Summary

Sink States:0(0×10^0)

Table 1: Sip4J Analysis Summary

Classes	Methods	States	Unreachable clauses	Unreachable states	Possible concurrent methods	Total. no. of method pairs	No. of concurrent method pairs	Percentage of concurrent methods pairs
JGFMonteCarloBenchSizeA	2	1	0	0	0	3	0	0
JGFMonteCarloBench	7	1	0	0	1	28	1	4
CallAppDemo	4	1	0	0	0	10	0	0
AppDemo	18	1	0	0	6	171	21	12
Universal	11	1	0	0	7	66	28	42
PathId	11	1	0	0	5	66	15	23
RatePath	11	1	0	0	4	66	10	15
ReturnPath	23	1	0	0	9	276	45	16
MonteCarloPath	21	1	0	0	20	231	56	24
ToInitAllTasks	21	1	0	0	10	231	55	24
ToResult	14	1	0	0	7	105	28	27
PriceStock	5	1	0	0	1	15	1	7
ToTask	5	1	0	0	2	15	3	20
DemoException	1	1	0	0	0	1	0	0
JGFInstrumentor	13	1	0	0	12	91	12	13
JGFTimer	9	1	0	0	3	45	6	13
test	4	1	0	0	1	10	1	10
Utilities	5	1	0	0	4	15	8	53
Total Classes=18	185	18	0	0	92	1445	290	20

Contents

1 JGFMonteCarloBenchSizeA

Table 2: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
JGFMonteCarloBenchSizeA	$\surd$
main	$\surd$

Table 3: State Transition Matrix

	alive
alive	$\uparrow$

Table 4: Methods Concurrency Matrix

	JGFMonteCarloBenchSizeA	main
JGFMonteCarloBenchSizeA	$\nparallel$	$\nparallel$
main	$\nparallel$	$\nparallel$

2 JGFMonteCarloBench

Table 5: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
JGFMonteCarloBench	√
JGFrun	√
JGFsetsize	√
JGFinitialise	√
JGFapplication	√
JGFvalidate	√
JGFtidyup	√

Table 6: State Transition Matrix

	alive
alive	↑

Table 7: Methods Concurrency Matrix

	JGFMonteCarloBench	JGFrun	JGFsetsize	JGFinitialise	JGFapplication	JGFvalidate	JGFtidyup
JGFMonteCarloBench	∦	∦	∦	∦	∦	∦	∦
JGFrun	∦	∦	∦	∦	∦	∦	∦
JGFsetsize	∦	∦	∦	∦	∦	∦	∦
JGFinitialise	∦	∦	∦	∦	∦	∦	∦
JGFapplication	∦	∦	∦	∦	∦	∦	∦
JGFvalidate	∦	∦	∦	∦	∦	‖	∦
JGFtidyup	∦	∦	∦	∦	∦	∦	∦

3 CallAppDemo

Table 8: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
CallAppDemo	$\surd$
initialise	$\surd$
presults	$\surd$
runiters	$\surd$

Table 9: State Transition Matrix

	alive
alive	$\uparrow$

Table 10: Methods Concurrency Matrix

	CallAppDemo	initialise	presults	runiters
CallAppDemo	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
initialise	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
presults	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
runiters	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$

4 AppDemo

Table 11: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
AppDemo	√
initSerial	√
initTasks	√
processSerial	√
processResults	√
runSerial	√
getdataDirname	√
setdataDirname	√
getdataFilename	√
setdataFilename	√
getnTimeStepsMC	√
setnTimeStepsMC	√
getnRunsMC	√
setnRunsMC	√
gettasks	√
settasks	√
getresults	√
setresults	√

