Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота №5

З дисципліни: Технології розроблення програмного забезпечення

Тема: “Паттерн Command”

Виконав:

студент групи ІА-14

Рисаков Богдан

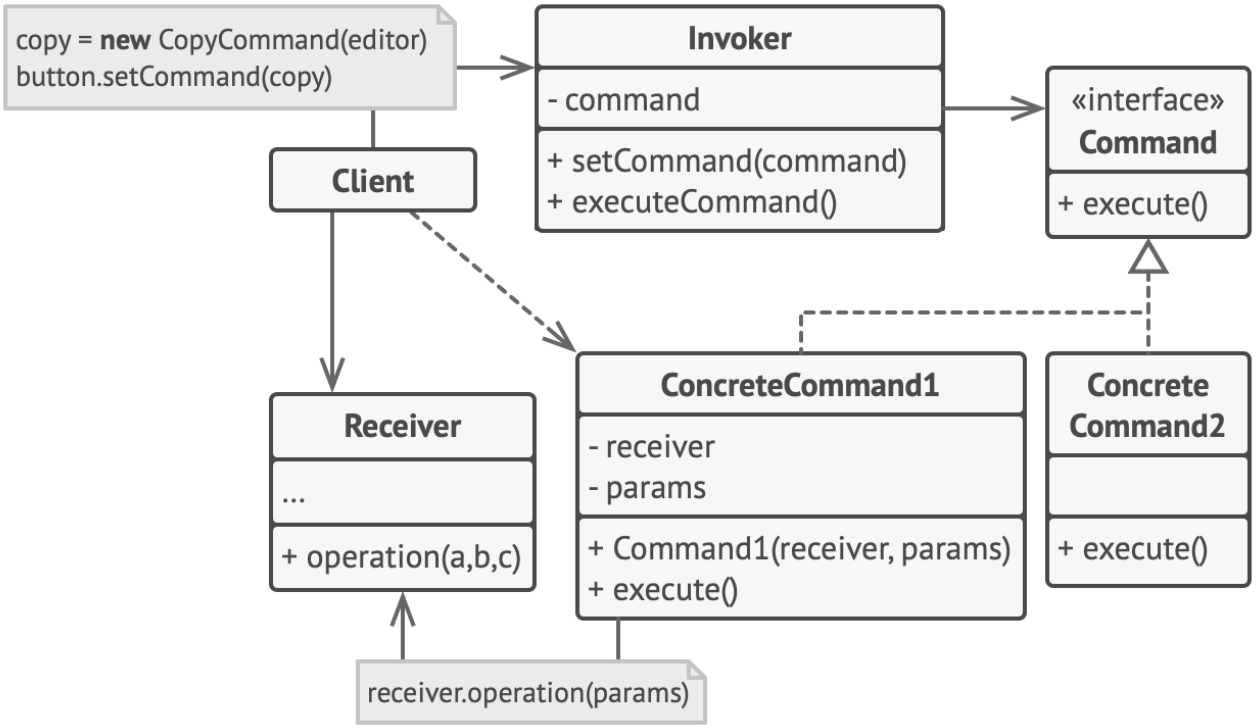
Перевірив:

Мягкий Михайло Юрійович

Київ, 2023

Мета: Реалізувати паттерн Command у проекті на тему System Activity Monitor

**Команда** — це поведінковий патерн проектування, який перетворює запити на об'єкти, дозволяючи передавати їх як аргументи під час виклику методів, ставити запити в чергу, логувати їх, а також підтримувати скасування операцій.




Приклад структури реалізації та використання патерну. Reference

**Переваги:**

- Прибирає пряму залежність між об'єктами, що викликають операції, та об'єктами, які їх безпосередньо виконують.
- Дозволяє реалізувати просте скасування і повтор операцій.
- Дозволяє реалізувати відкладений запуск операцій.
- Дозволяє збирати складні команди з простих.
- Реалізує *принцип відкритості/закритості*.

**Недоліки:**

- Ускладнює код програми внаслідок введення великої кількості додаткових класів.

Реалізація:

1) Загальний інтерфейс

```
public interface StartMonitoringCommand {

   void execute();

}
```

2)Конкретні команди

```
@Getter

@Setter

public class CpuLoadMonitoringCommand implements StartMonitoringCommand {



   private CpuLoadMonitoringService cpuLoadMonitoringService;



   private boolean isMonitoringStarted;



   public CpuLoadMonitoringCommand(CpuLoadMonitoringService
cpuLoadMonitoringService, boolean isMonitoringStarted) {

       this.cpuLoadMonitoringService = cpuLoadMonitoringService;

       this.isMonitoringStarted = isMonitoringStarted;

   }

   @Override

   public void execute() {

       cpuLoadMonitoringService.startMonitoring(isMonitoringStarted);

   }

}
```

```
@Getter

@Setter

public class KeyLoggerMonitoringCommand implements StartMonitoringCommand {



   private KeyLoggerMonitoringService keyLoggerMonitoringService;

   private boolean isMonitoringStarted;



   public KeyLoggerMonitoringCommand(KeyLoggerMonitoringService
keyLoggerMonitoringService, boolean isMonitoringStarted) {

       this.keyLoggerMonitoringService = keyLoggerMonitoringService;

       this.isMonitoringStarted = isMonitoringStarted;

   }

   @Override

   public void execute() {

       keyLoggerMonitoringService.startMonitoring(isMonitoringStarted);

   }

}







@Getter

@Setter

public class MemoryMonitoringCommand implements StartMonitoringCommand {



       private MemoryMonitoringService memoryMonitoringService;

       private boolean isMonitoringStarted;



       public MemoryMonitoringCommand(MemoryMonitoringService
memoryMonitoringService, boolean isMonitoringStarted) {

           this.memoryMonitoringService = memoryMonitoringService;
```

```
            this.isMonitoringStarted = isMonitoringStarted;

        }


        @Override

        public void execute() {

            memoryMonitoringService.startMonitoring(isMonitoringStarted);

        }

}
```

```
@Getter

@Setter

public class MouseTrackerMonitoringCommand implements StartMonitoringCommand {


    private MouseTrackerMonitoringService mouseTrackerMonitoringService;

    private boolean isMonitoringStarted;


    public MouseTrackerMonitoringCommand(MouseTrackerMonitoringService
mouseTrackerMonitoringService, boolean isMonitoringStarted) {

        this.mouseTrackerMonitoringService = mouseTrackerMonitoringService;

        this.isMonitoringStarted = isMonitoringStarted;

    }


    @Override

    public void execute() {

        mouseTrackerMonitoringService.startMonitoring(isMonitoringStarted);

    }

}
```

```java
@Setter

@Getter

public class WindowsMonitoringCommand implements StartMonitoringCommand {



   private WindowsMonitoringService windowsMonitoringService;

   private boolean isMonitoringStarted;



   public WindowsMonitoringCommand(WindowsMonitoringService
windowsMonitoringService, boolean isMonitoringStarted) {

       this.windowsMonitoringService = windowsMonitoringService;

       this.isMonitoringStarted = isMonitoringStarted;

   }



   @Override

   public void execute() {

       windowsMonitoringService.startMonitoring(isMonitoringStarted);

   }

}
```

Ці класи використовують конкретні сервіси для моніторінгу і фактично є їх вхідними точками.

Теперь будь який клас що має доступ до Команд може збирати конкретні метрики

Висновок: Я реалізував паттерн Команда