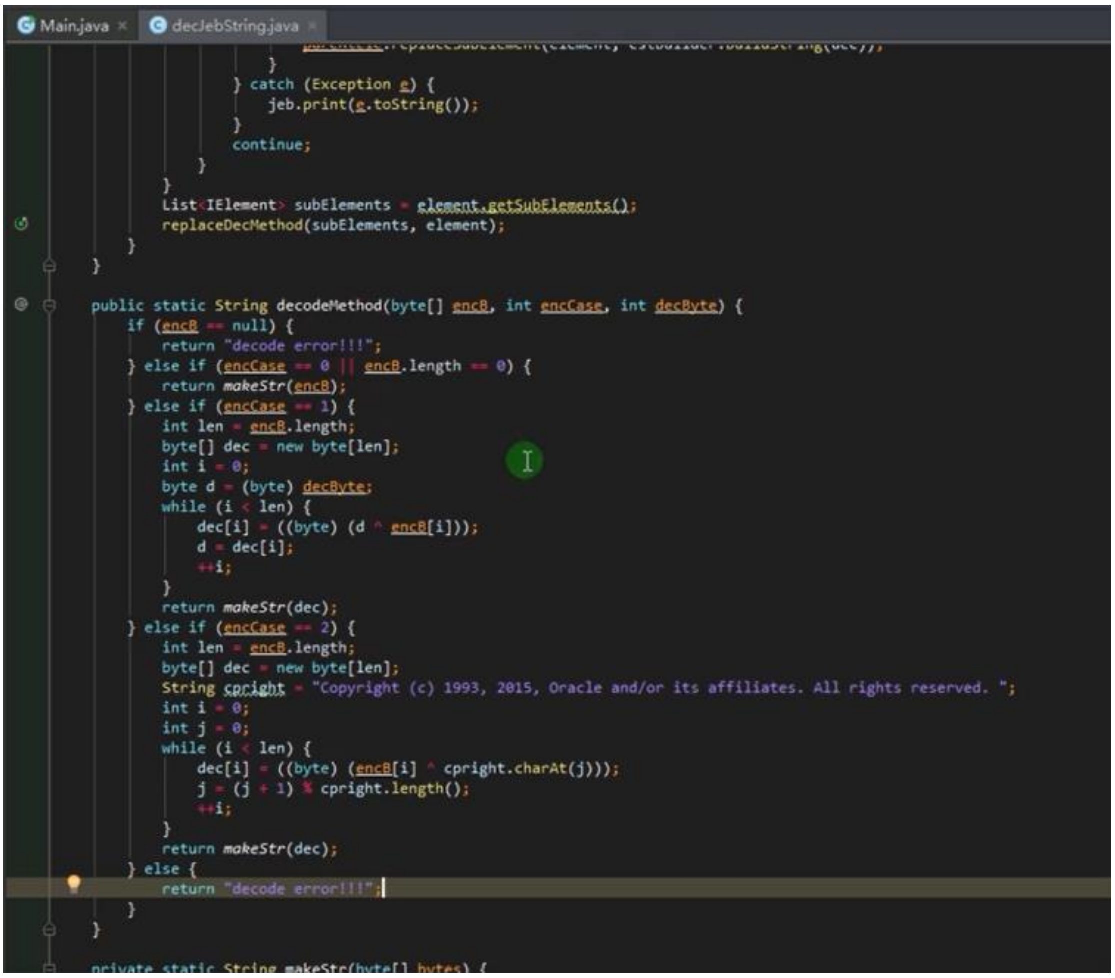
```java
cstBuilder = new Constant.Builder(jebInstance);
logFile = new File("F:/log.txt");
try {
    writer = new BufferedWriter(new OutputStreamWriter(new FileOutputStream(logFile), "utf-8"));
} catch (IOException e) {
    e.printStackTrace();
}


JebUI ui = jebInstance.getUI();
JavaView javaView = (JavaView) ui.getView(View.Type.JAVA);
Dex dex = jebInstance.getDex();
List<String> classSignatures = dex.getClassSignatures(true);

//获得decode方法的sig
int methodCount = dex.getMethodCount();
String decodeMtdSig;
for (int i = 0; i < methodCount; i++) {
    DexMethod dexMethod = dex.getMethod(i);
    int index = dexMethod.getIndex();
    decodeMtdSig = dex.getMethod(i).getSignature(true);
    if (decodeMtdSig.equals(decodeSignature)) {
        //找出所有使用该方法的地方
        List<Integer> methodReferences = dex.getMethodReferences(index);
//        jebInstance.print("共 " + methodReferences.size() + " ");
        Write("共 " + methodReferences.size() + " ");
        for (Integer refIdx : methodReferences) {
            DexMethod refDexMethod = dex.getMethod(refIdx);
            jebInstance.print("引用的方法: " + refDexMethod.getSignature(true));
            Write("引用的方法: " + refDexMethod.getSignature(true));
            //找到AST中对应的Method
            jebInstance.decompileMethod(refDexMethod.getSignature(true));
            Method decompileMethodTree = jebInstance.getDecompiledMethodTree(refDexMethod.getSignature(true));
            //拿到语句块，遍历所有语句
            List<IElement> subElements = decompileMethodTree.getSubElements();
            replaceDecMethod(subElements, decompileMethodTree);
        }
    }
}
//刷新
javaView.refresh();
try {
    writer.close();
} catch (IOException e) {
    e.printStackTrace();
```