Table 12: State Transition Matrix

	alive
alive	↑

Table 13: Methods Concurrency Matrix

	AppDemo	initSerial	initTasks	processSerial	processResults	runSerial	getdataDirname	setdataDirname	getdataFilename	setdataFilename	getnTimeStepsMC	setnTimeStepsMC	getnRunsMC	setnRunsMC	gettasks	settasks	getresults	setresults
AppDemo	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
initSerial	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
initTasks	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
processSerial	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
processResults	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
runSerial	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
getdataDirname	∦	∦	∦	∦	∦	∦	∥	∦	∥	∦	∥	∦	∥	∦	∥	∦	∥	∦
setdataDirname	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
getdataFilename	∦	∦	∦	∦	∦	∦	∥	∦	∥	∦	∥	∦	∥	∦	∥	∦	∥	∦
setdataFilename	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
getnTimeStepsMC	∦	∦	∦	∦	∦	∦	∥	∦	∥	∦	∥	∦	∥	∦	∥	∦	∥	∦
setnTimeStepsMC	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦

getnRunsMC	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$	$\nparallel$	$\parallel$	$\nparallel$	$\parallel$	$\nparallel$	$\parallel$	$\nparallel$	$\parallel$	$\nparallel$	$\parallel$	$\nparallel$
setnRunsMC	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
gettasks	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$	$\nparallel$	$\parallel$	$\nparallel$	$\parallel$	$\nparallel$	$\parallel$	$\nparallel$	$\parallel$	$\nparallel$	$\parallel$	$\nparallel$
settasks	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
getresults	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$	$\nparallel$	$\parallel$	$\nparallel$	$\parallel$	$\nparallel$	$\parallel$	$\nparallel$	$\parallel$	$\nparallel$	$\parallel$	$\nparallel$
setresults	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$

5 Universal

Table 14: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
Universal	√
setprompt	√
setDEBUG	√
dbgPrintln	√
errPrintln	√
getDEBUG	√
getUNIVERSALDEBUG	√
setUNIVERSALDEBUG	√
getprompt	√
dbgPrint	√
errPrint	√

Table 15: State Transition Matrix

	alive
alive	↑

Table 16: Methods Concurrency Matrix

	Universal	setprompt	setDEBUG	dbgPrintln	errPrintln	getDEBUG	getUNIVERSALDEBUG	setUNIVERSALDEBUG	getprompt	dbgPrint	errPrint
Universal	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
setprompt	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
setDEBUG	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
dbgPrintln	∦	∦	∦	∥	∥	∥	∥	∦	∥	∥	∥
errPrintln	∦	∦	∦	∥	∥	∥	∥	∦	∥	∥	∥
getDEBUG	∦	∦	∦	∥	∥	∥	∥	∦	∥	∥	∥
getUNIVERSALDEBUG	∦	∦	∦	∥	∥	∥	∥	∦	∥	∥	∥
setUNIVERSALDEBUG	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
getprompt	∦	∦	∦	∥	∥	∥	∥	∦	∥	∥	∥
dbgPrint	∦	∦	∦	∥	∥	∥	∥	∦	∥	∥	∥
errPrint	∦	∦	∦	∥	∥	∥	∥	∦	∥	∥	∥

6 PathId

Table 17: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
PathId	$\surd$
dbgDumpFields	$\surd$
copyInstanceVariables	$\surd$
getdTime	$\surd$
getname	$\surd$
getstartDate	$\surd$
getendDate	$\surd$
setname	$\surd$
setstartDate	$\surd$
setendDate	$\surd$
setdTime	$\surd$

Table 18: State Transition Matrix

	alive
alive	$\uparrow$

Table 19: Methods Concurrency Matrix

	PathId	dbgDumpFields	copyInstanceVariables	getdTime	getname	getstartDate	getendDate	setname	setstartDate	setendDate	setdTime
PathId	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
dbgDumpFields	$\nparallel$	$\parallel$	$\nparallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
copyInstanceVariables	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
getdTime	$\nparallel$	$\parallel$	$\nparallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
getname	$\nparallel$	$\parallel$	$\nparallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
getstartDate	$\nparallel$	$\parallel$	$\nparallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
getendDate	$\nparallel$	$\parallel$	$\nparallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
setname	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
setstartDate	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
setendDate	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
setdTime	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$

