

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**  
**ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ В.Н. КАРАЗІНА**  
**ФАКУЛЬТЕТ «КОМП'ЮТЕРНИХ НАУК»**

**КУРСОВА РОБОТА**

з дисципліни «Математичні методи та технології тестування і  
верифікації програмного забезпечення»

Тема «Тестування мутацій (Mutation testing)»

---

Виконав студент 2 курсу

групи КС-21

Зоренко Ярослав Сергійович

Керівник:

ст. викладач Мелкозьорова О. М.

# ЗМІСТ

- ☐ Вступ
- ☐ 1 Теоретичні відомості
  - ☐ 1.1 Тестування програмного забезпечення
  - ☐ 1.2 Особливості методу тестування мутаціями
- ☐ 2 Практичні відомості
  - ☐ 2.1 Інструментальні засоби реалізації мутаційного методу
  - ☐ 2.2 Модель програмного продукту
  - ☐ 2.3 Реалізація тестування мутаціями
- ☐ Висновок
- ☐ Список джерел інформації



# Вступ

- В роботі розглядається програмна модель калькулятора, за допомогою якого буде відтворено мутаційне тестування, а саме проведено декілька тестів з використанням різних логічних та арифметичних операторів для отримання бажаного результату.
- У першому розділі розглядаються теоретичні відомості щодо тестування програмного забезпечення і необхідності його проведення.
- У другому розділі розглядаються практичні відомості, а саме методи реалізації мутаційного тестування, шляхом зміни логічних та арифметичних операторів, значення змінної або навіть її видалення. Також будуть розглянуті всі види мутацій вихідного коду, пов'язаних з перевіркою працездатності програмної моделі калькулятора.
- Таким чином, в роботі буде порушено питання актуальності тестування програмного забезпечення цим методом та питання його надійності при використанні на практиці.

# Теоретичні відомості

## *Структура створення тестів:*

- ✓ Планування робіт (Test Management);
- ✓ Проектування тестів (Test Design);
- ✓ Виконання тестування (Test Execution);
- ✓ Аналіз отриманих результатів (Test Analysis).

*Види життєвих циклів:*

Каскадний;

V-образний;

Спіральний.



# Результатами виконання можуть бути:

- KILLED – в результаті мутації впали всі тести, що перевіряють цей рядок. Це означає, що всі помилки знайдені;
- SURVIVED – мутація пройшла непоміченою. Це означає, що зміна у функціональності не вкрита тестами;
- TIMED\_OUT – тест працював занадто довго (наприклад, в результаті виникнення нескінченного циклу);
- NON\_VIABLE – отриманий в результаті мутації байт-код з якоїсь причини виявився не дійсним;
- MEMORY\_ERROR – в результаті мутації код почав споживати дуже багато пам'яті і впав;
- RUN\_ERROR – в результаті мутації вийшов код, що генерує виняток.

# Переваги та недоліки

## *Переваги:*

- Дозволяє охопити всю вихідну програму;
- Всебічно тестуються програми-мутанти;
- Тестування розкриває всі неясності у вихідному коді.

