Checking.deposit() method

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Items | value | Method\_Value | Condition |  | Requirements |  |
| amount | (0.0, inf) | true if internal condition A is met | A | | getState() != State.CLOSE amount > 0.0 balance >= 0.0 | |
|  |  | false otherwise |
|  |  |  |

Boundary Value Analysis: amount = X

1. <= X <= 0.0000001

1000000.00 <= X <= Inf

Internal condition A not exercised.

Checking.withdraw() method

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Items | value | Method\_Value | Condition |  | Requirements |  |
| amount | (0.0, inf) | true if internal condition B is met | B | | getState()= State.OPEN getState() = State.OVERDRAWN amount > 0.0 balance > -100.00 numWithdraws > 10 | |
|  |  | false otherwise |
|  |  |  |

Boundary Value Analysis: amount = X

1. <= X <= 0.0000001

1000000.00 <= X <= Inf

Internal condition B not exercised.

# Ant JUnit task report

Testsuite: banking.primitive.core.AccountServerTest

Tests run: 5, Failures: 3, Errors: 0, Skipped: 0, Time elapsed: 0.951 sec

------------- Standard Output ---------------

Reading from file accounts.ser...

------------- ---------------- ---------------

Testcase: testNewAccountBalance took 0.003 sec

Testcase: testNewAccountType took 0.01 sec

FAILED

Expected exception: java.lang.IllegalArgumentException

junit.framework.AssertionFailedError: Expected exception: java.lang.IllegalArgumentException

Testcase: testNewAccountDuplicate took 0 sec

Testcase: testNewAccount took 0.001 sec

FAILED

failed to add new acocunts

junit.framework.AssertionFailedError: failed to add new acocunts

at banking.primitive.core.AccountServerTest.testNewAccount(Unknown Source)

Testcase: testGetAccount took 0.001 sec

FAILED

null

junit.framework.AssertionFailedError

at banking.primitive.core.AccountServerTest.testGetAccount(Unknown Source)

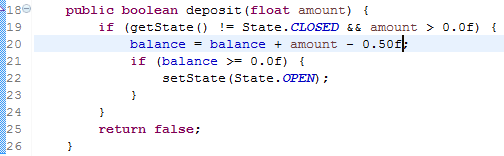
Testsuite: banking.primitive.core.CheckingTest

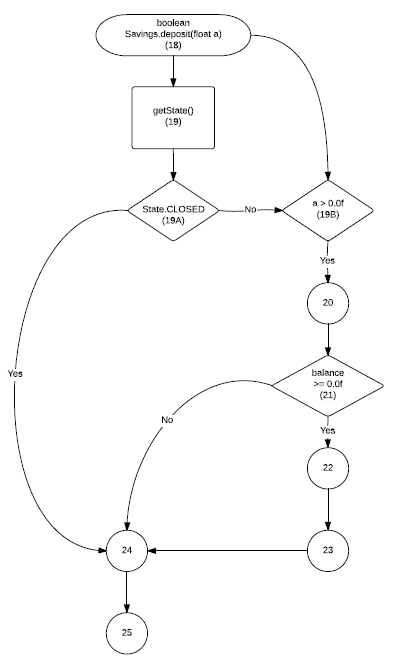
Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.356 sec

