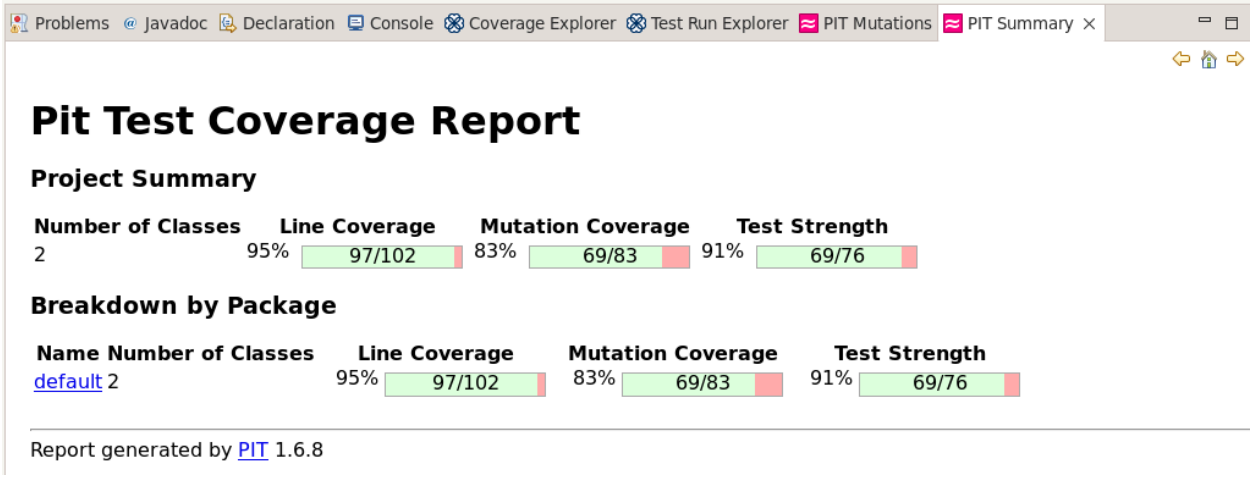
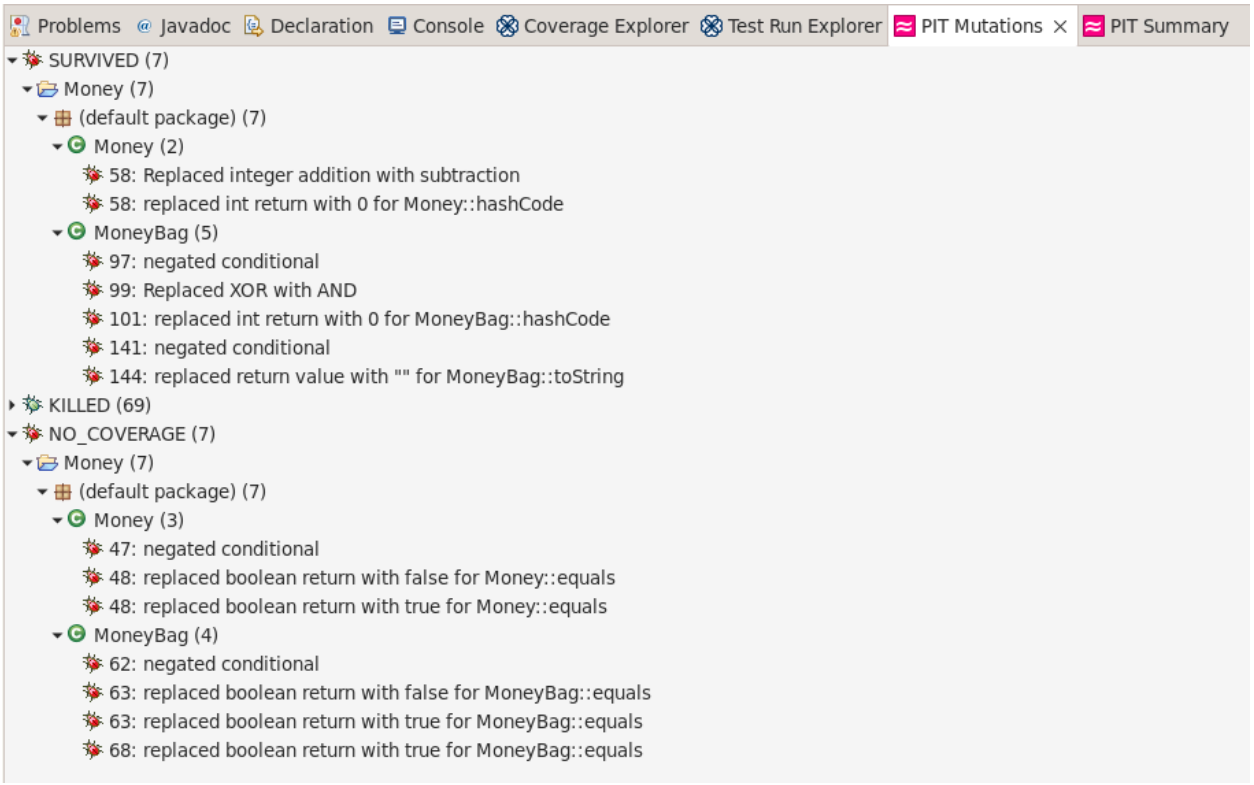
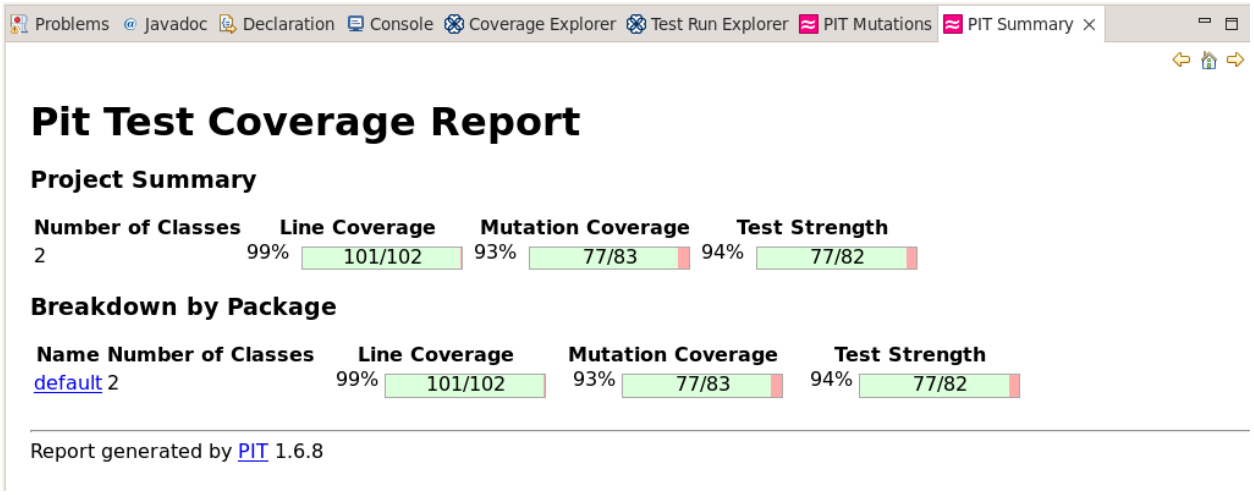
