

Report on JPF

Introduction

Java path finder is a model checking framework for java. This can be used as a standalone software, eclips or netbeans plugins or can be run with command line. I have used the command lines to run the model checks. Before that, I used the jpf-core source to build the necessary .jar files for the command. I used Windows Powershell to run all the commands.

Folder Structure

This section explains the folder structures.

1. Apps: This folder contains the apps that were used to run the model checking with jpf.
2. jpf-core: This folder contains the jpf core cloned from the jpf-core github repository.
 - i. bin: Contains the executables
 - ii. build: Contains the generated .jar files for jpf.
3. output: Some outputs were too large for the powershell so, for a better understanding I created text files containing the outputs of the apps while running model checking with jpf.
4. Screenshots: This folder contains all the screenshots of the project.
5. Root:
 - i. Report.docx: This is the Microsoft Word file that contains the report.
 - ii. README.md: This is the mark down version of the report to be read by github repository (For a simplified view).
 - iii. Report.pdf: pdf version of this report is also available here in this file.

Steps for model checking using JPF

The following steps were executed to run the simple java application.

1. [Clone JPS-core repository](#): I cloned the jpf-core repository inside my project folder from the jpf-core github repository by running the following command:

```
git clone https://github.com/javapathfinder/jpf-core.git
```
2. [Build Jpf-core](#): Then, I navigated into the folder "jpf-core". Now, I opened powershell and ran the following command to use gradle for building the .jar files:

```
./gradlew
```

