МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
САМАРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Институт автоматики и информационных технологий
Кафедра «Информатика и вычислительная техника»

О Т Ч Е Т

о выполнении лабораторной работы №1

«Акторы и акторные системы. Создание простейшей акторной системы»

по дисциплине «Архитектура программного обеспечения»

Преподаватель				Скобелев П.О.
	(должность)	*(подпись)*	*(дата)*	*(инициалы, фамилия)*
Студент	1-ИАИТ-114М			Абалымов А.А.
	(группа)	*(подпись)*	*(дата)*	*(инициалы, фамилия)*
Студент	1-ИАИТ-114М			Желябин К.А.
	(группа)	*(подпись)*	*(дата)*	*(инициалы, фамилия)*

Самара 2025

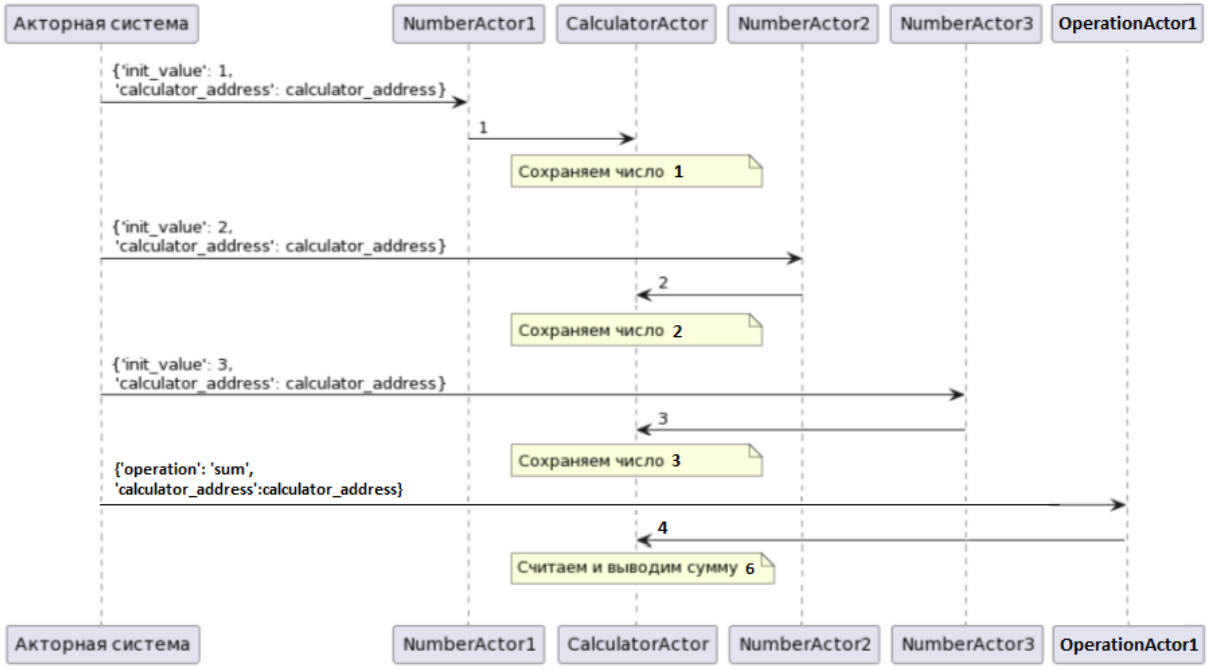
Самостоятельная работа:

В качестве самостоятельной работы предлагается расширить логику функционирования актора-калькулятора - реализовать в нем возможность не только суммирования, но и других арифметических действий. Для этого необходимо реализовать отправку еще одного типа сообщений и логику по его обработке.

Результаты работы:

Для расширения работы калькулятора был добавлен актор – OrepationActor, который хранит математическую операцию (произведение, сумма чисел, среднее значение, очистка списка, вывод списка) и отправляет сообщение актору калькулятор для выполнения математической операции.