## *Недоліки:*

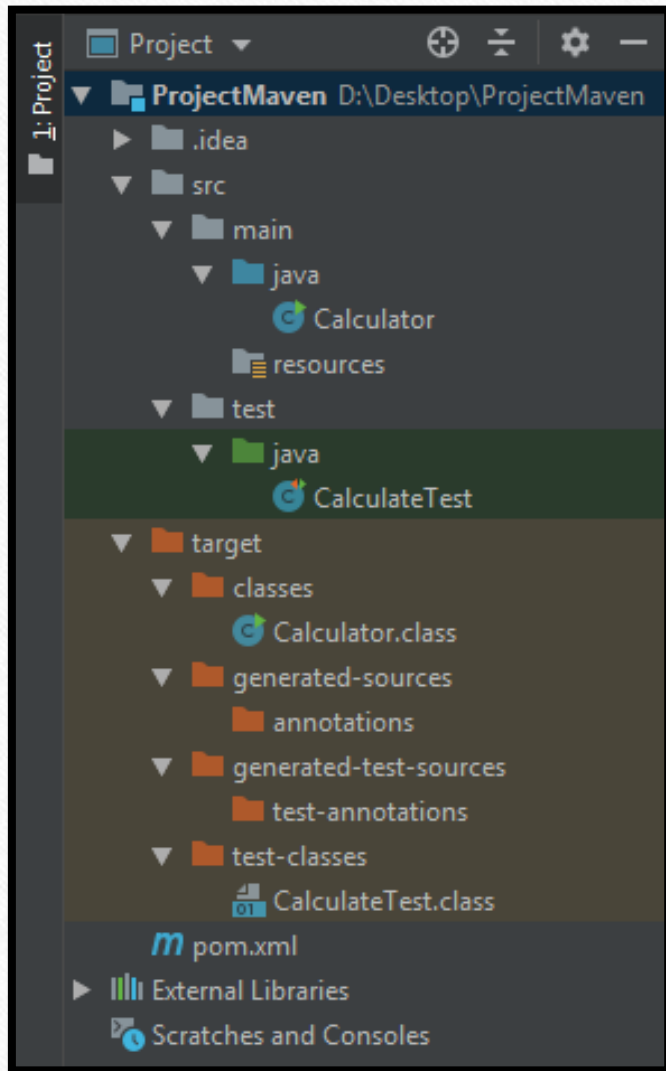
- Мутаційне тестування — надзвичайно дорогий і трудомісткий процес;
- Не автоматизований вид тестування;
- Не може бути застосований для тестування методом чорного ящика, оскільки цей спосіб передбачає зміни у вихідному коді.



