

# INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY BANGALORE

## Software Testing Report (CS 731)

---

### Mutation Testing

---



### Group Members:

Pallav Jain	MT2022067
Prashant Kumar	MT2022079

## Abstract

The following document is a report of the details of the concepts involved in the project of CS731 Software Testing Course.

*Keywords:* Mutation Testing, PITest, JUnit, Unit Testing

### 1. Problem Description

- a. **Problem Statement:** Mutation source code: Projects that use mutation testing, based on mutation operators applied at the level of a statement within a method or a function.
- b. **Aim:** The mutated program needs to be strongly killed by the designed test cases. At least three different mutation operators should be used.

### 2. Introduction

Mutation Testing is a type of Software Testing that is performed to design new software tests and also evaluate the quality of already existing software tests. Mutation testing is related to modification of a program in small ways. It focuses to help the tester develop effective tests or locate weaknesses in the test data used for the program.

Mutation testing involves making syntactically valid changes to a software artifact and then testing the artifact. We consider grammars of software artifacts again. Grammars generate valid strings. We use derivations in grammars to generate invalid strings too. Testing based on these valid and invalid strings is called mutation testing.

### 3. PITest

PITest is a state of the art mutation testing system, providing standard test coverage for Java and the JVM. It's fast, scalable and integrates with modern test and build tooling.

PITest runs unit tests against automatically modified versions of the application code. When the application code changes, it should produce different results and cause the unit tests to fail. If a unit test does not fail in this situation, it may indicate an issue with the test suite.

## 4. Source Code

Link to source code - <https://github.com/pallavjain12/MutationTesting>

Link to PTest Result - [PTest Result](#)

This versatile calculator offers a comprehensive range of functions, including nPr, nCr, factorial, Nth Fibonacci calculations, and almost all the functionalities of a scientific calculator.

Each method in this Calculator class performs a specific mathematical operation. Some of the functionalities implemented are:

- `calculateSum(int a, int b)`: Calculates the sum of two integers a and b.
- `calculateDifference(int a, int b)`: Calculates the difference between two integers a and b.
- `calculateFactorial(int n)`: Calculates the factorial of an integer n.
- `calculatePermutation(int n, int r)`: Calculates the number of permutations of n objects taken r at a time.
- `calculateCombination(int n, int r)`: Calculates the number of combinations of n objects taken r at a time.
- `calculateLogAtoTheBaseB(int a, int b)`: Calculates the logarithm of a to the base b.
- `calculateNthFibonacci(int n)`: Calculates the nth Fibonacci number.
- `calculateApowerB(int a, int b)`: Calculates a raised to the power of b (same as `calculateBinExpo`).
- `calculateBinExpo(int a, int b)`: Calculates a raised to the power of b using binary exponentiation.
- `calculateGCD(int a, int b)`: Calculates the greatest common divisor of two integers a and b.
- `calculateBMI(int weight, int height)`: Calculates the body mass index (BMI) of a person with the given weight and height.

- `calculateAddMatrices(int[], int[])`: Adds two matrices of the same dimensions.
- `calculateSubtractMatrices(int[], int[])`: Subtracts two matrices of the same dimensions.
- `calculateMultiplyMatrices(int[], int[])`: Multiplies two matrices.
- `calculateExponentiateMatrix(int[], int)`: Calculates the power of a square matrix.
- `calculateSimpleInterest(int, int, int)`: Calculates the simple interest for a given principal, rate, and time.

## 5. Mutation Operators

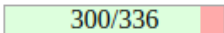
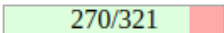
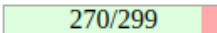
Some of the active mutators present were:

- **Math Mutator**: It is a mutation testing operator that modifies mathematical expressions in a program to test the robustness of the program's mathematical logic. It is a useful tool for identifying and correcting bugs in mathematical calculations and ensuring the program's accuracy in handling numerical operations.
- **Logical mutators**: These modify logical operators (`&&`, `||`, `!`) in a program to test the robustness of logical logic. These mutators can help identify and correct bugs in logical expressions and ensure the program's correctness when evaluating conditions.
- **Negate Conditionals Mutator**: The Negate Conditionals Mutator is a mutation testing operator that negates conditional statements in a program to test the robustness of conditional logic. It is a simple yet effective operator that can expose bugs in conditional statements.
- **Arithmetic Mutators**: These mutators modify arithmetic operators (`+`, `-`, `*`, `/`, `%`) to introduce errors in arithmetic expressions. They can help detect bugs in calculations and ensure the program's correctness when performing numerical operations.

## 6. PITest Results

### Pit Test Coverage Report

#### Project Summary

Number of Classes	Line Coverage	Mutation Coverage	Test Strength
1	89%  300/336	84%  270/321	90%  270/299

#### Breakdown by Package

Name	Number of Classes	Line Coverage	Mutation Coverage	Test Strength
<a href="#">org.example</a> 1		89%  300/336	84%  270/321	90%  270/299

---

Report generated by [PIT](#) 1.9.11

Enhanced functionality available at [arcmutate.com](http://arcmutate.com)

## 7. Execution

### a. Pre-Setup Required

- Make sure to have Java JDK 11 successfully setup.
- Make sure to have maven installed with proper PATH variables initialized.
- Also make sure you have added all the dependencies and plugins required like Junit, Pitest.

### b. How to Run

- Run the following command:
  1. `mvn install`
  2. `mvn test-compile org.pitest:pitest-maven:mutationCoverage`

## 8. Contributions

Prashant Kumar:

- Worked on the source code and mutation tests.
- Worked on curating the final project report.

Pallav Jain:

- Worked on setting up maven framework environment, JUnit4 and PITest initialization.

## 9. References

- a. **Maven and JDK setup** at <https://www.digitalocean.com/community/tutorials/install-maven-linux-ubuntu> and <https://www.baeldung.com/java-mutation-testing-with-pitest>
- b. **Unit testing with JUnit4** at <https://www.vogella.com/tutorials/JUnit4/article.html>
- c. **PITest mutation coverage documentation** at <https://pitest.org/>