7 RatePath

Table 20: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
RatePath	√
getReturnCompounded	√
getReturnNonCompounded	√
readRatesFile	√
getEndPathValue	√
getPathValue	√
incpathValue	√
getpathValue	√
setpathValue	√
getpathDate	√
setpathDate	√

Table 21: State Transition Matrix

	alive
alive	↑

Table 22: Methods Concurrency Matrix

	RatePath	getReturnCompounded	getReturnNonCompounded	readRatesFile	getEndPathValue	getPathValue	incpathValue	getpathValue	setpathValue	getpathDate	setpathDate
RatePath	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
getReturnCompounded	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
getReturnNonCompounded	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
readRatesFile	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
getEndPathValue	∦	∦	∦	∦	∥	∥	∦	∥	∦	∥	∦
getPathValue	∦	∦	∦	∦	∥	∥	∦	∥	∦	∥	∦
incpathValue	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
getpathValue	∦	∦	∦	∦	∥	∥	∦	∥	∦	∥	∦
setpathValue	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
getpathDate	∦	∦	∦	∦	∥	∥	∦	∥	∦	∥	∦
setpathDate	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦

8 ReturnPath

Table 23: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
ReturnPath	√
estimatePath	√
computeMean	√
computeVariance	√
computeExpectedReturnRate	√
computeVolatility	√
dbgDumpFields	√
getexpectedReturnRate	√
getvolatility	√
getreturnDefinition	√
getvolatility2	√
getpathValue	√
setpathValue	√
getnPathValue	√
setnPathValue	√
setreturnDefinition	√
setexpectedReturnRate	√
setvolatility	√
setvolatility2	√
getmean	√
setmean	√
getvariance	√
setvariance	√

Table 24: State Transition Matrix

	alive
alive	↑

Table 25: Methods Concurrency Matrix

	ReturnPath	estimatePath	computeMean	computeVariance	computeExpectedReturnRate	computeVolatility	dbgDumpFields	getexpectedReturnRate	getvolatility	getreturnDefinition	getvolatility2	getpathValue	setpathValue	getnPathValue	setnPathValue	setreturnDefinition	setexpectedReturnRate	setvolatility	setvolatility2	getmean
ReturnPath	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
estimatePath	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
computeMean	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦

computeVariance	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
computeExpectedReturnRate	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
computeVolatility	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
dbgDumpFields	∦	∦	∦	∦	∦	∦	∥	∥	∥	∥	∥	∥	∦	∥	∦	∦	∦	∦	∦	∥
getexpectedReturnRate	∦	∦	∦	∦	∦	∦	∥	∥	∥	∥	∥	∥	∦	∥	∦	∦	∦	∦	∦	∥
getvolatility	∦	∦	∦	∦	∦	∦	∥	∥	∥	∥	∥	∥	∦	∥	∦	∦	∦	∦	∦	∥
getreturnDefinition	∦	∦	∦	∦	∦	∦	∥	∥	∥	∥	∥	∥	∦	∥	∦	∦	∦	∦	∦	∥
getvolatility2	∦	∦	∦	∦	∦	∦	∥	∥	∥	∥	∥	∥	∦	∥	∦	∦	∦	∦	∦	∥
getpathValue	∦	∦	∦	∦	∦	∦	∥	∥	∥	∥	∥	∥	∦	∥	∦	∦	∦	∦	∦	∥
setpathValue	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
getnPathValue	∦	∦	∦	∦	∦	∦	∥	∥	∥	∥	∥	∥	∦	∥	∦	∦	∦	∦	∦	∥
setnPathValue	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
setreturnDefinition	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
setexpectedReturnRate	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
setvolatility	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
setvolatility2	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
getmean	∦	∦	∦	∦	∦	∦	∥	∥	∥	∥	∥	∥	∦	∥	∦	∦	∦	∦	∦	∥
setmean	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
getvariance	∦	∦	∦	∦	∦	∦	∥	∥	∥	∥	∥	∥	∦	∥	∦	∦	∦	∦	∦	∥
setvariance	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦

9 MonteCarloPath

Table 26: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
MonteCarloPath	√
copyInstanceVariables	√
setreturnDefinition	√
setexpectedReturnRate	√
setvolatility	√
setnTimeSteps	√
setpathStartValue	√
setpathValue	√
setfluctuations	√
computeFluctuationsGaussian	√
computePathValue	√
getpathValue	√
getnTimeSteps	√
getfluctuations	√
getreturnDefinition	√
getexpectedReturnRate	√
getvolatility	√
getpathStartValue	√
writeFile	√
getRatePath	√
computeFluctuationsGaussianOverload	√

Table 27: State Transition Matrix

	alive
alive	↑

Table 28: Methods Concurrency Matrix

	MonteCarloPath	copyInstanceVariables	setreturnDefinition	setexpectedReturnRate	setvolatility	setnTimeSteps	setpathStartValue	setpathValue	setfluctuations	computeFluctuationsGaussian	computePathValue	getpathValue	getnTimeSteps	getfluctuations	getreturnDefinition	getexpectedReturnRate	getvolatility	getpathStartValue
MonteCarloPath	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
copyInstanceVariables	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦

setreturnDefinition	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
setexpectedReturnRate	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
setvolatility	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
setnTimeSteps	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
setpathStartValue	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
setpathValue	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
setfluctuations	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
computeFluctuationsGaussian	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
computePathValue	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
getpathValue	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∥	∥	∥	∥	∥	∥	∥
getnTimeSteps	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∥	∥	∥	∥	∥	∥	∥
getfluctuations	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∥	∥	∥	∥	∥	∥	∥
getreturnDefinition	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∥	∥	∥	∥	∥	∥	∥
getexpectedReturnRate	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∥	∥	∥	∥	∥	∥	∥
getvolatility	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∥	∥	∥	∥	∥	∥	∥
getpathStartValue	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∥	∥	∥	∥	∥	∥	∥
writeFile	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∥	∥	∥	∥	∥	∥	∥
getRatePath	∦	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥
computeFluctuationsGaussianOverload	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦

10 ToInitAllTasks

Table 29: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
ToInitAllTasks	√
getname	√
getstartDate	√
getendDate	√
getdTime	√
getreturnDefinition	√
getexpectedReturnRate	√
getvolatility	√
getnTimeSteps	√
getpathStartValue	√
getheader	√
setheader	√
setname	√
setstartDate	√
setendDate	√
setDTime	√
setReturnDefinition	√
setExpectedReturnRate	√
setVolatility	√
setnTimeSteps	√
setpathStartValue	√

Table 30: State Transition Matrix

	alive
alive	↑

Table 31: Methods Concurrency Matrix

	ToInitAllTasks	getname	getstartDate	getendDate	getdTime	getreturnDefinition	getexpectedReturnRate	getvolatility	getnTimeSteps	getpathStartValue	getheader	setheader	setname	setstartDate	setendDate	setDTime	setReturnDefinition	setExpectedReturnRate	setVolatility	setnTimeSteps	setpathStartValue
ToInitAllTasks	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
getname	∦	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
getstartDate	∦	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
getendDate	∦	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
getdTime	∦	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
getreturnDefinition	∦	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
getexpectedReturnRate	∦	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦

getvolatility	∦	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
getnTimeSteps	∦	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
getpathStartValue	∦	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
getheader	∦	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
setheader	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
setname	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
setstartDate	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
setendDate	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
setDTime	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
setReturnDefinition	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
setExpectedReturnRate	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
setVolatility	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
setnTimeSteps	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
setpathStartValue	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦

11 ToResult

Table 32: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
ToResult	√
getexpectedReturnRate	√
getvolatility	√
toString	√
getheader	√
setheader	√
setexpectedReturnRate	√
setvolatility	√
getVolatility2	√
setvolatility2	√
getfinalStockPrice	√
setfinalStockPrice	√
getpathValue	√
setpathValue	√

Table 33: State Transition Matrix

	alive
alive	↑

Table 34: Methods Concurrency Matrix

	ToResult	getexpectedReturnRate	getvolatility	toString	getheader	setheader	setexpectedReturnRate	setvolatility	getVolatility2	setvolatility2	getfinalStockPrice	setfinalStockPrice	getpathValue	setpathValue
ToResult	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
getexpectedReturnRate	∦	∥	∥	∥	∥	∦	∦	∦	∥	∦	∥	∦	∥	∦
getvolatility	∦	∥	∥	∥	∥	∦	∦	∦	∥	∦	∥	∦	∥	∦
toString	∦	∥	∥	∥	∥	∦	∦	∦	∥	∦	∥	∦	∥	∦
getheader	∦	∥	∥	∥	∥	∦	∦	∦	∥	∦	∥	∦	∥	∦
setheader	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
setexpectedReturnRate	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
setvolatility	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
getVolatility2	∦	∥	∥	∥	∥	∦	∦	∦	∥	∦	∥	∦	∥	∦
setvolatility2	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
getfinalStockPrice	∦	∥	∥	∥	∥	∦	∦	∦	∥	∦	∥	∦	∥	∦
setfinalStockPrice	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
getpathValue	∦	∥	∥	∥	∥	∦	∦	∦	∥	∦	∥	∦	∥	∦
setpathValue	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦

12 PriceStock

Table 35: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
PriceStock	√
setInitAllTasks	√
setTask	√
run	√
getResult	√

Table 36: State Transition Matrix

	alive
alive	↑

Table 37: Methods Concurrency Matrix

	PriceStock	setInitAllTasks	setTask	run	getResult
PriceStock	∦	∦	∦	∦	∦
setInitAllTasks	∦	∦	∦	∦	∦
setTask	∦	∦	∦	∦	∦
run	∦	∦	∦	∦	∦
getResult	∦	∦	∦	∦	∥

13 ToTask

Table 38: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
ToTask	$\surd$
getheader	$\surd$
getrandomSeed	$\surd$
setheader	$\surd$
setrandomSeed	$\surd$

Table 39: State Transition Matrix

	alive
alive	$\uparrow$

Table 40: Methods Concurrency Matrix

	ToTask	getheader	getrandomSeed	setheader	setrandomSeed
ToTask	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
getheader	$\nparallel$	$\parallel$	$\parallel$	$\nparallel$	$\nparallel$
getrandomSeed	$\nparallel$	$\parallel$	$\parallel$	$\nparallel$	$\nparallel$
setheader	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
setrandomSeed	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$

14 DemoException

Table 41: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
DemoException	$\surd$

Table 42: State Transition Matrix

	alive
alive	$\uparrow$

15 JGFInstrumentor

Table 43: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
JGFInstrumentor	$\surd$
addTimer	$\surd$
startTimer	$\surd$
stopTimer	$\surd$
addOpsToTimer	$\surd$
readTimer	$\surd$
resetTimer	$\surd$
printTimer	$\surd$
printperfTimer	$\surd$
storeData	$\surd$
retrieveData	$\surd$
printHeader	$\surd$
main	$\surd$

Table 44: State Transition Matrix

	alive
alive	$\uparrow$

Table 45: Methods Concurrency Matrix

	JGFInstrumentor	addTimer	startTimer	stopTimer	addOpsToTimer	readTimer	resetTimer	printTimer	printperfTimer	storeData	retrieveData	printHeader	main
JGFInstrumentor	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
addTimer	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$	$\nparallel$
startTimer	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$	$\nparallel$
stopTimer	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$	$\nparallel$
addOpsToTimer	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$	$\nparallel$
readTimer	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$	$\nparallel$
resetTimer	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$	$\nparallel$
printTimer	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$	$\nparallel$
printperfTimer	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$	$\nparallel$
storeData	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$	$\nparallel$
retrieveData	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$	$\nparallel$
printHeader	$\nparallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$
main	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$	$\nparallel$