# Інструментальні засоби реалізації мутаційного методу

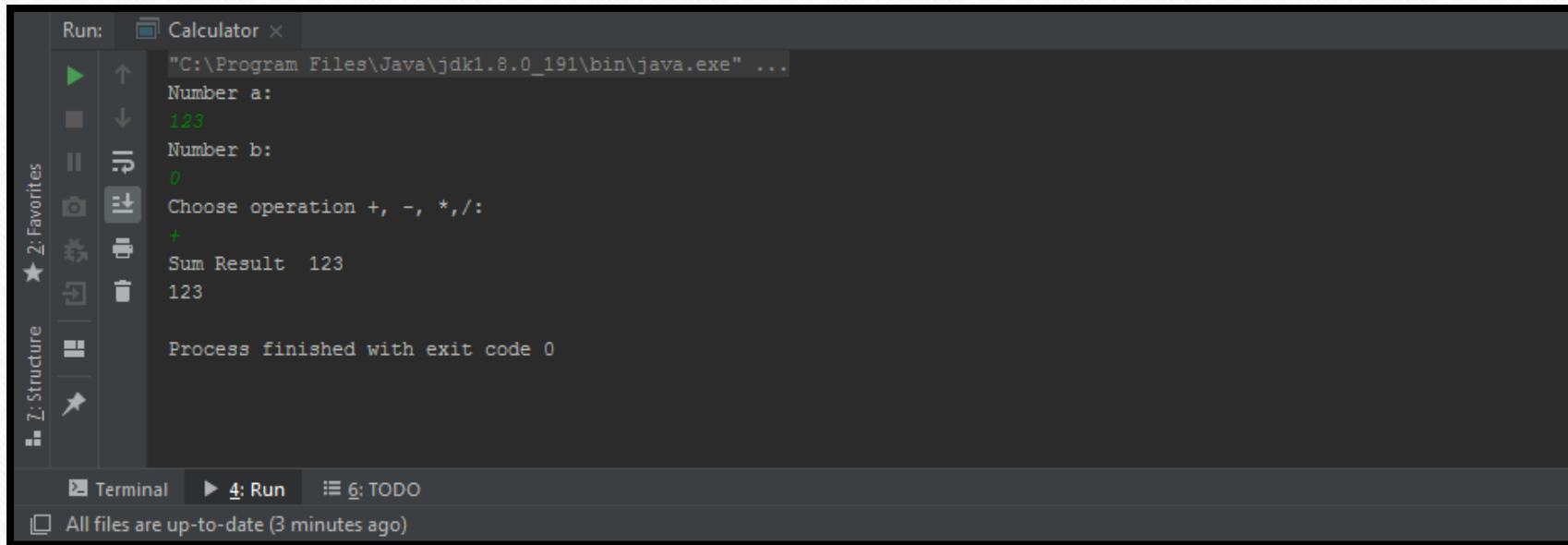






# Модель программного продукту

# Реалізація тестування мутаціями



The screenshot shows an IDE's terminal window with a dark theme. On the left, there is a sidebar with icons for 'Run' (a green play button), 'Debug' (a magnifying glass), and 'Test' (a checkmark). The terminal itself displays the output of a Java program. The first line is the command to run the program: `"C:\Program Files\Java\jdk1.8.0_191\bin\java.exe" ...`. The program then prompts for 'Number a:' and receives the input '123'. It then prompts for 'Number b:' and receives the input '0'. Next, it prompts for 'Choose operation +, -, \*, /:' and receives the input '+'. The program then outputs 'Sum Result 123' and '123'. Finally, it outputs 'Process finished with exit code 0'. At the bottom of the terminal, there are tabs for 'Terminal', 'Run', and 'TODO', and a status bar indicating 'All files are up-to-date (3 minutes ago)'.

```
Run: Calculator x
"C:\Program Files\Java\jdk1.8.0_191\bin\java.exe" ...
Number a:
123
Number b:
0
Choose operation +, -, *, /:
+
Sum Result 123
123

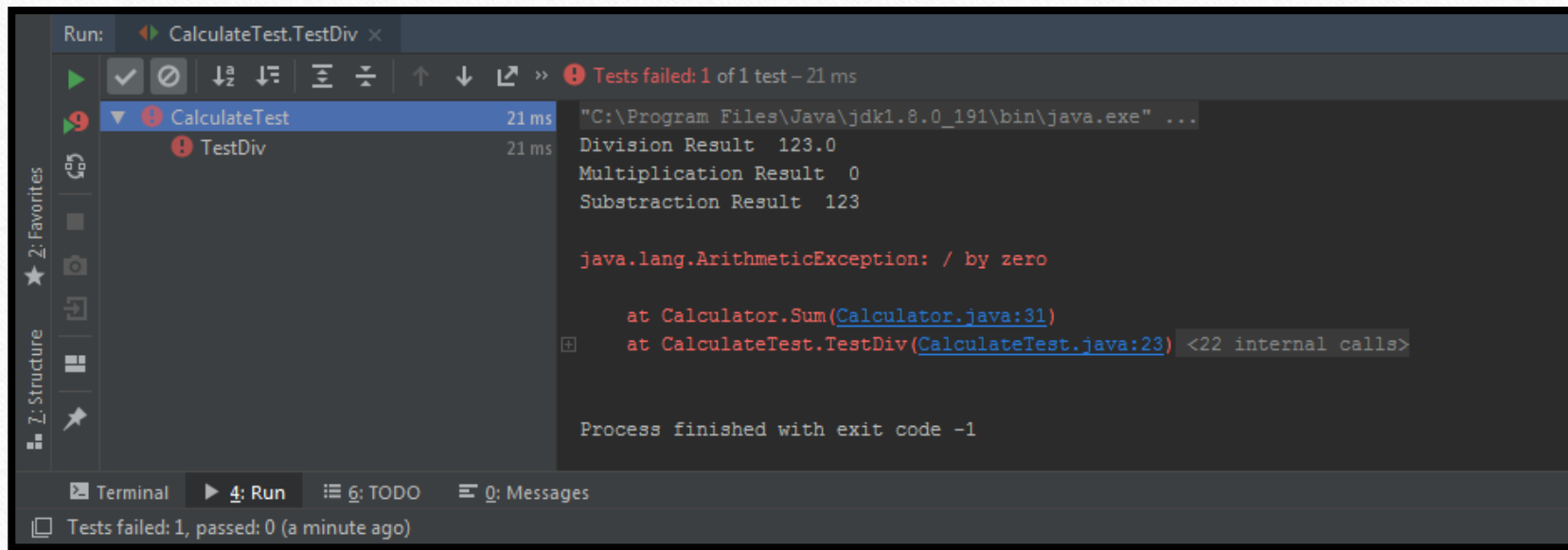
Process finished with exit code 0

Terminal 4: Run 6: TODO
All files are up-to-date (3 minutes ago)
```