В виде диаграммы последовательности это можно представить так:



Код актора OperationActor:

```
class OperationActor(Actor):
    def __init__(self):
        super().__init__()
        # В это поле будем сохранять полученное значение.
        self.operation = ''

    def receiveMessage(self, msg, sender):
        print(f'Актор операции с адресом {self.myAddress} получил {msg} от
{sender}')
        # Ожидаем, что в сообщении будет содержаться словарь со значением
математической операции и адресом актора-калькулятора.
        self.operation = msg['operation']
        calculator_address = msg.get('calculator_address')
        # Отправляем калькулятору свое значение.
        self.send(calculator_address, self.operation)
```

Код доработанного актора калькулятора:

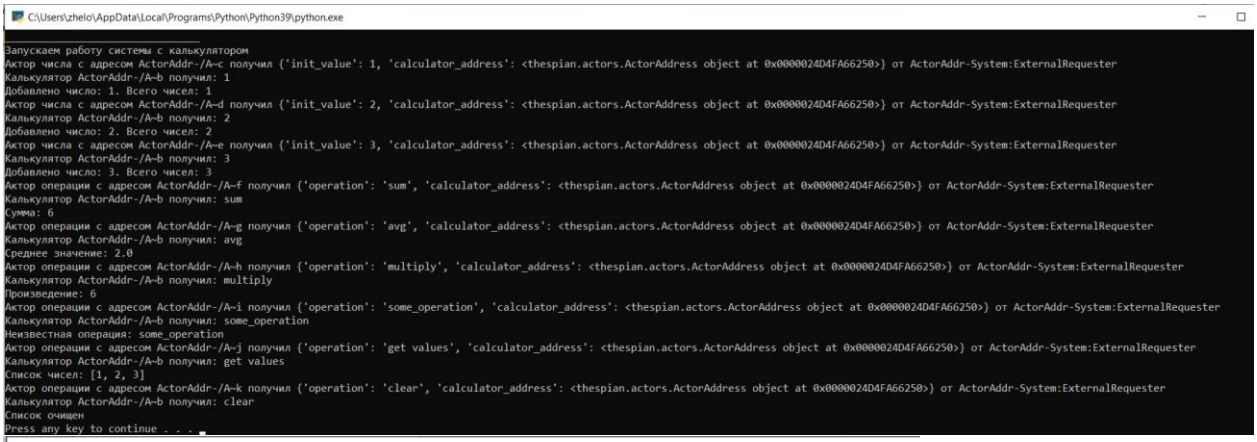
```
class CalculatorActor(Actor):
    def __init__(self):
        super().__init__()
        # Все полученные числа будем сохранять в списке.
        self.values = []

    def receiveMessage(self, msg, sender):
        print(f'Калькулятор {self.myAddress} получил: {msg}')

        if isinstance(msg, (int, float)):
            self.values.append(msg)
            print(f'Добавлено число: {msg}. Всего чисел: {len(self.values)}')

        elif isinstance(msg, str):
            if msg == 'sum':
                result = sum(self.values)
                print(f'Сумма: {result}')
            elif msg == 'multiply':
                result = 1
                for n in self.values: result *= n
                print(f'Произведение: {result}')
            elif msg == 'avg':
                avg = sum(self.values)/len(self.values) if self.values else 0
                print(f'Среднее значение: {avg}')
            elif msg == 'clear':
                self.values = []
                print('Список очищен')
            elif  msg == 'get values':
                print(f'Список чисел: {self.values}')
            else:
                print(f'Неизвестная операция: {msg}')
```

Проверка работы программы:



