Obfuscated GATE build instructions

1. 1stBarrier installation

1.1. Download 1stBarrier MINI edition from http://www.jproof.com/products/1stBarrierMINI.html

1.2. Unzip the JAR file

2. Obfuscate GATE classes

Follow the standard GATE build process and before making the gate.jar execute 1st Barrier obfuscator over the gate.* classes. It is important that you obfuscate **only** the gate.* classes and not any 3rd party classes.

2.1. Run the standard GATE build process (up to the "make jar" step):

```
$ ./configure

$ make clean

$ make depend

$ make
```

2.2. Create a temporary jar file for the gate. classes:*

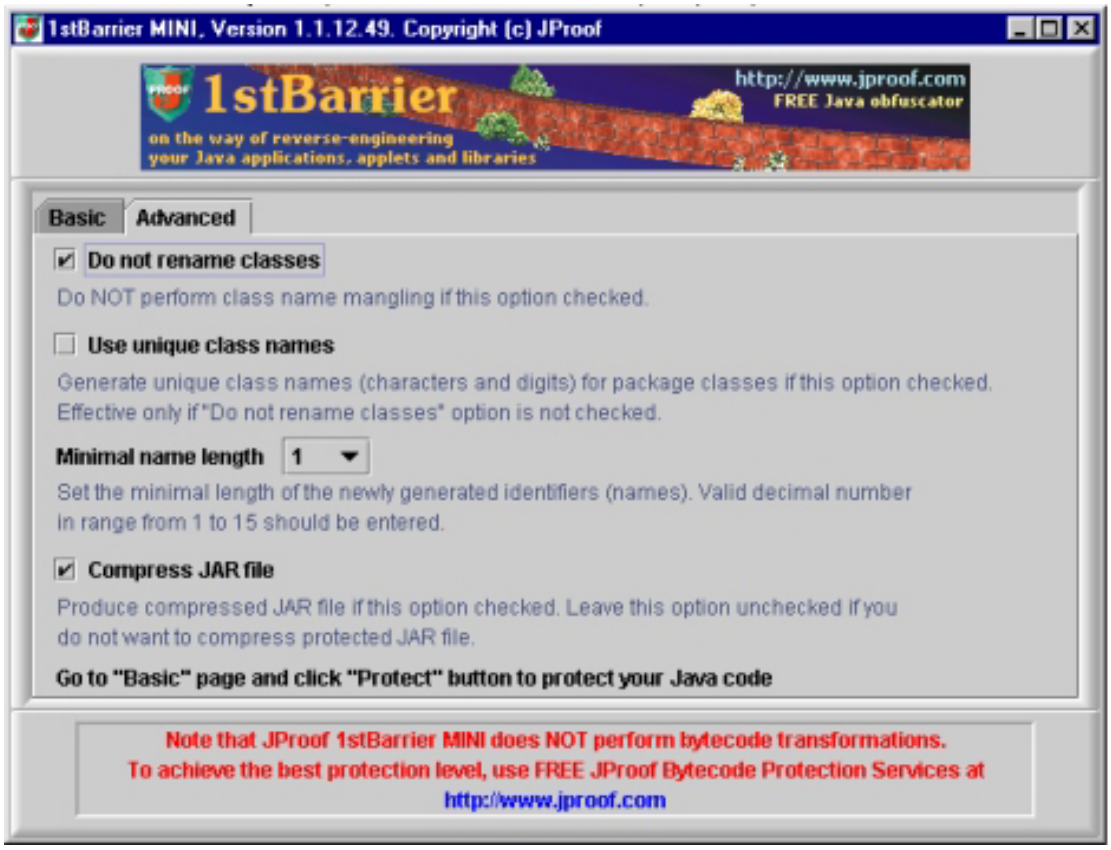
```
$ cd $GATE_HOME/classes

$ jar cvf gate_classes.jar  gate/*
```

2.3. Start the 1st Barrier obfuscator:

```
$ java - jar 1stBarrierMINI.jar
```

2.4. Set 1stBarrier options (see screenshot):

- In the *Advanced* tab check "Do **not** rename classes"
- In the *Advanced* tab check "Compress JAR file"
- In the *Basic* tab set the input jar (gate_classes.jar)



