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
ArrayCollection	8	1	0	0	4	36	10	28
ObjectClass	2	1	0	0	0	3	0	0
Client	2	1	0	0	0	3	0	0
Total Classes=3	12	3	0	0	4	42	10	24

Contents

1 ArrayCollection

Table 2: Methods Requires Clause Satisfiability

Method	Satisfiability
ArrayCollection	√
createColl	√
printColl	√
computeStat	√
isSorted	√
findMax	√
incrColl	√
tidyupColls	√

Table 3: State Transition Matrix

	alive
alive	↑

Table 4: Methods Concurrency Matrix

	ArrayCollection	createColl	printColl	computeStat	isSorted	findMax	incrColl	tidyupColls
ArrayCollection	∦	∦	∦	∦	∦	∦	∦	∦
createColl	∦	∦	∦	∦	∦	∦	∦	∦
printColl	∦	∦	‖	‖	‖	‖	∦	∦
computeStat	∦	∦	‖	‖	‖	‖	∦	∦
isSorted	∦	∦	‖	‖	‖	‖	∦	∦
findMax	∦	∦	‖	‖	‖	‖	∦	∦
incrColl	∦	∦	∦	∦	∦	∦	∦	∦
tidyupColls	∦	∦	∦	∦	∦	∦	∦	∦

2 ObjectClass

Table 5: Methods Requires Clause Satisfiability

Method	Satisfiability
ObjectClass	√
manipulateObjects	√

Table 6: State Transition Matrix

	alive
alive	↑

Table 7: Methods Concurrency Matrix

	ObjectClass	manipulateObjects
ObjectClass	∦	∦
manipulateObjects	∦	∦

3 Client

Table 8: Methods Requires Clause Satisfiability

Method	Satisfiability
Client	√
main	√

Table 9: State Transition Matrix

	alive
alive	↑

Table 10: Methods Concurrency Matrix

	Client	main
Client	∦	∦
main	∦	∦

4 Abbreviation

Table 11: Used Abbreviation

Symbol	Meaning
$\surd$	requires clause of the method is satisfiable
$\times$	requires clause of the method is unsatisfiable
$\uparrow$	The row-state can be transitioned to the column-state
$\times$	The row-state cannot be transitioned to the column-state
$\parallel$	The row-method can be possibly executed parallel with the column-method
$\nparallel$	The row-method cannot be executed parallel with the column-method

5 Annotated Version of Sequential Java Program generated by Sip4j

```
1  package outputs;
2  import edu.cmu.cs.plural.annot.*;

4  @ClassStates({@State(name = "alive")})
5  class ArrayCollection {
6  @Perm(ensures="unique(this) in alive")
7  ArrayCollection() {   }

9  @Perm(requires="unique(this) in alive",
10 ensures="unique(this) in alive")
11 public void createColl(Integer[] coll) {
12 }
13 @Perm(requires="pure(this) in alive",
14 ensures="pure(this) in alive")
15 public void printColl(Integer[] coll) {
16 }
17 @Perm(requires="pure(this) in alive",
18 ensures="pure(this) in alive")
19 public void computeStat(Integer[] coll) {
20 }
21 @Perm(requires="pure(this) in alive",
22 ensures="pure(this) in alive")
23  boolean isSorted(Integer[] coll) {
24  return 0;
25 }
26 @Perm(requires="pure(this) in alive",
27 ensures="pure(this) in alive")
28  Integer findMax(Integer[] coll) {
29  return null;
30 }
31 @Perm(requires="full(this) in alive",
32 ensures="full(this) in alive")
33 public void incrColl(Integer[] coll) {
34 }
35 @Perm(requires="unique(this) in alive",
36 ensures="unique(this) in alive")
37 public void tidyupColls(Integer[] coll) {
38 }

40 }ENDOFCLASS

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

44 class ObjectClass {
45 @Perm(ensures="unique(this) in alive")
46 ObjectClass() {   }

48 @Perm(requires="full(this) in alive",
49 ensures="full(this) in alive")
50  void manipulateObjects(Client p1, Client p2) {
51 }

53 }ENDOFCLASS

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

57 class Client {
58 @Perm(ensures="unique(this) in alive")
59 Client() {   }

61 @Perm(requires="unique(this) in alive",
62 ensures="unique(this) in alive")
63  void main(String[] args) {
64 }

66 }ENDOFCLASS
```