16 JGFTimer

Table 46: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
JGFTimer	$\surd$
start	$\surd$
stop	$\surd$
addops	$\surd$
reset	$\surd$
print	$\surd$
perf	$\surd$
printperf	$\surd$
longprint	$\surd$

Table 47: State Transition Matrix

	alive
alive	$\uparrow$

Table 48: Methods Concurrency Matrix

	JGFTimer	start	stop	addops	reset	print	perf	printperf	longprint
JGFTimer	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
start	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
stop	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
addops	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
reset	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
print	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
perf	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$	$\parallel$	$\parallel$
printperf	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$	$\parallel$	$\parallel$
longprint	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$	$\parallel$	$\parallel$

17 test

Table 49: Method’s Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
test	$\surd$
createObject	$\surd$
readA	$\surd$
main	$\surd$

Table 50: State Transition Matrix

	alive
alive	$\uparrow$

Table 51: Methods Concurrency Matrix

	test	createObject	readA	main
test	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
createObject	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
readA	$\nparallel$	$\nparallel$	$\parallel$	$\nparallel$
main	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$

18 Utilities

Table 52: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
Utilities	√
which	√
splitString	√
joinString	√
joinStringOverloaded	√

Table 53: State Transition Matrix

	alive
alive	↑

Table 54: Methods Concurrency Matrix

	Utilities	which	splitString	joinString	joinStringOverloaded
Utilities	∦	∦	∦	∦	∦
which	∦	∦	∦	‖	‖
splitString	∦	∦	‖	‖	‖
joinString	∦	‖	‖	‖	‖
joinStringOverloaded	∦	‖	‖	‖	‖

19 Abbreviation

Table 55: 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

20 Annotated version of the input program generated by Sip4J

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

@ClassStates({@State(name = "alive")})
class JGFMonteCarloBenchSizeA {
@Perm(ensures="unique(this) in alive")
JGFMonteCarloBenchSizeA() {   }

@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
  void main(String argv[]) {

}

}ENDOFCLASS

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

class JGFMonteCarloBench {
@Perm(ensures="unique(this) in alive")
JGFMonteCarloBench() {   }

@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public void JGFrun(int size) {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public void JGFsetsize(int size) {

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public void JGFinitialise() {

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public void JGFapplication() {

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public void JGFvalidate() {

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public void JGFtidyup() {

}

}ENDOFCLASS

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

class CallAppDemo {
@Perm(ensures="unique(this) in alive")
CallAppDemo() {   }

@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public void initialise() {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public void presults() {

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public void runiters() {

}

}ENDOFCLASS
```

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

class AppDemo {
@Perm(ensures="unique(this) in alive")
AppDemo() {   }

@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public void initSerial() {

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 private void initTasks(int nRunsMC) {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public void processSerial() {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 private void processResults() {

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public void runSerial() {

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public String getdataDirname() {
 return null;

}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
 public void setdataDirname(String dataDirname) {

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public String getdataFilename() {
 return null;

}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
 public void setdataFilename(String dataFilename) {

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public int getnTimeStepsMC() {
 return 0;

}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
 public void setnTimeStepsMC(int nTimeStepsMC) {

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public int getnRunsMC() {
 return 0;

}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
 public void setnRunsMC(int nRunsMC) {

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public Vector gettasks() {
 return null;
}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public void settasks(Vector tasks) {

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public Vector getresults() {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public void setresults(Vector results) {

}

}ENDOFCLASS

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

class Universal {
@Perm(ensures="unique(this) in alive")
Universal() {   }

@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
 public void setprompt(String prompt) {

}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
 public void setDEBUG(boolean DEBUG) {

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public void dbgPrintln(String s) {

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public void errPrintln(String s) {

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public boolean getDEBUG() {
 return 0;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public boolean getUNIVERSALDEBUG() {
 return 0;

}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
 public void setUNIVERSALDEBUG(boolean UNIVERSAL_DEBUG) {

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public String getprompt() {
 return null;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public void dbgPrint(String s) {

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public void errPrint(String s) {

}

}ENDOFCLASS
```