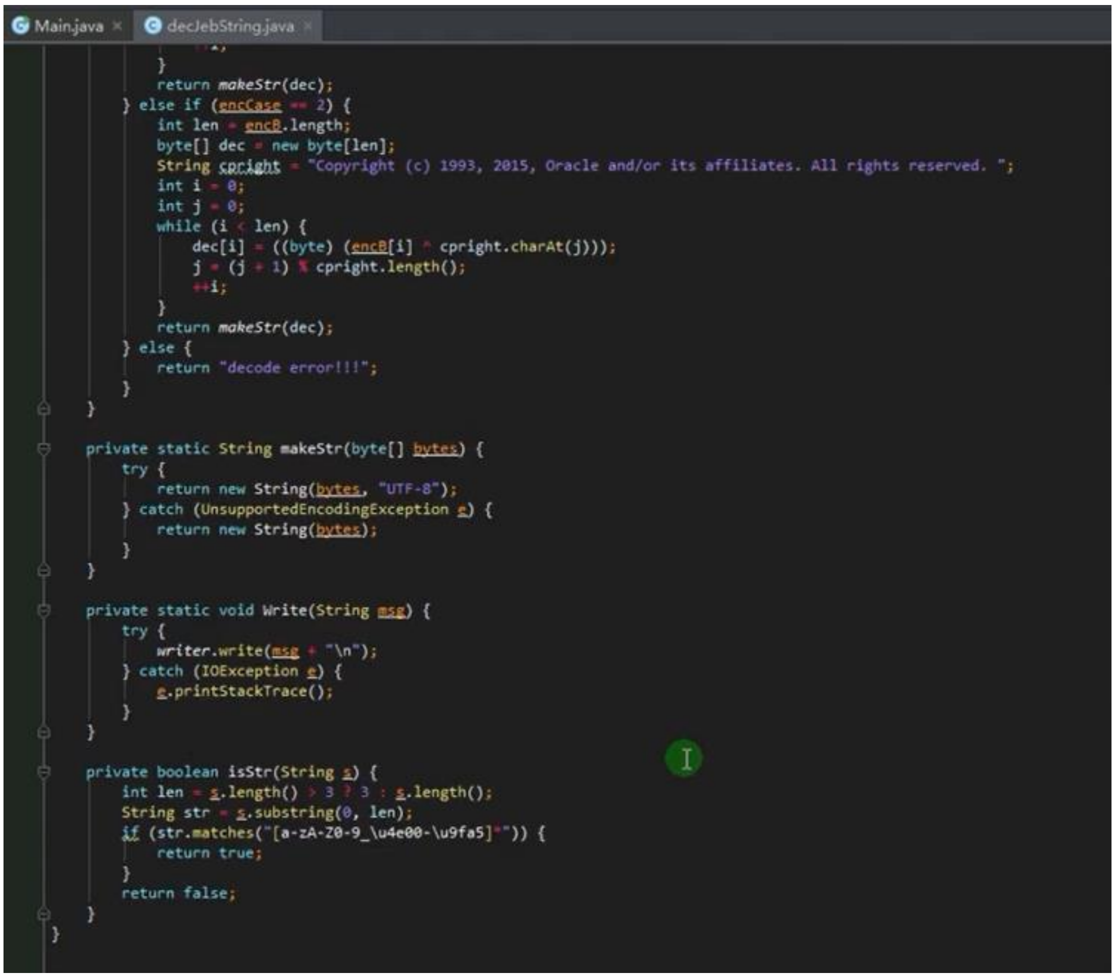
```java
        }
        //刷新
        javaView.refresh();
        try {
            writer.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    private void replaceDecMethod(List<IElement> elements, IElement parentEle) {
        for (IElement element : elements) {
            if (element instanceof Call) {
                Call call = (Call) element;
                Method method = call.getMethod();
                if (method.getSignature().equals(decodeSignature)) {
                    try {
                        List<IExpression> arguments = call.getArguments();
                        NewArray arg1 = (NewArray) arguments.get(0);
                        List encBL = arg1.getInitialValues();
                        if (encBL == null) {
                            continue;
                        }
                        int size = encBL.size();
                        byte[] encB = new byte[size];
                        int encCase;
                        int decByte;
                        int i = 0;

                        for (i = 0; i < size; i++) {
                            encB[i] = ((Constant) encBL.get(i)).getByte();
                        }
                        if (arguments.get(1) instanceof Constant) {
                            encCase = ((Constant) (arguments.get(1))).getInt();
                        } else {
                            encCase = 4;            //Any way, i need to check by myself
                        }
                        decByte = ((Constant) (arguments.get(2))).getInt();

                        String dec = "";
                        if (encCase != 4) {
                            dec = decodeMethod(encB, encCase, decByte);
                        } else {                    //0,1,2
                            dec = decodeMethod(encB, 2, decByte);
                            if (!isStr(dec)) {
                                dec = decodeMethod(encB, 1, decByte);
                            }
```



```
if (method.getSignature().equals(decodeSignature)) {
    try {
        List<IExpression> arguments = call.getArguments();
        NewArray arg1 = (NewArray) arguments.get(0);
        List encBL = arg1.getInitialValues();
        if (encBL == null) {
            continue;
        }
        int size = encBL.size();
        byte[] encB = new byte[size];
        int encCase;
        int decByte;
        int i = 0;

        for (i = 0; i < size; i++) {
            encB[i] = ((Constant) encBL.get(i)).getByte();
        }
        if (arguments.get(1) instanceof Constant) {
            encCase = ((Constant) (arguments.get(1))).getInt();
        } else {
            encCase = 4;            //Any way, i need to check by myself
        }
        decByte = ((Constant) (arguments.get(2))).getInt();

        String dec = "";
        if (encCase != 4) {
            dec = decodeMethod(encB, encCase, decByte);
        } else {                    //0,1,2
            dec = decodeMethod(encB, 2, decByte);
            if (!isStr(dec)) {
                dec = decodeMethod(encB, 1, decByte);
            }
            if (!isStr(dec)) {
                dec = decodeMethod(encB, 0, decByte);
            }
        }

        if (dec != null) {
            Write("解密后字符串:  " + dec);
//            jeb.print("解密后字符串:  " + dec);
            parentEle.replaceSubElement(element, cstBuilder.buildString(dec));
        }
    } catch (Exception e) {
        jeb.print(e.toString());
    }
    continue;
}
```

