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	7	1	0	0	6	28	13	46
ArrayClient	2	1	0	0	0	3	0	0
Total Classes=2	9	2	0	0	6	31	13	42

Contents

1 ArrayCollection

Table 2: Methods Requires Clause Satisfiability

Method	Satisfiability
ArrayCollection	$\surd$
initColl	$\surd$
displayColl	$\surd$
displayE	$\surd$
initE	$\surd$
copyColl	$\surd$
tidyupColl	$\surd$

Table 3: State Transition Matrix

	alive
alive	$\uparrow$

Table 4: Methods Concurrency Matrix

	ArrayCollection	initColl	displayColl	displayE	initE	copyColl	tidyupColl
ArrayCollection	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
initColl	$\nparallel$	$\nparallel$	$\parallel$	$\parallel$	$\parallel$	$\nparallel$	$\nparallel$
displayColl	$\nparallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\nparallel$
displayE	$\nparallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$
initE	$\nparallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\nparallel$
copyColl	$\nparallel$	$\nparallel$	$\parallel$	$\parallel$	$\parallel$	$\nparallel$	$\nparallel$
tidyupColl	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$	$\nparallel$	$\nparallel$	$\nparallel$

2 ArrayClient

Table 5: Methods Requires Clause Satisfiability

Method	Satisfiability
ArrayClient	√
main	√

Table 6: State Transition Matrix

	alive
alive	↑

Table 7: Methods Concurrency Matrix

	ArrayClient	main
ArrayClient	∦	∦
main	∦	∦

3 Abbreviation

Table 8: 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

4 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="full(this) in alive",
10 ensures="full(this) in alive")
11 public void initColl() {
12 }
13 @Perm(requires="pure(this) in alive",
14 ensures="pure(this) in alive")
15 public void displayColl() {
16 }
17 @Perm(requires="pure(#0) in alive",
18 ensures="pure(#0) in alive")
19 public void displayE(Integer[] e) {
20 }
21 @Perm(requires="pure(this) * full(#0) in alive",
22 ensures="pure(this) * full(#0) in alive")
23 public void initE(Integer[] e1) {
24 }
25 @Perm(requires="full(this) in alive",
26 ensures="full(this) in alive")
27 public void copyColl() {
28 }
29 @Perm(requires="unique(this) in alive",
30 ensures="unique(this) in alive")
31 public void tidyupColl() {
32 }

34 }ENDOFCLASS

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

38 class ArrayClient {
39 @Perm(ensures="unique(this) in alive")
40 ArrayClient() {   }

42 @Perm(requires="unique(this) in alive",
43 ensures="unique(this) in alive")
44  void main(String[] args) {
45 }

47 }ENDOFCLASS
```