**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»**

Кафедра інформаційних систем та мереж

**Звіт**

**з дисципліни «Спеціалізовані мови програмування»**

**Лабораторна робота № 6**

**Виконав:**

студент групи ІТ-21сп

Масняк Б. В.

**Прийняв**:

Щербак С.С.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Львів – 2023

**Мета:** Cтворення юніт-тестів для додатка-калькулятора на основі класів

**Хід роботи**

Завдання 1: Тестування Додавання

Напишіть юніт-тест, щоб перевірити, що операція додавання в вашому додатку-калькуляторі працює правильно. Надайте тестові випадки як для позитивних, так і для негативних чисел.

def test\_results\_addition(self):  
 app = QApplication(sys.argv)  
 window = Lab2MainWindow()  
 centralwidget = QWidget()  
 window.setCentralWidget(centralwidget)  
 label\_result = QLabel(centralwidget)  
 label\_history = QLabel(centralwidget)  
 label\_result.setText("1+9")  
 results(label\_result, label\_history)  
 self.assertEqual((label\_result.text()), "10")

label\_result.setText("-11+9")  
 results(label\_result, label\_history)  
 self.assertEqual((label\_result.text()), "-2")

Завдання 2: Тестування Віднімання

Створіть юніт-тести для переконання, що операція віднімання працює правильно. Тестуйте різні сценарії, включаючи випадки з від'ємними результатами.

def test\_results\_subtraction(self):  
 app = QApplication(sys.argv)  
 window = Lab2MainWindow()  
 centralwidget = QWidget()  
 window.setCentralWidget(centralwidget)  
 label\_result = QLabel(centralwidget)  
 label\_history = QLabel(centralwidget)  
 label\_result.setText("8-5")  
 results(label\_result, label\_history)  
 self.assertEqual((label\_result.text()), "3")

label\_result.setText("-1-5")  
 results(label\_result, label\_history)  
 self.assertEqual((label\_result.text()), "-6")

Завдання 3: Тестування Множення

Напишіть юніт-тести, щоб перевірити правильність операції множення в вашому калькуляторі. Включіть випадки з нулем, позитивними та від'ємними числами.

def test\_root\_multiplication(self):  
 app = QApplication(sys.argv)  
 window = Lab2MainWindow()  
 centralwidget = QWidget()  
 window.setCentralWidget(centralwidget)  
 label\_result = QLabel(centralwidget)  
 label\_history = QLabel(centralwidget)  
 label\_result.setText("11\*9")  
 results(label\_result, label\_history)  
 self.assertEqual((label\_result.text()), "99")

Завдання 4: Тестування Ділення

Розробіть юніт-тести для підтвердження точності операції ділення. Тести повинні охоплювати ситуації, пов'язані з діленням на нуль та різними числовими значеннями.

def test\_results\_division(self):  
 app = QApplication(sys.argv)  
 window = Lab2MainWindow()  
 centralwidget = QWidget()  
 window.setCentralWidget(centralwidget)  
 label\_result = QLabel(centralwidget)  
 label\_result.setText("8/2")  
 label\_history = QLabel(centralwidget)  
 results(label\_result, label\_history)  
 self.assertEqual((label\_result.text()), "4.0")

Завдання 5: Тестування Обробки Помилок

Створіть юніт-тести, щоб перевірити, як ваш додаток-калькулятор обробляє помилки. Включіть тести для ділення на нуль та інших потенційних сценаріїв помилок. Переконайтеся, що додаток відображає відповідні повідомлення про помилки.

def test\_results\_division\_by\_zero(self):  
 app = QApplication(sys.argv)  
 window = Lab2MainWindow()  
 centralwidget = QWidget()  
 window.setCentralWidget(centralwidget)  
 label\_result = QLabel(centralwidget)  
 label\_result.setText("12/0")  
 label\_history = QLabel(centralwidget)  
 results(label\_result, label\_history)  
 self.assertEqual((label\_result.text()), "Error")

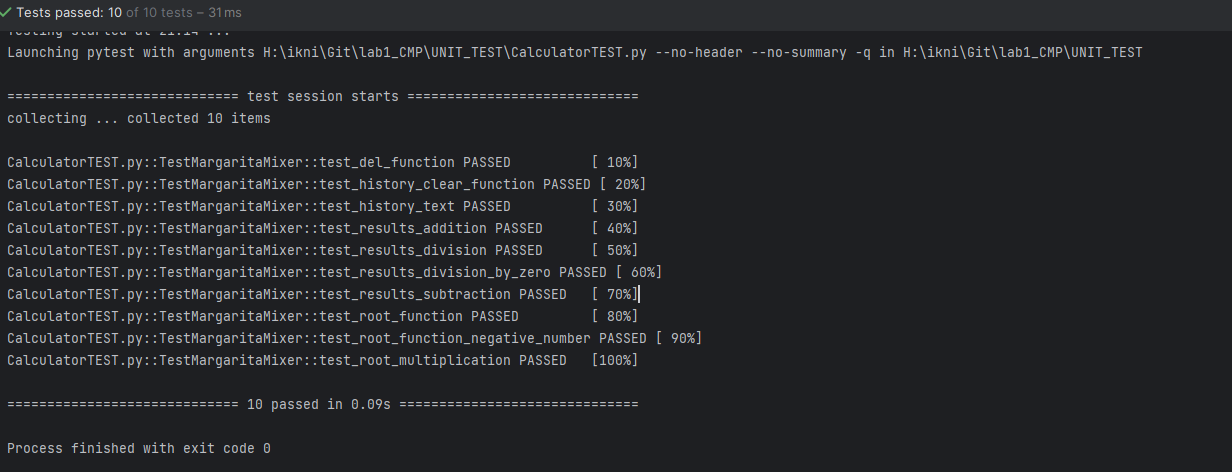
def test\_root\_function\_negative\_number(self):  
 app = QApplication(sys.argv)  
 window = Lab2MainWindow()  
 centralwidget = QWidget()  
 window.setCentralWidget(centralwidget)  
 label\_result = QLabel(centralwidget)  
 label\_result.setText("-9")  
 label\_history = QLabel(centralwidget)  
 root\_function(label\_result, label\_history)  
 self.assertEqual(label\_result.text(), "Error")  
 window.close()

Перевірка функції пам’яті:

def test\_history\_text(self):  
 app = QApplication(sys.argv)  
 window = Lab2MainWindow()  
 label\_history = window.label\_history  
 self.assertEqual(label\_history.text(), "Історія:\n")  
 window.close()  
  
def test\_history\_clear\_function(self):  
 app = QApplication(sys.argv)  
 window = Lab2MainWindow()  
 centralwidget = QWidget()  
 window.setCentralWidget(centralwidget)  
 label\_history = QLabel(centralwidget)  
 self.assertEqual(clear\_function(label\_history), "Історія:\n")  
 window.close()

Корінь квадратний:

def test\_root\_function(self):  
 app = QApplication(sys.argv)  
 window = Lab2MainWindow()  
 centralwidget = QWidget()  
 window.setCentralWidget(centralwidget)  
 label\_result = QLabel(centralwidget)  
 label\_result.setText("9")  
 label\_history = QLabel(centralwidget)  
 root\_function(label\_result, label\_history)  
 self.assertEqual(float(label\_result.text()), 3.0)  
 window.close()



**Висновок:** під час виконання цієї лабораторної роботи я створив юніт-тести для додатка-калькулятора на основі класів