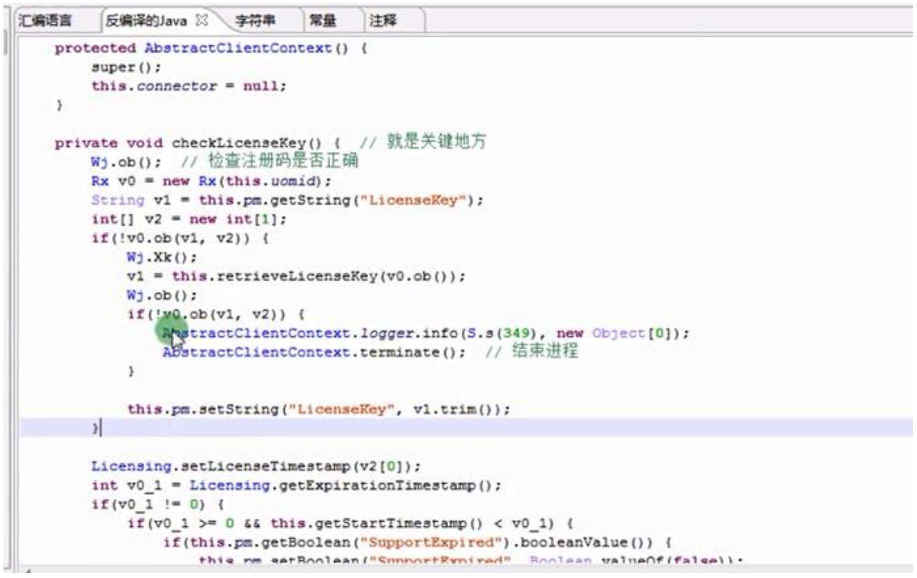


```java
                    }
                } catch (Exception e) {
                    jeb.print(e.toString());
                }
                continue;
            }
        }
        List<IElement> subElements = element.getSubElements();
        replaceDecMethod(subElements, element);
    }
}

public static String decodeMethod(byte[] encB, int encCase, int decByte) {
    if (encB == null) {
        return "decode error!!!";
    } else if (encCase == 0 || encB.length == 0) {
        return makeStr(encB);
    } else if (encCase == 1) {
        int len = encB.length;
        byte[] dec = new byte[len];
        int i = 0;
        byte d = (byte) decByte;
        while (i < len) {
            dec[i] = ((byte) (d ^ encB[i]));
            d = dec[i];
            ++i;
        }
        return makeStr(dec);
    } else if (encCase == 2) {
        int len = encB.length;
        byte[] dec = new byte[len];
        String cpright = "Copyright (c) 1993, 2015, Oracle and/or its affiliates. All rights reserved. ";
        int i = 0;
        int j = 0;
        while (i < len) {
            dec[i] = ((byte) (encB[i] ^ cpright.charAt(j)));
            j = (j + 1) % cpright.length();
            ++i;
        }
        return makeStr(dec);
    } else {
        return "decode error!!!";
    }
}

private static String makeStr(byte[] bytes) {
```

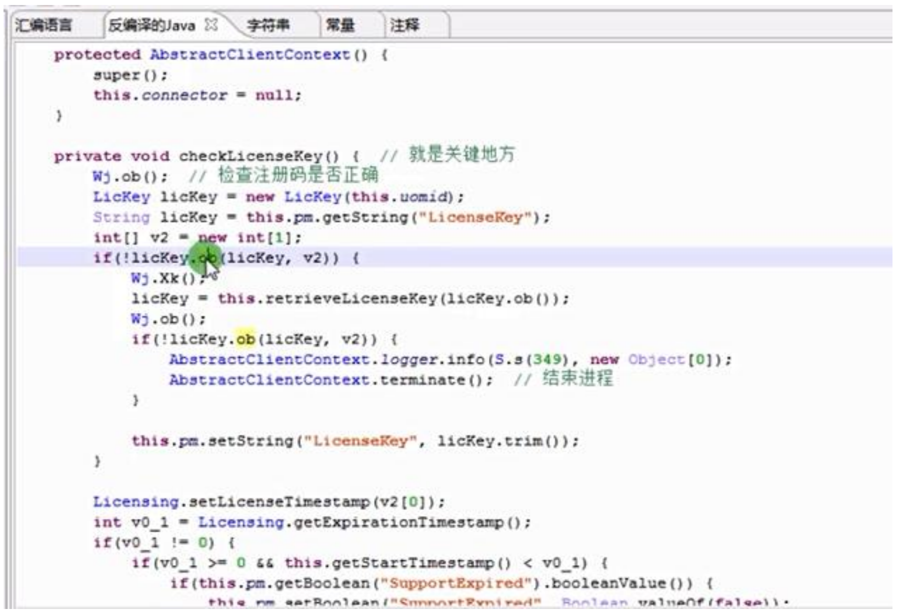


```java
            }
            return makeStr(dec);
        } else if (encCase == 2) {
            int len = encB.length;
            byte[] dec = new byte[len];
            String cpright = "Copyright (c) 1993, 2015, Oracle and/or its affiliates. All rights reserved. ";
            int i = 0;
            int j = 0;
            while (i < len) {
                dec[i] = ((byte) (encB[i] ^ cpright.charAt(j)));
                j = (j + 1) % cpright.length();
                ++i;
            }
            return makeStr(dec);
        } else {
            return "decode error!!!";
        }
    }

    private static String makeStr(byte[] bytes) {
        try {
            return new String(bytes, "UTF-8");
        } catch (UnsupportedEncodingException e) {
            return new String(bytes);
        }
    }

    private static void Write(String msg) {
        try {
            writer.write(msg + "\n");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    private boolean isStr(String s) {
        int len = s.length() > 3 ? 3 : s.length();
        String str = s.substring(0, len);
        if (str.matches("[a-zA-Z0-9_\u4e00-\u9fa5]*")) {
            return true;
        }
        return false;
    }
}
```