# Реалізація тестування мутаціями

```
public int Sum(int a,int b){  
    System.out.println("Sum Result "+ (a/b)); //Заменяем знак "+" на любой другой "*", "+", "-",  
    return (a+b);                             //получаем результат не "123" -> тест не пройден, мутант убит  
}
```



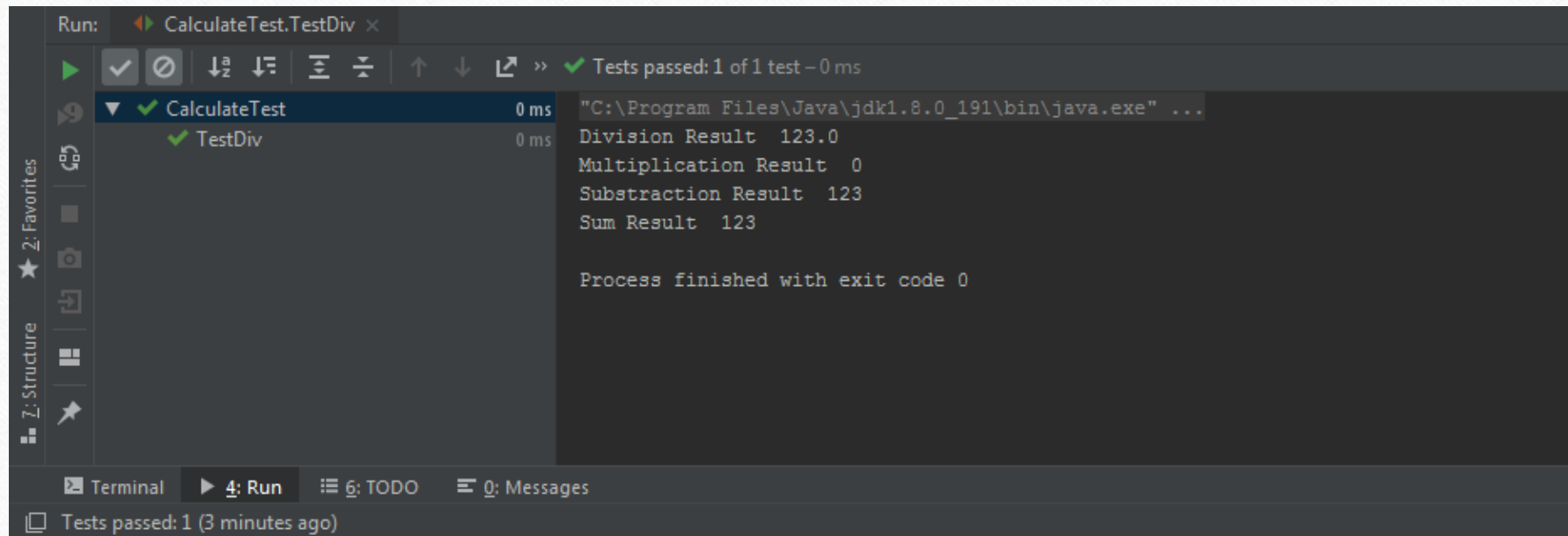
The screenshot shows an IDE's Run window with the following details:

- Run:** CalculateTest.TestDiv x
- Test Results:**
  - CalculateTest (21 ms) - Failed (indicated by a red icon)
  - TestDiv (21 ms) - Failed (indicated by a red icon)
- Output:**

```
"C:\Program Files\Java\jdk1.8.0_191\bin\java.exe" ...  
Division Result 123.0  
Multiplication Result 0  
Substraction Result 123  
  
java.lang.ArithmeticException: / by zero  
  
    at Calculator.Sum(Calculator.java:31)  
    at CalculateTest.TestDiv(CalculateTest.java:23) <22 internal calls>  
  
Process finished with exit code -1
```
- Bottom Bar:** Terminal | 4: Run | 6: TODO | 0: Messages
- Status Bar:** Tests failed: 1, passed: 0 (a minute ago)

# Реалізація тестування мутаціями

```
public double Division(double a, double b) {  
    System.out.println("Division Result  "+ (a-b)); //Заменяем знак "/" на любой другой "*", "+", "-",  
    return (a/b); //получаем результат не "Infinity" -> тест не пройден, мутант убит  
}
```



The screenshot shows an IDE interface with a 'Run' tab active. The top status bar indicates 'Tests passed: 1 of 1 test - 0 ms'. The 'Run' tab displays a list of tests: 'CalculateTest' (0 ms) and 'TestDiv' (0 ms), both marked with green checkmarks. The output window shows the following text:

```
"C:\Program Files\Java\jdk1.8.0_191\bin\java.exe" ...  
Division Result  123.0  
Multiplication Result  0  
Substraction Result  123  
Sum Result  123  
  
Process finished with exit code 0
```

The bottom status bar shows 'Tests passed: 1 (3 minutes ago)'.



# Висновок

- В роботі була розглянута програмна модель калькулятора, за допомогою якого було відтворено мутаційне тестування, а саме проведено декілька тестів з використанням різних логічних та арифметичних операторів для отримання бажаного результату.
- У першому розділі розглядалися теоретичні відомості щодо тестування програмного забезпечення і необхідності його проведення.
- У другому розділі розглядалися практичні відомості, а саме методи реалізації мутаційного тестування, шляхом зміни логічних та арифметичних операторів, значення змінної або навіть її видалення. Також були розглянуті всі види результатів виконання мутацій вихідного коду, пов'язаних з перевіркою працездатності програмної моделі калькулятора.
- Таким чином, в роботі були порушені питання актуальності тестування програмного забезпечення цим методом та його надійності використання на практиці.

# СПИСОК ДЖЕРЕЛ ІНФОРМАЦІЇ

- 1 Jefferson O. A Practical System for Mutation Testing: Help for the Common Programmer. / 2016. – 437 с.
- 2 Weisfeld M. The Object-Oriented Thought Process / Matt Weisfeld. // Addison-Wesely. – 2013. – №4. – С. 306.
- 3 Available Mutators And Groups [Електронний ресурс] / Mutators – Режим доступу до ресурсу: <http://pitest.org/quickstart/mutators/>.
- 4 Frankl P./ All-uses versus mutation testing: An experimental comparison of effectiveness. Journal of Systems and Software, /. Weiss S., / Hu C. – 38:235-253, 1997.
- 5 Roger A., Mutation of Java Objects /. Bieman J./ Sudipto G/ Bixia J. – 2014 –№1 – С. 285.
- 6 Larman C. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development. / Creig Larman., 2004. – 627 с. – (Addison-Wesely).
- 7 Jeremy S. Mutation Operators for Concurrent Java (J2SE 5.0) / James R., Juergen D./ – С.142.
- 8 Yu-Seung M. / MuJava: An Automated Class Mutation System by / – С. 43 –, Jeff O., Yong R.