This build the necessary .jar files for the jpf-core. Fig. 1 shows the output of the command.

```

PS D:\Projects\Heya\jpf\jpf-core> ./gradlew
> Task :compileJava
D:\Projects\Heya\jpf\jpf-core\src\main\gov\nasa\jpf\vm\HashedAllocationContext.java:21: warning: SharedSecrets is internal proprietary API and may be removed in a future release
import sun.misc.SharedSecrets;
D:\Projects\Heya\jpf\jpf-core\src\main\gov\nasa\jpf\vm\HashedAllocationContext.java:22: warning: JavalangAccess is internal proprietary API and may be removed in a future release
import sun.misc.JavalangAccess;
D:\Projects\Heya\jpf\jpf-core\src\main\gov\nasa\jpf\vm\HashedAllocationContext.java:85: warning: JavalangAccess is internal proprietary API and may be removed in a future release
static final JavalangAccess JLA = SharedSecrets.getJavalangAccess();
D:\Projects\Heya\jpf\jpf-core\src\main\gov\nasa\jpf\vm\HashedAllocationContext.java:85: warning: SharedSecrets is internal proprietary API and may be removed in a future release
static final JavalangAccess JLA = SharedSecrets.getJavalangAccess();
Note: D:\Projects\Heya\jpf\jpf-core\src\main\gov\nasa\jpf\vm\choice\PermutationCG.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
Note: Some input files use unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
4 warnings

> Task :compileClassesJava
D:\Projects\Heya\jpf\jpf-core\src\classes\java\lang\ClassLoader.java:29: warning: CompoundEnumeration is internal proprietary API and may be removed in a future release
import sun.misc.CompoundEnumeration;
D:\Projects\Heya\jpf\jpf-core\src\classes\java\lang\ClassLoader.java:114: warning: CompoundEnumeration is internal proprietary API and may be removed in a future release
return new CompoundEnumeration<URL>(resEnum);
D:\Projects\Heya\jpf\jpf-core\src\classes\sun\misc\JavaNetAccess.java:32: warning: URLClassPath is internal proprietary API and may be removed in a future release
URLClassPath getURLClassPath (URLClassLoader ucl);
D:\Projects\Heya\jpf\jpf-core\src\classes\sun\misc\SharedSecrets.java:52: warning: JavaUtilJarAccess is internal proprietary API and may be removed in a future release
private static JavaUtilJarAccess javaUtilJarAccess;
D:\Projects\Heya\jpf\jpf-core\src\classes\sun\misc\SharedSecrets.java:60: warning: JavaOISAccess is internal proprietary API and may be removed in a future release
private static JavaOISAccess javaOISAccess;
D:\Projects\Heya\jpf\jpf-core\src\classes\sun\misc\SharedSecrets.java:61: warning: JavaObjectInputStreamAccess is internal proprietary API and may be removed in a future release
private static JavaObjectInputStreamAccess javaObjectInputStreamAccess;
D:\Projects\Heya\jpf\jpf-core\src\classes\sun\misc\SharedSecrets.java:82: warning: JavaUtilJarAccess is internal proprietary API and may be removed in a future release
public static JavaUtilJarAccess javaUtilJarAccess() {
D:\Projects\Heya\jpf\jpf-core\src\classes\sun\misc\SharedSecrets.java:88: warning: JavaUtilJarAccess is internal proprietary API and may be removed in a future release
public static void setJavaUtilJarAccess(JavaUtilJarAccess access) {
D:\Projects\Heya\jpf\jpf-core\src\classes\sun\misc\SharedSecrets.java:142: warning: JavaObjectInputStreamAccess is internal proprietary API and may be removed in a future release
public static JavaObjectInputStreamAccess getJavaObjectInputStreamAccess() {
D:\Projects\Heya\jpf\jpf-core\src\classes\sun\misc\SharedSecrets.java:151: warning: JavaObjectInputStreamAccess is internal proprietary API and may be removed in a future release
public static void setJavaObjectInputStreamAccess(JavaObjectInputStreamAccess access) {
D:\Projects\Heya\jpf\jpf-core\src\classes\sun\misc\SharedSecrets.java:162: warning: JavaOISAccess is internal proprietary API and may be removed in a future release
public static void setJavaOISAccess(JavaOISAccess access) {
D:\Projects\Heya\jpf\jpf-core\src\classes\sun\misc\SharedSecrets.java:166: warning: JavaOISAccess is internal proprietary API and may be removed in a future release
public static JavaOISAccess getJavaOISAccess() {
12 warnings

> Task :compilePeersJava
D:\Projects\Heya\jpf\jpf-core\src\peers\gov\nasa\jpf\vm\JPF_java_util_Random.java:32: warning: Unsafe is internal proprietary API and may be removed in a future release
import sun.misc.Unsafe;
D:\Projects\Heya\jpf\jpf-core\src\peers\gov\nasa\jpf\vm\JPF_java_util_Random.java:93: warning: Unsafe is internal proprietary API and may be removed in a future release
private static Unsafe unsafe;
D:\Projects\Heya\jpf\jpf-core\src\peers\gov\nasa\jpf\vm\JPF_java_util_Random.java:99: warning: Unsafe is internal proprietary API and may be removed in a future release
Field singletonField = Unsafe.class.getDeclaredField("theUnsafe");
D:\Projects\Heya\jpf\jpf-core\src\peers\gov\nasa\jpf\vm\JPF_java_util_Random.java:101: warning: Unsafe is internal proprietary API and may be removed in a future release
unsafe = (Unsafe)singletonField.get(null);
4 warnings

> Task :compileTestJava
D:\Projects\Heya\jpf\jpf-core\src\tests\gov\nasa\jpf\test\vm\reflection\ReflectionTest.java:34: warning: Reflection is internal proprietary API and may be removed in a future release
Class<T> callerCls = sun.reflect.Reflection.getCallerClass(0); // that would be getCallerClass()
D:\Projects\Heya\jpf\jpf-core\src\tests\gov\nasa\jpf\test\vm\reflection\ReflectionTest.java:38: warning: Reflection is internal proprietary API and may be removed in a future release
callerCls = sun.reflect.Reflection.getCallerClass(1); // foo()
D:\Projects\Heya\jpf\jpf-core\src\tests\gov\nasa\jpf\test\vm\reflection\ReflectionTest.java:42: warning: Reflection is internal proprietary API and may be removed in a future release
callerCls = sun.reflect.Reflection.getCallerClass(2); // bar()
D:\Projects\Heya\jpf\jpf-core\src\tests\gov\nasa\jpf\test\vm\reflection\ReflectionTest.java:46: warning: Reflection is internal proprietary API and may be removed in a future release
callerCls = sun.reflect.Reflection.getCallerClass(3); // callIt()
Note: D:\Projects\Heya\jpf\jpf-core\src\tests\gov\nasa\jpf\test\vm\reflection\ReflectionTest.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
Note: Some input files use unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
4 warnings

BUILD SUCCESSFUL in 1m 27s
16 actionable tasks: 16 executed
PS D:\Projects\Heya\jpf\jpf-core> ./gradlew

```

Fig 1: Building necessary .jar files

I also ran the following command to build again and this time I stored the outputs in a text file in the output folder mentioned in the folder structure.

```
./gradlew > ../output/jpfbuild.txt
```

3. **Creating example java application “Rand.java”:** I created the following files in the project folder.