找到检查注册码的函数

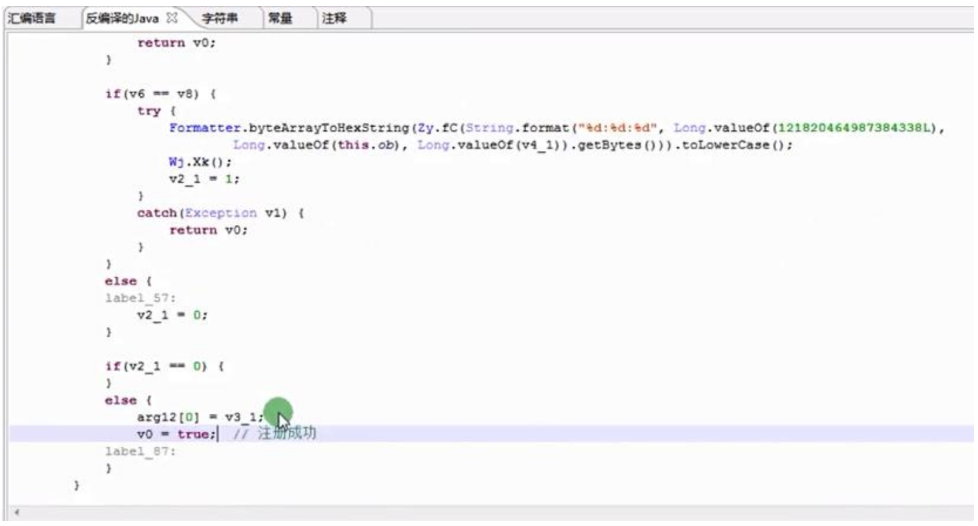


```java
    protected AbstractClientContext() {
        super();
        this.connector = null;
    }

    private void checkLicenseKey() {  // 就是关键地方
        Wj.ob();  // 检查注册码是否正确
        Rx v0 = new Rx(this.uomid);
        String v1 = this.pm.getString("LicenseKey");
        int[] v2 = new int[1];
        if(!v0.ob(v1, v2)) {
            Wj.Xk();
            v1 = this.retrieveLicenseKey(v0.ob());
            Wj.ob();
            if(!v0.ob(v1, v2)) {
                AbstractClientContext.logger.info(S.s(349), new Object[0]);
                AbstractClientContext.terminate();  // 结束进程
            }

            this.pm.setString("LicenseKey", v1.trim());
        }

        Licensing.setLicenseTimestamp(v2[0]);
        int v0_1 = Licensing.getExpirationTimestamp();
        if(v0_1 != 0) {
            if(v0_1 >= 0 && this.getStartTimestamp() < v0_1) {
                if(this.pm.getBoolean("SupportExpired").booleanValue()) {
```



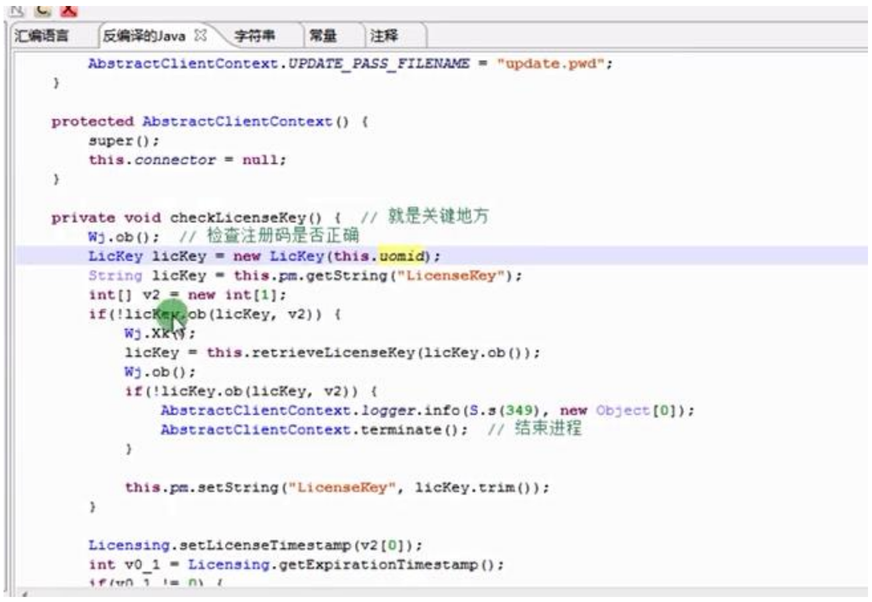
```
protected AbstractClientContext() {
    super();
    this.connector = null;
}

private void checkLicenseKey() {  // 就是关键地方
    Wj.ob();  // 检查注册码是否正确
    LicKey licKey = new LicKey(this.uomid);
    String licKey = this.pm.getString("LicenseKey");
    int[] v2 = new int[1];
    if(!licKey.ob(licKey, v2)) {
        Wj.Xk();
        licKey = this.retrieveLicenseKey(licKey.ob());
        Wj.ob();
        if(!licKey.ob(licKey, v2)) {
            AbstractClientContext.logger.info(S.s(349), new Object[0]);
            AbstractClientContext.terminate();  // 结束进程
        }

        this.pm.setString("LicenseKey", licKey.trim());
    }

    Licensing.setLicenseTimestamp(v2[0]);
    int v0_1 = Licensing.getExpirationTimestamp();
    if(v0_1 != 0) {
        if(v0_1 >= 0 && this.getStartTimestamp() < v0_1) {
            if(this.pm.getBoolean("SupportExpired").booleanValue()) {
                this.pm.setBoolean("SupportExpired", Boolean.valueOf(false));
```