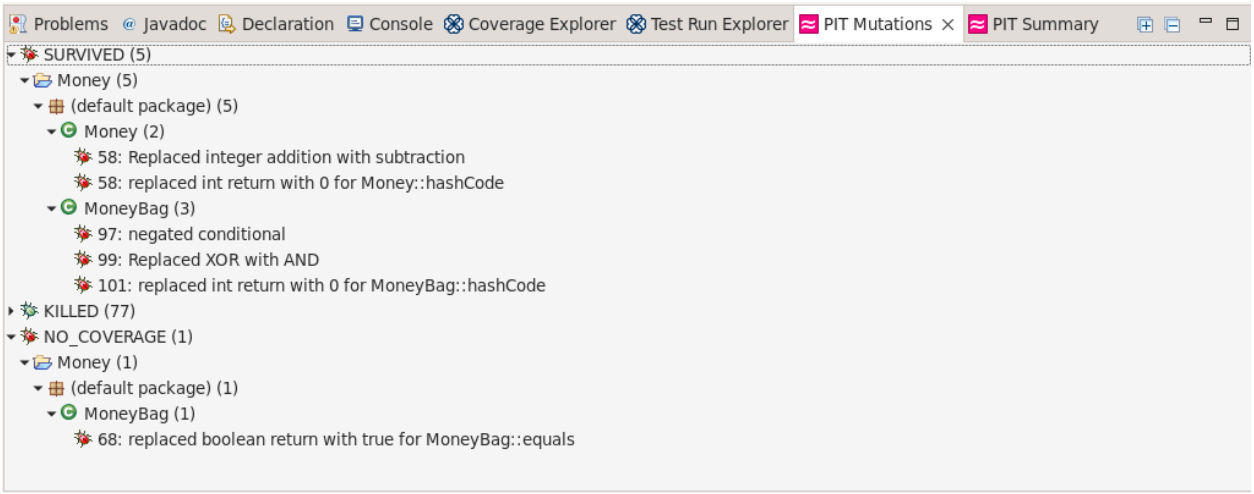
Q1:
2.
Initial PIT Mutations and PIT Summary. Code not modified.