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

class PathId {
@Perm(ensures="unique(this) in alive")
PathId() {   }

@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public void dbgDumpFields() {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public void copyInstanceVariables(PathId obj) {

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public double getdTime() {
 return 0;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public String getname() {
 return null;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public int getstartDate() {
 return 0;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public int getendDate() {
 return 0;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public void setname(String name) {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public void setstartDate(int startDate) {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public void setendDate(int endDate) {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public void setdTime(double dTime) {

}

}ENDOFCLASS

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

class RatePath {
@Perm(ensures="unique(this) in alive")
RatePath() {   }

@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public ReturnPath getReturnCompounded() {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public ReturnPath getReturnNonCompounded() {
 return null;
}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 private void readRatesFile(String dirName, String filename) {

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public double getEndPathValue() {
 return 0;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public double getPathValue(int index) {
 return 0;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public void incpathValue(double[] operandPath) {

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public double[] getpathValue() {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public void setpathValue(double[] pathValue) {

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public int[] getpathDate() {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public void setpathDate(int[] pathDate) {

}

}ENDOFCLASS

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

class ReturnPath {
@Perm(ensures="unique(this) in alive")
ReturnPath() {   }

@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public void estimatePath() {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public void computeMean() {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public void computeVariance() {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public void computeExpectedReturnRate() {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public void computeVolatility() {

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
```

```
 public void dbgDumpFields() {

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public double getexpectedReturnRate() {
 return 0;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public double getvolatility() {
 return 0;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public int getreturnDefinition() {
 return 0;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public double getvolatility2() {
 return 0;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public double[] getpathValue() {
 return null;

}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
 public void setpathValue(double[] pathValue) {

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public int getnPathValue() {
 return 0;

}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
 public void setnPathValue(int nPathValue) {

}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
 public void setreturnDefinition(int returnDefinition) {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public void setexpectedReturnRate(double expectedReturnRate) {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public void setvolatility(double volatility) {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public void setvolatility2(double volatility2) {

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public double getmean() {
 return 0;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public void setmean(double mean) {

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public double getvariance() {
 return 0;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public void setvariance(double variance) {

}

}ENDOFCLASS

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

class MonteCarloPath {
@Perm(ensures="unique(this) in alive")
MonteCarloPath() {   }

@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 private void copyInstanceVariables(ReturnPath obj) {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public void setreturnDefinition(int returnDefinition) {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public void setexpectedReturnRate(double expectedReturnRate) {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public void setvolatility(double volatility) {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public void setnTimeSteps(int nTimeSteps) {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public void setpathStartValue(double pathStartValue) {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public void setpathValue(double[] pathValue) {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public void setfluctuations(double[] fluctuations) {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public void computeFluctuationsGaussian(long randomSeed) {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public void computePathValue(double startValue) {

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public double[] getpathValue() {
 return null;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public int getnTimeSteps() {
 return 0;

}
```

```
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public double[] getfluctuations() {
 return null;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public int getreturnDefinition() {
 return 0;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public double getexpectedReturnRate() {
 return 0;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public double getvolatility() {
 return 0;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public double getpathStartValue() {
 return 0;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public void writeFile(String dirName, String filename) {

}

 public RatePath getRatePath() {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public void computeFluctuationsGaussianOverload() {

}

}ENDOFCLASS

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

class ToInitAllTasks {
@Perm(ensures="unique(this) in alive")
ToInitAllTasks() {   }