查看对比注册码的函数 ob 的代码，找到注册成功的标志位



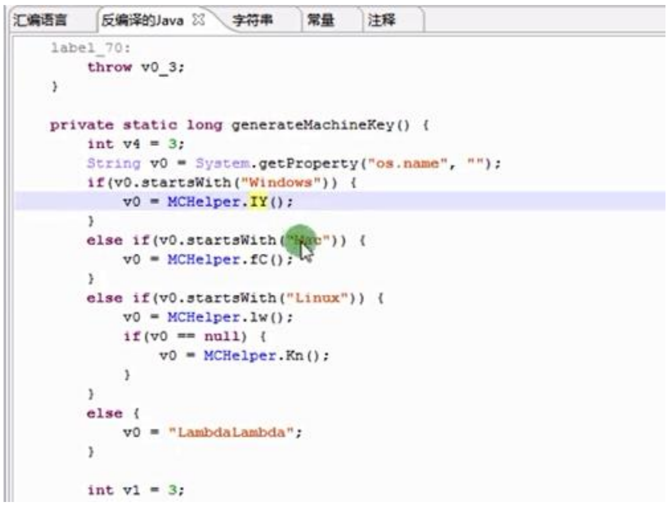
```
            return v0;
        }

        if(v6 == v8) {
            try {
                Formatter.byteArrayToHexString(Zy.fC(String.format("%d:%d:%d", Long.valueOf(1218204649873843338L),
                        Long.valueOf(this.ob), Long.valueOf(v4_1)).getBytes())).toLowerCase();
                Wj.Xk();
                v2_1 = 1;
            }
            catch(Exception v1) {
                return v0;
            }
        }
        else {
        label_57:
            v2_1 = 0;
        }

        if(v2_1 == 0) {
        }
        else {
            arg12[0] = v3_1;
            v0 = true;  // 注册成功
        label_87:
        }
    }
```

查看注册码生成函数 Lickey 的代码，uomid 为机器码

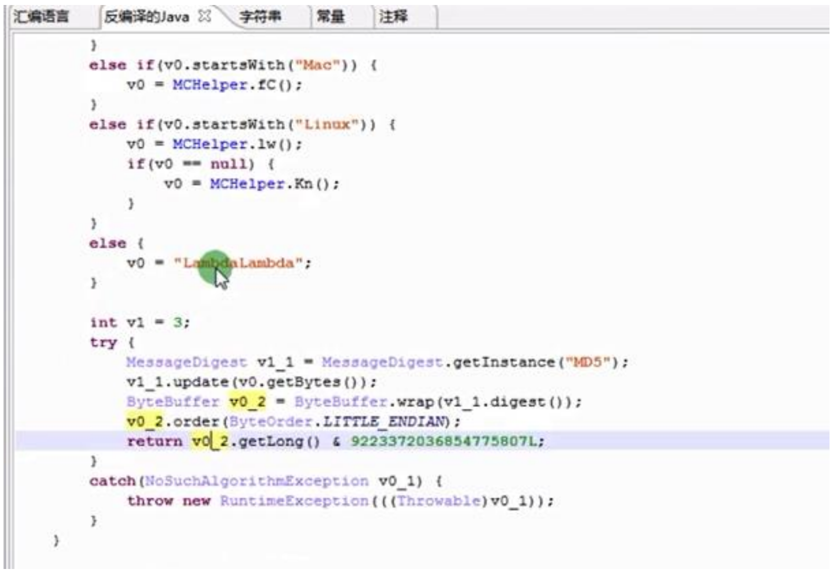


```
        AbstractClientContext.UPDATE_PASS_FILENAME = "update.pwd";
    }

    protected AbstractClientContext() {
        super();
        this.connector = null;
    }

    private void checkLicenseKey() {  // 就是关键地方
        Wj.ob();  // 检查注册码是否正确
        LicKey licKey = new LicKey(this.uomid);
        String licKey = this.pm.getString("LicenseKey");
        int[] v2 = new int[1];
        if(!licKey.ob(licKey, v2)) {
            Wj.Xk();
            licKey = this.retrieveLicenseKey(licKey.ob());
            Wj.ob();
            if(!licKey.ob(licKey, v2)) {
                AbstractClientContext.logger.info(S.s(349), new Object[0]);
                AbstractClientContext.terminate();  // 结束进程
            }

            this.pm.setString("LicenseKey", licKey.trim());
        }

        Licensing.setLicenseTimestamp(v2[0]);
        int v0_1 = Licensing.getExpirationTimestamp();
        if(v0_1 != 0) {
```

生成机器码的函数

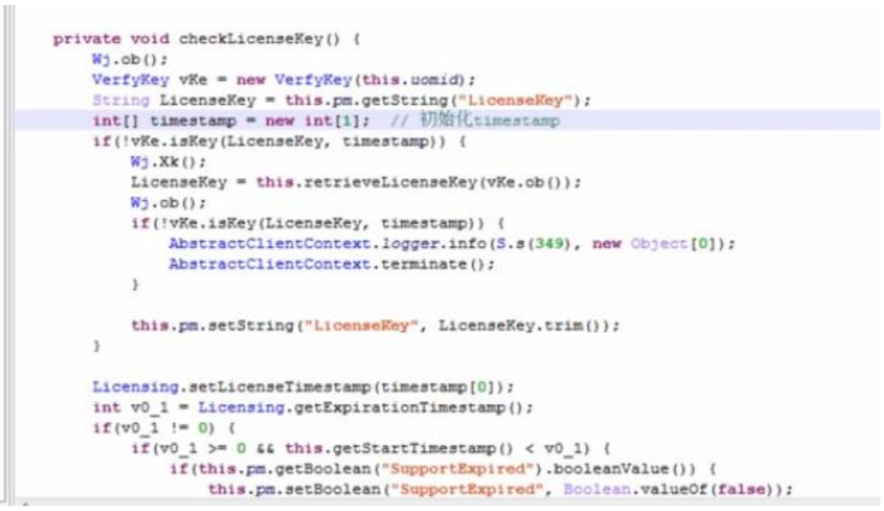


```java
    label_70:
        throw v0_3;
    }

    private static long generateMachineKey() {
        int v4 = 3;
        String v0 = System.getProperty("os.name", "");
        if(v0.startsWith("Windows")) {
            v0 = MCHelper.IY();
        }
        else if(v0.startsWith("Mac")) {
            v0 = MCHelper.fC();
        }
        else if(v0.startsWith("Linux")) {
            v0 = MCHelper.lw();
            if(v0 == null) {
                v0 = MCHelper.Kn();
            }
        }
        else {
            v0 = "LambdaLambda";
        }

        int v1 = 3;
```



