

# Summary

**Sink States:** $0(0 \times 10^0)$

Table 1: Pulse Analysis Summary

Classes	Methods	States	Unsatisfiable Clauses	Unreachable States	Possible concurrent Methods	Total. no. of pairs	No. of concurrent pairs	Percentage of concurrent Methods
JGFCryptBenchSizeA	2	1	0	0	1	3	1	33
JGFInstrumentor	3	1	0	0	2	6	2	33
JGFCryptBench	6	1	0	0	5	21	7	33
IDEATest	9	1	0	0	8	45	21	47
JGFTimer	3	1	0	0	2	6	2	33
Total Classes=5	23	5	0	0	18	81	33	41

## Contents

<b>1</b>	<b>JGFCryptBenchSizeA</b>	<b>3</b>
<b>2</b>	<b>JGFInstrumentor</b>	<b>4</b>
<b>3</b>	<b>JGFCryptBench</b>	<b>5</b>
<b>4</b>	<b>IDEATest</b>	<b>6</b>
<b>5</b>	<b>JGFTimer</b>	<b>7</b>
<b>6</b>	<b>Abbreviation</b>	<b>8</b>
<b>7</b>	<b>Annotated Version of Sequential Java Program generated by Sip4j</b>	<b>9</b>

## 1 JGFCryptBenchSizeA

Table 2: Methods Requires Clause Satisfiability

Method	Satisfiability
JGFCryptBenchSizeA	✓
main	✓

Table 3: State Transition Matrix

	alive
alive	↑

Table 4: Methods Concurrency Matrix

	JGFCryptBenchSizeA	main
JGFCryptBenchSizeA	⌘	⌘
main	⌘	

## 2 JGFInstrumentor

Table 5: Methods Requires Clause Satisfiability

Method	Satisfiability
JGFInstrumentor	✓
printHeader	✓
printTimer	✓

Table 6: State Transition Matrix

	alive
alive	↑

Table 7: Methods Concurrency Matrix

	JGFInstrumentor	printHeader	printTimer
JGFInstrumentor	⌘	⌘	⌘
printHeader	⌘		
printTimer	⌘		⌘

### 3 JGFCryptBench

Table 8: Methods Requires Clause Satisfiability

Method	Satisfiability
JGFCryptBench	✓
JGFrun	✓
JGFinitialise	✓
JGFvalidate	✓
JGFtidyup	✓
JGFsetsize	✓

Table 9: State Transition Matrix

	alive
alive	↑

Table 10: Methods Concurrency Matrix

	JGFCryptBench	JGFrun	JGFinitialise	JGFvalidate	JGFtidyup	JGFsetsize
JGFCryptBench	⌘	⌘	⌘	⌘	⌘	⌘
JGFrun	⌘	⌘	⌘	⌘	⌘	⌘
JGFinitialise	⌘	⌘	⌘	⌘	⌘	⌘
JGFvalidate	⌘	⌘	⌘	⌘	⌘	⌘
JGFtidyup	⌘	⌘	⌘	⌘	⌘	⌘
JGFsetsize	⌘	⌘	⌘	⌘	⌘	⌘

## 4 IDEATest

Table 11: Methods Requires Clause Satisfiability

Method	Satisfiability
IDEATest	✓
buildTestData	✓
calcEncryptKey	✓
calcDecryptKey	✓
inv	✓
Do	✓
cipheridea	✓
mul	✓
freeTestData	✓

Table 12: State Transition Matrix

	alive
alive	↑

Table 13: Methods Concurrency Matrix

	IDEATest	buildTestData	calcEncryptKey	calcDecryptKey	inv	Do	cipheridea	mul	freeTestData
IDEATest	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
buildTestData	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
calcEncryptKey	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
calcDecryptKey	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
inv	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
Do	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
cipheridea	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
mul	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
freeTestData	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘

## 5 JGFTimer

Table 14: Methods Requires Clause Satisfiability

Method	Satisfiability
JGFTimer	✓
print	✓
perf	✓

Table 15: State Transition Matrix

	alive
alive	↑

Table 16: Methods Concurrency Matrix

	JGFTimer	print	perf
JGFTimer	⌈	⌈	⌈
print	⌈	⌈	
perf	⌈		

## 6 Abbreviation

Table 17: Used Abbreviation

Symbol	Meaning
✓	requires clause of the method is satisfiable
✗	requires clause of the method is unsatisfiable
↑	The row-state can be transitioned to the column-state
✕	The row-state cannot be transitioned to the column-state
	The row-method can be possibly executed parallel with the column-method
⋈	The row-method cannot be executed parallel with the column-method



## 7 Annotated Version of Sequential Java Program generated by Sip4j

```
1 package outputs;
2 import edu.cmu.cs.plural.annot.*;
3
4 @ClassStates({@State(name = "alive")})
5 class JGFCryptBenchSizeA {
6   @Perm(ensures="unique(this) in alive")
7   JGFCryptBenchSizeA() { }
8
9   @Perm(requires="none(this) in alive",
10  ensures="unique(this) in alive")
11   void main(String argv[]) {
12   }
13
14 }ENDOFCLASS
15
16 @ClassStates({@State(name = "alive")})
17
18 class JGFInstrumentor {
19   @Perm(ensures="unique(this) in alive")
20   JGFInstrumentor() { }
21
22   void printHeader(int section, int size) {
23   }
24   @Perm(requires="full(this) in alive",
25  ensures="full(this) in alive")
26   void printTimer(String name) {
27   }
28
29 }ENDOFCLASS
30
31 @ClassStates({@State(name = "alive")})
32
33 class JGFCryptBench {
34   @Perm(ensures="unique(this) in alive")
35   JGFCryptBench() { }
36
37   @Perm(requires="unique(this) in alive",
38  ensures="unique(this) in alive")
39   public void JGFrun(int size) {
40   }
41   @Perm(requires="unique(this) in alive",
42  ensures="unique(this) in alive")
43   public void JGFinitialise() {
44   }
45   @Perm(requires="pure(this) in alive",
46  ensures="pure(this) in alive")
47   public void JGFvalidate() {
48   }
49
50   public void JGFtidyup() {
51   }
52   @Perm(requires="full(this) in alive",
53  ensures="full(this) in alive")
54   public void JGFsetsize(int size) {
55   }
56
57 }ENDOFCLASS
58
59 @ClassStates({@State(name = "alive")})
60
61 class IDEATest {
62   @Perm(ensures="unique(this) in alive")
63   IDEATest() { }
64
65   @Perm(requires="unique(this) in alive",
66  ensures="unique(this) in alive")
67   void buildTestData() {
68   }
69   @Perm(requires="full(this) in alive",
70  ensures="full(this) in alive")
71   private void calcEncryptKey() {
72   }
73   @Perm(requires="full(this) in alive",
74  ensures="full(this) in alive")
75 }
```

```

76 private void calcDecryptKey() {
77 }

79 private int inv(int x) {
80     return 0;
81 }
82 @Perm(requires="full(this) in alive",
83 ensures="full(this) in alive")
84 public void Do() {
85 }
86 @Perm(requires="full(#0) * full(#1) * full(#2) in alive",
87 ensures="full(#0) * full(#1) * full(#2) in alive")
88 private void cipheridea(byte[] text1, byte[] text2, int[] key) {
89 }

91 private int mul(int a, int b) {
92     return 0;
93 }
94 @Perm(requires="unique(this) in alive",
95 ensures="unique(this) in alive")
96 void freeTestData() {
97 }

99 }ENDOFCLASS

101 @ClassStates({@State(name = "alive")})

103 class JGFTimer {
104     @Perm(ensures="unique(this) in alive")
105     JGFTimer() { }

107     @Perm(requires="full(this) in alive",
108     ensures="full(this) in alive")
109     public void print() {
110     }
111     @Perm(requires="pure(this) in alive",
112     ensures="pure(this) in alive")
113     public double perf() {
114         return 0;
115     }

117 }ENDOFCLASS

```