@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public String getname() {
 return null;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public int getstartDate() {
 return 0;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public int getendDate() {
 return 0;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public double getdTime() {
 return 0;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public int getreturnDefinition() {
```

```
 return 0;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public double getexpectedReturnRate() {
 return 0;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public double getvolatility() {
 return 0;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public int getnTimeSteps() {
 return 0;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public double getpathStartValue() {
 return 0;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public String getheader() {
 return null;

}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
 public void setheader(String header) {

}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
 public void setname(String name) {

}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
 public void setstartDate(int startDate) {

}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
 public void setendDate(int endDate) {

}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
 public void setDTime(double dTime) {

}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
 public void setReturnDefinition(int returnDefinition) {

}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
 public void setExpectedReturnRate(double expectedReturnRate) {

}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
 public void setVolatility(double volatility) {

}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
 public void setnTimeSteps(int nTimeSteps) {

}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
 public void setpathStartValue(double pathStartValue) {
}

}ENDOFCLASS

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

class ToResult {
@Perm(ensures="unique(this) in alive")
ToResult() {   }

@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public double getexpectedReturnRate() {
 return 0;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public double getvolatility() {
 return 0;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public String toString() {
 return null;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public String getheader() {
 return null;

}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
 public void setheader(String header) {

}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
 public void setexpectedReturnRate(double expectedReturnRate) {

}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
 public void setvolatility(double volatility) {

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public double getVolatility2() {
 return 0;

}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
 public void setvolatility2(double volatility2) {

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public double getfinalStockPrice() {
 return 0;

}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
 public void setfinalStockPrice(double finalStockPrice) {

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public double[] getpathValue() {
 return null;

}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
 public void setpathValue(double[] pathValue) {
}

}ENDOFCLASS

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

class PriceStock {
@Perm(ensures="unique(this) in alive")
PriceStock() {   }

@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public void setInitAllTasks(Object obj) {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public void setTask(Object obj) {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public void run() {

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public Object getResult() {
 return null;

}

}ENDOFCLASS

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

class ToTask {
@Perm(ensures="unique(this) in alive")
ToTask() {   }

@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public String getheader() {
 return null;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public long getrandomSeed() {
 return 0;

}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
 public void setheader(String header) {

}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
 public void setrandomSeed(long randomSeed) {

}

}ENDOFCLASS

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

class DemoException {
@Perm(ensures="unique(this) in alive")
DemoException() {   }


}ENDOFCLASS

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

class JGFInstrumentor {
@Perm(ensures="unique(this) in alive")
JGFInstrumentor() {   }
```

```
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
  void addTimer(String name) {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
  void startTimer(String name) {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
  void stopTimer(String name) {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
  void addOpsToTimer(String name, double count) {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
  double readTimer(String name) {
 return 0;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
  void resetTimer(String name) {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
  void printTimer(String name) {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
  void printperfTimer(String name) {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
  void storeData(String name, Object obj) {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
  void retrieveData(String name, Object obj) {

}

  void printHeader(int section, int size) {

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
  void main(String argv[]) {

}

}ENDOFCLASS

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

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

@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public void start() {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public void stop() {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
```

```
 public void addops(double count) {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public void reset() {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public void print() {

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public double perf() {
 return 0;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public void printperf() {

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public void longprint() {

}

}ENDOFCLASS

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

class test {
@Perm(ensures="unique(this) in alive")
test() {   }

@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public void createObject() {

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public void readA() {

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
  void main(String[] arg) {

}

}ENDOFCLASS

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

class Utilities {
@Perm(ensures="unique(this) in alive")
Utilities() {   }

@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
  String which(String executable, String pathEnv) {
 return null;

}
@Perm(requires="immutable(this) in alive",
ensures="immutable(this) in alive")
  String[] splitString(String splitChar, String arg) {
 return null;

}

  String joinString(String joinChar, String stringArray[]) {
 return null;

}

  String joinStringOverloaded(String joinChar, String stringArray[], int index) {
 return null;

}

}ENDOFCLASS
```