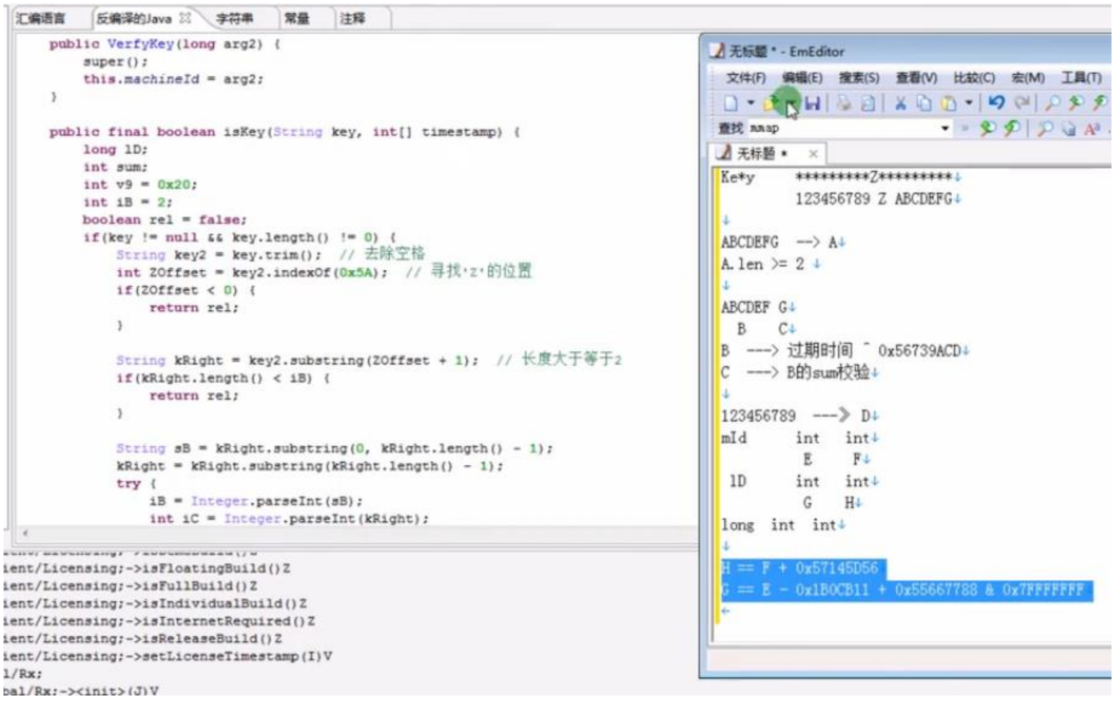
```java
        }
        else if(v0.startsWith("Mac")) {
            v0 = MCHelper.fC();
        }
        else if(v0.startsWith("Linux")) {
            v0 = MCHelper.lw();
            if(v0 == null) {
                v0 = MCHelper.Kn();
            }
        }
        else {
            v0 = "LambdaLambda";
        }

        int v1 = 3;
        try {
            MessageDigest v1_1 = MessageDigest.getInstance("MD5");
            v1_1.update(v0.getBytes());
            ByteBuffer v0_2 = ByteBuffer.wrap(v1_1.digest());
            v0_2.order(ByteOrder.LITTLE_ENDIAN);
            return v0_2.getLong() & 9223372036854775807L;
        }
        catch(NoSuchAlgorithmException v0_1) {
            throw new RuntimeException(((Throwable)v0_1));
        }
    }
```

在进入到 Windows 对应的机器码生成函数中查看，最后一步步的往上回溯。



```java
    private void checkLicenseKey() {
        Wj.ob();
        VerfyKey vKe = new VerfyKey(this.uomid);
        String LicenseKey = this.pm.getString("LicenseKey");
        int[] timestamp = new int[1];  // 初始化timestamp
        if(!vKe.isKey(LicenseKey, timestamp)) {
            Wj.Xk();
            LicenseKey = this.retrieveLicenseKey(vKe.ob());
            Wj.ob();
            if(!vKe.isKey(LicenseKey, timestamp)) {
                AbstractClientContext.logger.info(S.s(349), new Object[0]);
                AbstractClientContext.terminate();
            }

            this.pm.setString("LicenseKey", LicenseKey.trim());
        }

        Licensing.setLicenseTimestamp(timestamp[0]);
        int v0_1 = Licensing.getExpirationTimestamp();
        if(v0_1 != 0) {
            if(v0_1 >= 0 && this.getStartTimestamp() < v0_1) {
                if(this.pm.getBoolean("SupportExpired").booleanValue()) {
                    this.pm.setBoolean("SupportExpired", Boolean.valueOf(false));
```