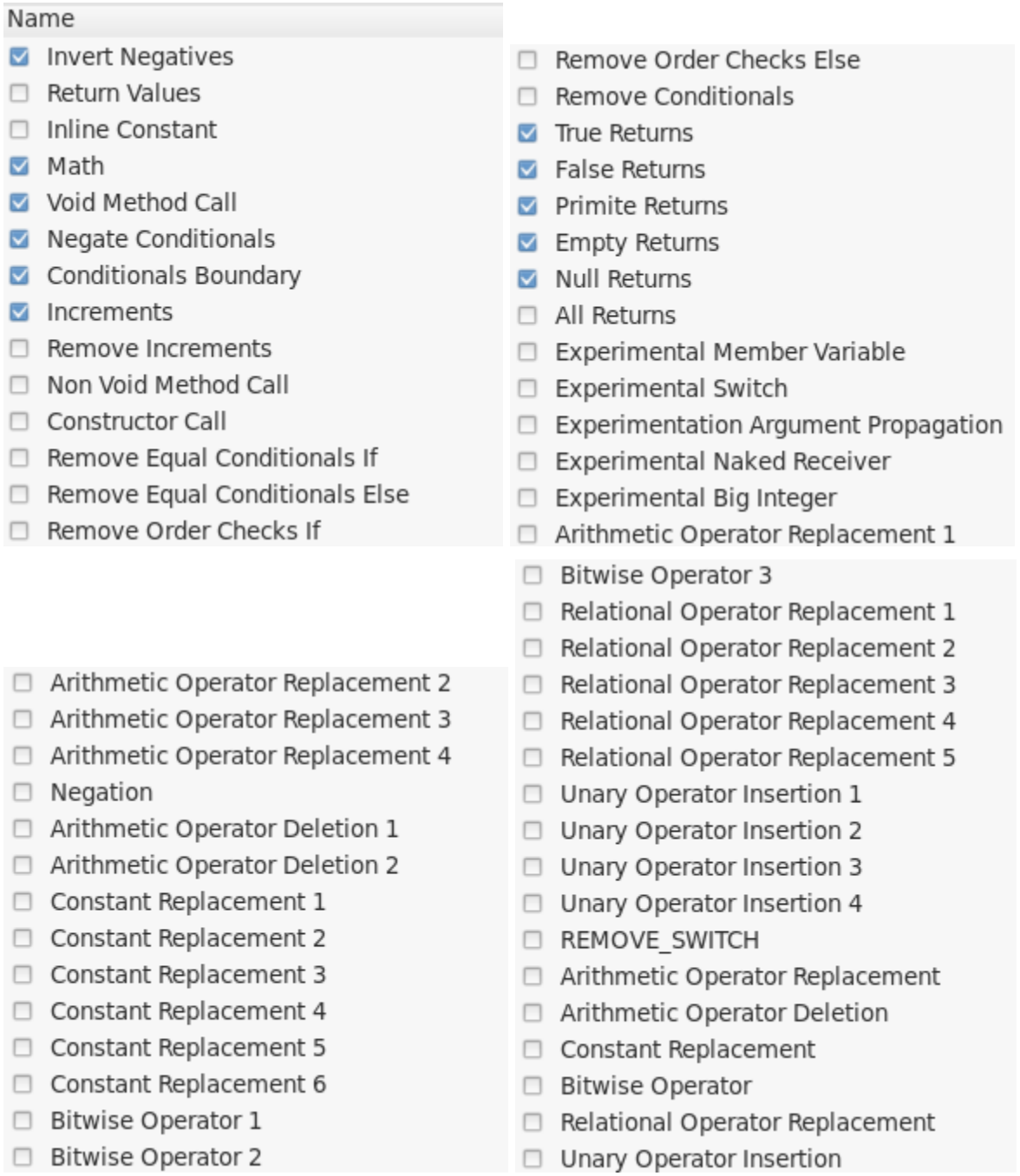


1.
Modified code to achieve a statement/line coverage near or equal to 100%. I added 3 test cases (testMoneyEqualsSpecific, testMoneyBagEqualsSpecific, and testMoneyBagToString)





3 and 4.
Initial Mutators and Final Mutators. I did not add any mutators because everytime I added one, the number of surviving mutants would either increase or just stay the same. If the number of surviving mutants increases, this causes the mutation coverage to go down. So, I just left it as is.