2.5. Run the obfuscator

2.6. Remove the temporary files (gate_classes.map, gate_classes.log, gate_classes.jar.old)

2.7. Replace the GATE_HOME/classes/gate directory classes with the obfuscated ones

```
$ rm $GATE_HOME/classes/gate

$ jar xvf gate_classes.jar
```

2.8. Continue the GATE build process:

```
$ make jar
```

2.9. Verify that GATE runs (either "make test" or with the GUI)

3. Sample obfuscated output

The following sample is the java source generated from a decompiler (such as Mocha) for the obfuscated gate.persist.OracleDataStore class. It is important that:

- Public methods are not obfuscated
- Public constants are not obfuscated
- Class names/packages are not renamed

```
/* Decompiled by Mocha from OracleDataStore.class */

package gate.persist;

import gate.*;
import gate.corpora.*;
import gate.security.*;
import gate.util.*;
import java.io.*;
import java.sql.*;
import java.util.*;
import oracle.jdbc.driver.*;
import oracle.sql.*;
import gate.annotation.DatabaseAnnotationSetImpl;
import gate.annotation.EventAwareAnnotationSet;
import gate.creole.ResourceInstantiationException;
import gate.event.DatastoreEvent;
import gate.event.DatastoreListener;
import java.net.URL;
import junit.framework.Assert;

public synchronized class OracleDataStore extends JDBCDataStore
{
    private static final String LThtIF = "GATE Oracle datastore";
    private static final String ltHtIF = "ora_ds.gif";
    private static final boolean LtHtIF = false;
    private static final int lTHtIF = 1;
    private static final int LTHtIF = 0;
    private static final int lthTIF = 10;
    private static final int LthTIF = 4000;
    private static final int lThTIF = 3;
    private static final int LThTIF = 1333;
    private static final int ltHTIF = 16384;

    public OracleDataStore()
    {
        datastoreComment = "GATE Oracle datastore";
        iconName = "ora_ds.gif";
    }

    public void setStorageUrl(String string)
```

```
        throws PersistenceException
    {
        super.setStorageUrl(string);
    }

…….


private Long LThtlf(Long long1, int i, String string1, Object object, int j,
CallableStatement callableStatement)
        throws PersistenceException
    {
        Long long2;
        long2 = null;
        try
        {
            callableStatement.setLong(1, long1.longValue());
            callableStatement.setLong(2, (long)i);
            callableStatement.setString(3, string1);
            callableStatement.setNull(4, 2);
            callableStatement.setNull(5, 12);
            callableStatement.setLong(6, (long)j);
            callableStatement.registerOutParameter(7, -5);
            switch (j)
            {
            case 103:
                boolean flag = ((Boolean)object).booleanValue();
                callableStatement.setLong(4, (flag != 0) ? 1L : 0L);
                break;

            case 101:
                callableStatement.setLong(4, (long)((Integer)object).intValue());
                break;

            case 102:
                callableStatement.setLong(4, ((Long)object).longValue());
                break;

            case 106:
                Double double = (Double)object;
                callableStatement.setDouble(4, double.doubleValue());
                break;

            case 104:
                String string2 = (String)object;
                if (lthTlf(string2))
                    callableStatement.setString(5, string2);
                break;

            case 100:
```

```
            case 105:
               break;

            default:
               throw new IllegalArgumentException("unsuppoeted feature type");
            }
            callableStatement.execute();
            long2 = new Long(callableStatement.getLong(7));
         }
         catch (SQLException e)
         {
            switch (e.getErrorCode())
            {
            case 20115:
               throw new PersistenceException("can't create feature [step
1],[invalid feature type] in DB: [" + e.getMessage() + "]");

            default:
               throw new PersistenceException("can't create feature [step 1] in
DB: [" + e.getMessage() + "]");
            }
         }
         return long2;
         pop local1
         throw local1;
         pop local2
         endfinalize local2
      }
```