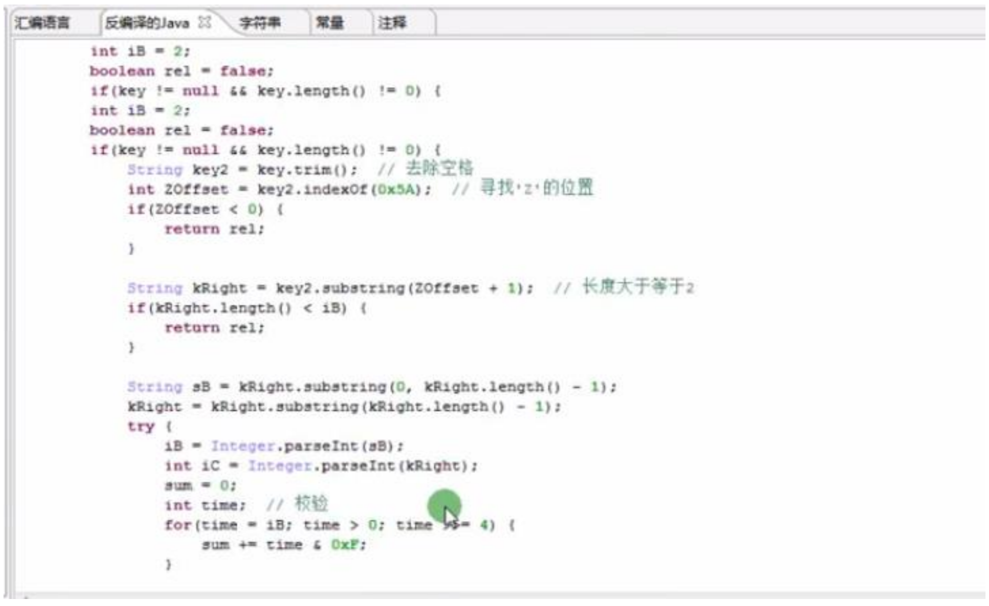
isKey()函数的分析



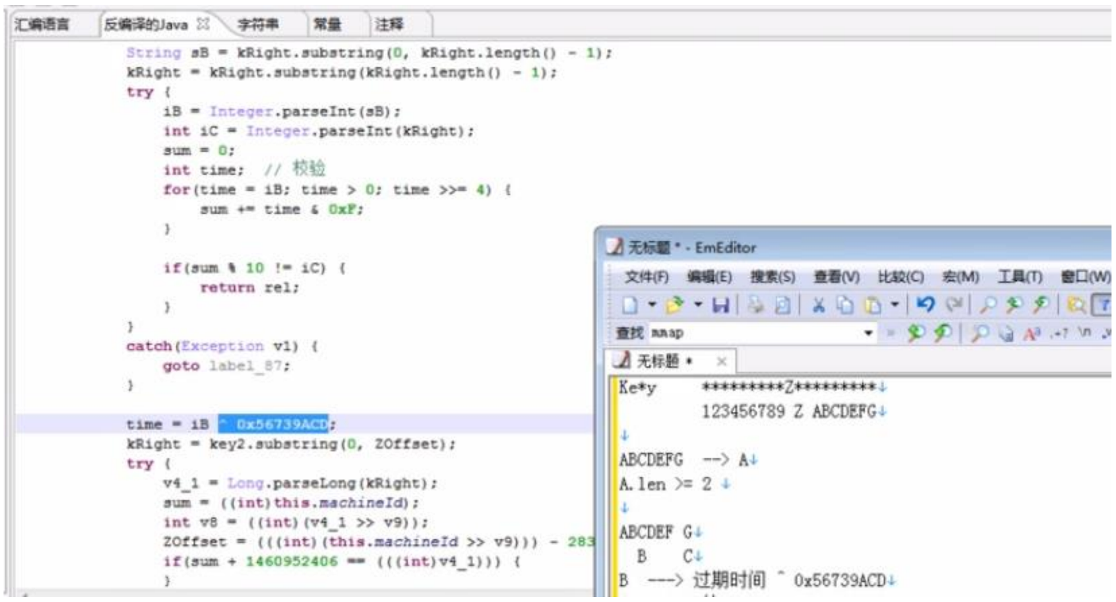
```
    public VerfyKey(long arg2) {
        super();
        this.machineId = arg2;
    }

    public final boolean isKey(String key, int[] timestamp) {
        long lD;
        int sum;
        int v9 = 0x20;
        int iB = 2;
        boolean rel = false;
        if(key != null && key.length() != 0) {
            String key2 = key.trim();  // 去除空格
            int ZOffset = key2.indexOf(0x5A);  // 寻找'z'的位置
            if(ZOffset < 0) {
                return rel;
            }

            String kRight = key2.substring(ZOffset + 1);  // 长度大于等于2
            if(kRight.length() < iB) {
                return rel;
            }

            String sB = kRight.substring(0, kRight.length() - 1);
            kRight = kRight.substring(kRight.length() - 1);
            try {
                iB = Integer.parseInt(sB);
                int iC = Integer.parseInt(kRight);
```

```
ient/Licensing;->isFloatingBuild()Z
ient/Licensing;->isFullBuild()Z
ient/Licensing;->isIndividualBuild()Z
ient/Licensing;->isInternetRequired()Z
ient/Licensing;->isReleaseBuild()Z
ient/Licensing;->setLicenseTimestamp(I)V
l/Rx;
bal/Rx:-><init>(J)V
```

```
Ke*y      *********Z*********
          123456789 Z ABCDEFG

ABCDEFG  --> A
A.len >= 2

ABCDEF G
  B    C
B  ---> 过期时间 ^ 0x56739ACD
C  ---> B的sum校验

123456789  ---> D
mId       int   int
           E     F
 lD       int   int
           G    H
long  int  int

H == F + 0x57145D56
G == E - 0x1B0CB11 + 0x55667788 & 0x7FFFFFFF
```