Rand.java:

```

import java.util.Random;

public class Rand {
    public static void main (String[] args) {
        System.out.println("computing c = a/(b+a - 2) ..");
        Random random = new Random(42); // (1)
    }
}

```

```

    int a = random.nextInt(2); // (2)
    System.out.printf("a=%d\n", a);

    //... lots of code here

    int b = random.nextInt(3); // (3)
    System.out.printf("  b=%d      ,a=%d\n", b, a);

    int c = a/(b+a -2); // (4)
    System.out.printf("=>  c=%d      , b=%d, a=%d\n", c, b, a);
}
}

```

Rand.jpf:

```

target = Rand

cg.enumerate_random = true
report.console.property_violation=error,trace

```

4. Running the model check:

I ran the following command to run the model check.

```

../jpf-core/bin/jpf +cg.enumerate_random=true Rand

```

The output of the model check can be seen the screenshot in figure 2. I also wanted to store the output of the command in a file for a better understanding so, I again ran the following command.

```

../jpf-core/bin/jpf +cg.enumerate_random=true Rand >
..\output\randoutput.txt

```

This created a randoutput.txt file with the powershell outputs.

```
Windows PowerShell
Tasnia Ashrafi Heya
PS D:\Projects\Heya\jpf\Apps> ../jpf-core/bin/jpf +cg.enumerate_random=true Rand
D:\Projects\Heya\jpf\Apps>REM
D:\Projects\Heya\jpf\Apps>REM overly simplified batch file to start JPF from a command prompt
D:\Projects\Heya\jpf\Apps>REM
JavaPathfinder core system v8.0 (rev 26e11d1de726c19ba8ae10551e048ec0823aabc6) - (C) 2005-2014 United States Government. All rights reserved.

===== system under test
Rand.main()
===== search started: 12/5/19 2:45 PM
computing c = a/(b+a - 2)..
a=0
  b=0      ,a=0
=> c=0     ,b=0, a=0
  b=1      ,a=0
=> c=0     ,b=1, a=0
  b=2      ,a=0
===== error 1
gov.nasa.jpf.vm.NoUncaughtExceptionsProperty
java.lang.ArithmeticException: division by zero
    at Rand.main(Rand.java:34)
===== snapshot #1
thread java.lang.Thread: {id:0,name:main,status:RUNNING,priority:5,isDaemon:false,lockCount:0,suspendCount:0}
call stack:
    at Rand.main(Rand.java:34)
===== results
error #1: gov.nasa.jpf.vm.NoUncaughtExceptionsProperty "java.lang.ArithmeticException: division by zero a..."
===== statistics
elapsed time: 00:00:01
states: new=4,visited=1,backtracked=2,end=2
search: maxDepth=3,constraints=0
choice generators: thread=1 (signal=0,lock=1,sharedRef=0,threadApi=0,reschedule=0), data=2
heap: new=653,released=27,maxLive=619,gcCycles=4
instructions: 3365
max memory: 243MB
loaded code: classes=66,methods=1371
===== search finished: 12/5/19 2:45 PM
PS D:\Projects\Heya\jpf\Apps>
```

Figure 2: Running model check on Rand.java

5. Example2 App:

I also ran another example. The source code of the file is large to put here. It is under “Apps” folder with the name “RobotManager.java”. A related jpf file was also created “RobotManager.jpf”.

6. Example2 App model check:

I ran the second example with the following command.

```
../jpf-core/bin/jpf +cg.enumerate_random=true RobotManager
```

The output of the command can be observed in figure 3. I also ran the following command to store the output in a text file by running the following command.

```
../jpf-core/bin/jpf +cg.enumerate_random=true RobotManager >
..\output\robotmanageroutput.txt
```

The output file is available in the “output” folder mentioned in the folder structure. The name of the output file is “robotmanageroutput.txt”.