5.
Line Coverage: 99% (101/102)
Mutation Coverage: 93% (77/83)
Test Strength: 94% (77/82)
Total Mutants Generated: 83
Killed Mutants: 77
Survived Mutants: 5
Uncovered Mutants: 1
Active Mutators: Invert Negatives, Math, Void Method Call, Negate Conditionals, Conditionals Boundary, Increments, True Returns, False Returns, Primite Returns, Empty Returns, Null Returns

6.

Metric	Initial Run (Default)	Final Run	Change
Line Coverage	95% (97/102)	99% (101/102)	+4%
Mutation Coverage	83% (69/83)	93% (77/83)	+10%
Test Strength	91% (69/76)	94% (77/82)	+3%
Killed Mutants	69	77	+8
Survived Mutants	7	5	-2
Uncovered Mutants	7	1	-6

7.
Killed from Initially Survived (2 mutants):

- MoneyBag: 141: negated conditional (in simplify): This was likely killed by tests involving MoneyBag.create followed by operations (like testSimplify or testNormalize variants) where the bag should simplify to a single Money object. The new, more specific equality tests might have made assertions fail when the negation prevented simplification.
- MoneyBag: 144: replaced return value with "" for MoneyBag::toString: This was directly killed by the new testMoneyBagToString assertion which explicitly checked the format of the string output. The mutant returning an empty string failed this check.

Killed from Initially Uncovered (6 mutants):

- Money: 47: negated conditional (in equals)
- Money: 48: replaced boolean return with false for Money::equals
- Money: 48: replaced boolean return with true for Money::equals
- MoneyBag: 62: negated conditional (in equals)
- MoneyBag: 63: replaced boolean return with false for MoneyBag::equals
- MoneyBag: 63: replaced boolean return with true for MoneyBag::equals

How: These 6 mutants, all within the equals methods, were initially marked as NO_COVERAGE. The new, more detailed tests (testMoneyEqualsSpecific, testMoneyBagEqualsSpecific) specifically exercised the conditions these mutants reside in (comparing zero vs non-zero, same currency vs different, same amounts vs different, object type checks). By executing these specific paths and having assertions that depended on the correct boolean outcome of equals, these mutants were killed when they altered the return value or logic.

8.
Survived Mutants (5):

- Money: 58: Replaced integer addition with subtraction (in hashCode)
- Money: 58: Replaced int return with 0 for Money::hashCode
- MoneyBag: 97: Negated conditional (in findMoney)
- MoneyBag: 99: Replaced XOR with AND (in hashCode)
- MoneyBag: 101: replaced int return with 0 for MoneyBag::hashCode

Why they survive:
hashCode issues (#1, #2, #4, #5): The tests for hashCode (testMoneyHash, testMoneyBagHash) likely only assert that equal objects have equal hash codes. They probably don't test if the hash codes are good (well-distributed) or use them in collections. Mutants like "return 0" or changing addition/XOR might still produce hash codes that satisfy the basic equals/hashCode contract for the specific objects used in the tests, even though the hash function is broken. To kill these, tests using HashMap or HashSet with carefully chosen Money/MoneyBag objects as keys would be needed to demonstrate incorrect behavior (e.g., collisions or failure to retrieve).

MoneyBag: 97: negated conditional (#3): Although line coverage is high, the specific test path through findMoney where negating the m.currency().equals(currency) check causes a test failure isn't being hit. Perhaps the tests only call findMoney in scenarios where the currency is always found or always not found in a way that the negation doesn't change the outcome asserted by the test. A more specific test asserting the state of a bag after an operation that relies critically on findMoney returning the correct object (or null) under specific conditions is needed.

Uncovered Mutant (1):

- MoneyBag: 68: replaced boolean return with true for MoneyBag::equals

Why it's uncovered:
This mutant changes the final return true in the MoneyBag.equals method (after the loop successfully checks all elements). PIT still marks this as NO_COVERAGE, meaning it believes no test executes this specific return statement in a context where this mutation could be detected. Even though the line is covered (99% line coverage), PIT's analysis suggests the existing tests comparing equal MoneyBag objects aren't sufficient for this specific mutation. It might require comparing two complex, non-empty, identical MoneyBag objects where the loop

must complete fully and return true, and the test assertion relies solely on this final true. It could also be an edge case in PIT's analysis.