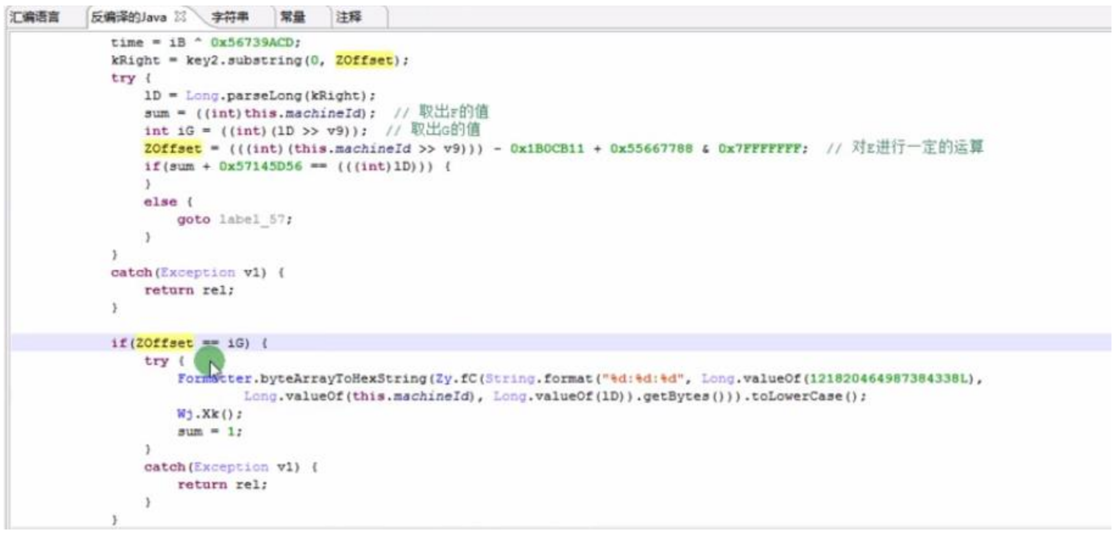


```
        int iB = 2;
        boolean rel = false;
        if(key != null && key.length() != 0) {
        int iB = 2;
        boolean rel = false;
        if(key != null && key.length() != 0) {
            String key2 = key.trim();  // 去除空格
            int ZOffset = key2.indexOf(0x5A);  // 寻找'z'的位置
            if(ZOffset < 0) {
                return rel;
            }

            String kRight = key2.substring(ZOffset + 1);  // 长度大于等于2
            if(kRight.length() < iB) {
                return rel;
            }

            String sB = kRight.substring(0, kRight.length() - 1);
            kRight = kRight.substring(kRight.length() - 1);
            try {
                iB = Integer.parseInt(sB);
                int iC = Integer.parseInt(kRight);
                sum = 0;
                int time;  // 校验
                for(time = iB; time > 0; time >>= 4) {
                    sum += time & 0xF;
                }
```

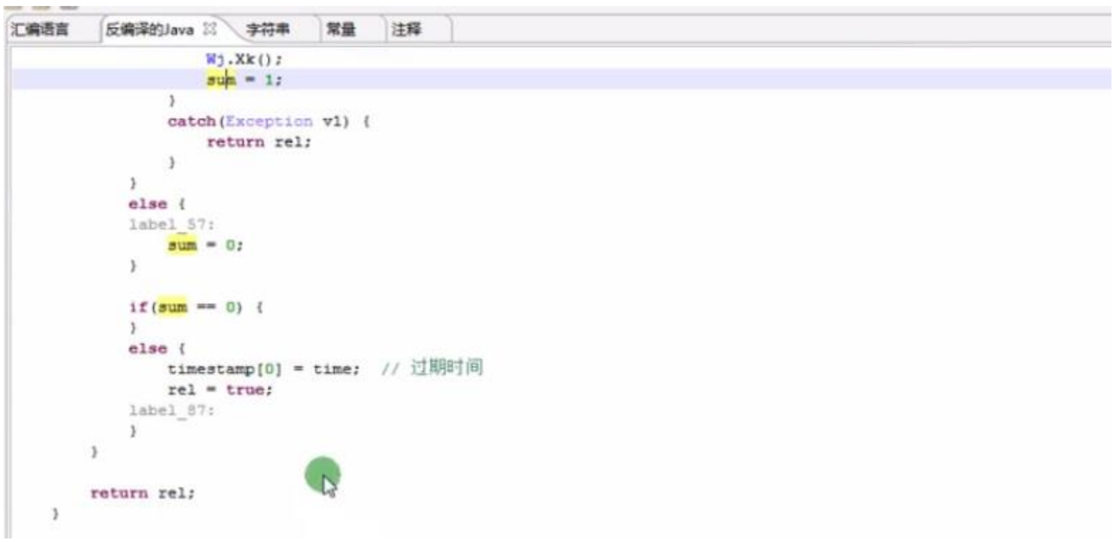


```java
            String sB = kRight.substring(0, kRight.length() - 1);
            kRight = kRight.substring(kRight.length() - 1);
            try {
                iB = Integer.parseInt(sB);
                int iC = Integer.parseInt(kRight);
                sum = 0;
                int time;  // 校验
                for(time = iB; time > 0; time >>= 4) {
                    sum += time & 0xF;
                }

                if(sum % 10 != iC) {
                    return rel;
                }
            }
            catch(Exception v1) {
                goto label_87;
            }

            time = iB ^ 0x56739ACD;
            kRight = key2.substring(0, ZOffset);
            try {
                v4_1 = Long.parseLong(kRight);
                sum = ((int)this.machineId);
                int v8 = ((int)(v4_1 >> v9));
                ZOffset = (((int)(this.machineId >> v9))) - 283
                if(sum + 1460952406 == (((int)v4_1))) {
                }
```

```
Ke*y      *********Z*********
          123456789 Z ABCDEFG

ABCDEFG  --> A
A.len >= 2

ABCDEF G
  B    C
B  ---> 过期时间 ^ 0x56739ACD
```



```java
            time = iB ^ 0x56739ACD;
            kRight = key2.substring(0, ZOffset);
            try {
                lD = Long.parseLong(kRight);
                sum = ((int)this.machineId);  // 取出F的值
                int iG = ((int)(lD >> v9));  // 取出G的值
                ZOffset = (((int)(this.machineId >> v9))) - 0x1B0CB11 + 0x55667788 & 0x7FFFFFFF;  // 对E进行一定的运算
                if(sum + 0x57145D56 == (((int)lD))) {
                }
                else {
                    goto label_57;
                }
            }
            catch(Exception v1) {
                return rel;
            }

            if(ZOffset == iG) {
                try {
                    Formatter.byteArrayToHexString(Zy.fC(String.format("%d:%d:%d", Long.valueOf(1218204649873843381L),
                            Long.valueOf(this.machineId), Long.valueOf(lD)).getBytes())).toLowerCase();
                    Wj.Xk();
                    sum = 1;
                }
                catch(Exception v1) {
                    return rel;
                }
            }
```



```java
                    Wj.Xk();
                    sum = 1;
                }
                catch(Exception v1) {
                    return rel;
                }
            }
            else {
            label_57:
                sum = 0;
            }

            if(sum == 0) {
            }
            else {
                timestamp[0] = time;  // 过期时间
                rel = true;
            label_87:
            }
        }

        return rel;
    }
```