C:\Users\zhelo\AppData\Local\Programs\Python\Python39\python.exe
Запускаем работу системы с калькулятором
Актор числа с адресом ActorAddr-/A~c получил {'init_value': 1, 'calculato
Калькулятор ActorAddr-/A~b получил: 1
Добавлено число: 1. Всего чисел: 1
Актор числа с адресом ActorAddr-/A~d получил {'init_value': 2, 'calculato
Калькулятор ActorAddr-/A~b получил: 2
Добавлено число: 2. Всего чисел: 2
Актор числа с адресом ActorAddr-/A~e получил {'init_value': 3, 'calculato
Калькулятор ActorAddr-/A~b получил: 3
Добавлено число: 3. Всего чисел: 3
Актор операции с адресом ActorAddr-/A~f получил {'operation': 'sum', 'cal
Калькулятор ActorAddr-/A~b получил: sum
Сумма: 6
Актор операции с адресом ActorAddr-/A~g получил {'operation': 'avg', 'cal
Калькулятор ActorAddr-/A~b получил: avg
Среднее значение: 2.0
Актор операции с адресом ActorAddr-/A~h получил {'operation': 'multiply',
Калькулятор ActorAddr-/A~b получил: multiply
Произведение: 6
Актор операции с адресом ActorAddr-/A~i получил {'operation': 'some_opera
Калькулятор ActorAddr-/A~b получил: some_operation
Неизвестная операция: some_operation
Актор операции с адресом ActorAddr-/A~j получил {'operation': 'get values
Калькулятор ActorAddr-/A~b получил: get values
Список чисел: [1, 2, 3]
Актор операции с адресом ActorAddr-/A~k получил {'operation': 'clear', 'c
Калькулятор ActorAddr-/A~b получил: clear
Список очищен
Press any key to continue . . .

Код проверки работы акторов:

```
if __name__ == "__main__":
    # Создаем систему акторов, внутри которой они будут жить
    actorSystem = ActorSystem()
    # Создаем экземпляр созданного нами класса и сохраняем его адрес
    actorAddress1 = actorSystem.createActor(SimplestActor)
    # Отправляем по сохраненному адресу сообщение.
    actorSystem.tell(actorAddress1, "Первое сообщение")
    actorSystem.tell(actorAddress1, 'CREATE_ACTOR')
    actorSystem.tell(actorAddress1, 'RETRANSLATED MESSAGE')

    print('_____________________________________')
```

```
    print('Запускаем работу системы с калькулятором')

    # Создаем актор-калькулятор, сохраняем его адрес.
    calculator_address = actorSystem.createActor(CalculatorActor)

    # Создаем актор-число
    number_agent_1 = actorSystem.createActor(NumberActor)
    # Формируем сообщение со значением числа и адресом калькулятора.
    init_message_1 = {'init_value': 1, 'calculator_address':
calculator_address}

    # Отправляем актору числа сообщение, после которого он должен отправить
другое сообщение калькулятору.
    actorSystem.tell(number_agent_1, init_message_1)

    number_agent_2 = actorSystem.createActor(NumberActor)
    init_message_2 = {'init_value': 2, 'calculator_address':
calculator_address}
    actorSystem.tell(number_agent_2, init_message_2)

    number_agent_3 = actorSystem.createActor(NumberActor)
    init_message_3 = {'init_value': 3, 'calculator_address':
calculator_address}
    actorSystem.tell(number_agent_3, init_message_3)

    operation_agent_1 = actorSystem.createActor(OperationActor)
    operation_message_1 = {'operation': 'sum', 'calculator_address':
calculator_address}
    actorSystem.tell(operation_agent_1, operation_message_1)

    operation_agent_2 = actorSystem.createActor(OperationActor)
    operation_message_2 = {'operation': 'avg', 'calculator_address':
calculator_address}
    actorSystem.tell(operation_agent_2, operation_message_2)

    operation_agent_3 = actorSystem.createActor(OperationActor)
    operation_message_3 = {'operation': 'multiply', 'calculator_address':
calculator_address}
    actorSystem.tell(operation_agent_3, operation_message_3)

    operation_agent_4 = actorSystem.createActor(OperationActor)
    operation_message_4 = {'operation': 'some_operation',
'calculator_address': calculator_address}
    actorSystem.tell(operation_agent_4, operation_message_4)

    operation_agent_5 = actorSystem.createActor(OperationActor)
    operation_message_5 = {'operation': 'get values', 'calculator_address':
calculator_address}
    actorSystem.tell(operation_agent_5, operation_message_5)

    operation_agent_6 = actorSystem.createActor(OperationActor)
    operation_message_6 = {'operation': 'clear', 'calculator_address':
calculator_address}
    actorSystem.tell(operation_agent_6, operation_message_6)
```