Testcase: testWithdraw took 0.057 sec

Testcase: testDeposit took 0.001 sec

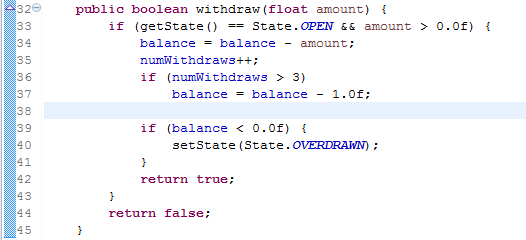
Task 2, Part 1

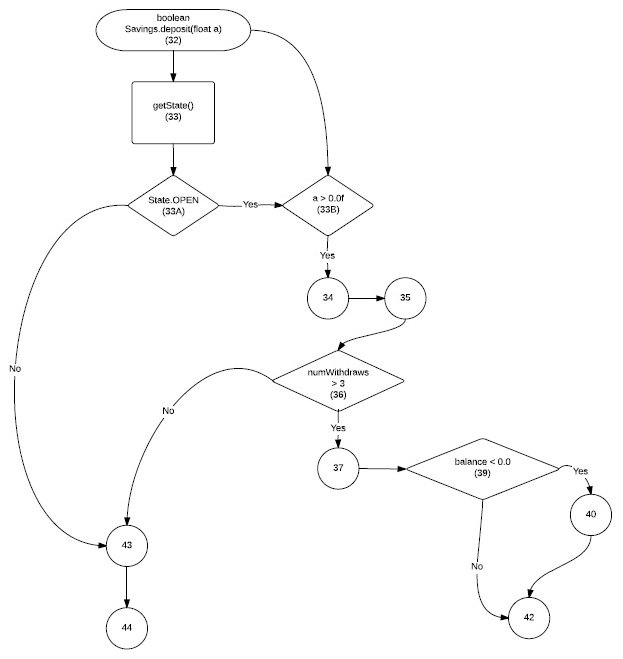




Savings deposit() Coverage

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Item | In | Out | Remark | |
| 18 | (0.0, Inf) | [0,1] |  |  |
| 19A | [0, 2] | [0,1] | If, enum | && |
| 19B | (0, Inf) | [0,1] | > |
| 20 | (0, Inf) | (0, Inf) | Not sure of the - 0.5f |  |
| 21 | [0, Inf) | [0, 1] | If |  |
| 22 | [0, 2] | None | void |  |
| 23 | None | None | If, fall-through |  |
| 24 | None | None | If, Fall-through |  |
| 25 | [0, 1] | [0, 1] | Return value |  |

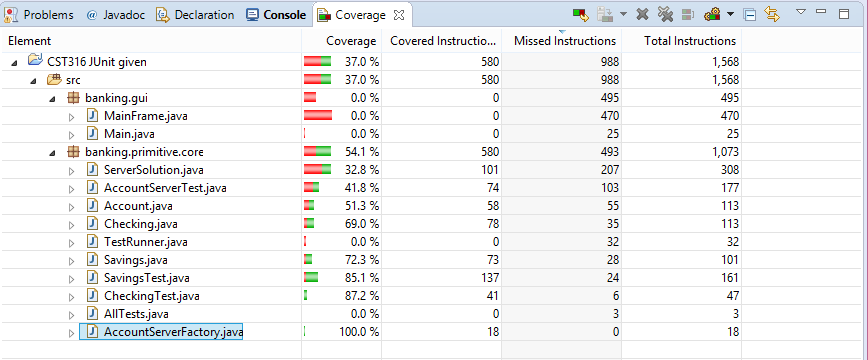




Savings withdraw() Coverage

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Item | In | Out | Remark | |
| 32 | (0.0, Inf) | [0,1] |  |  |
| 33A | [0, 2] | [0,1] | If, enum | && |
| 33B | (0, Inf) | [0,1] | > |
| 34 | (0, Inf) | (0, Inf) | - |  |
| 35 | [0, 3] | [0, 1] | inc |  |
| 36 | [0, 3] | [0, 1] | if |  |
| 37 | [0, Inf) | [0, Inf) | - |  |
| 39 | [0, Inf) | [0, 1] | If |  |
| 40 | [0, 2] | None | If, enum |  |
| 41 | None | None | Fall-through |  |
| 42 | [0, 1] | [0, 1] | return |  |
| 43 | None | None | Fall-through |  |
| 44 | [0, 1] | [0, 1] | return |  |

Coverage Results



ii. What is the overall code coverage for the project? [37%], For the ServerSolution.java file? [32.8%]

iii. Does the EclEmma coverage tool perform node or edge or condition or path coverage? My impression tells me that EclEmma coverage tool can do all of these, but for this lab it performed path coverage.

iv. What is the code coverage of Savings.java, specifically the deposit [85.3%] and withdraw [84.3%] methods?[85.1%] If these methods are not at 100%, update your unit tests to improve the coverage.

v. What is the code coverage of Checking.java [87.2%], specifically the deposit [88.9%] and withdraw [84.6%] methods