Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформаційні системи та технології

Лабораторна робота №5
Технології розроблення програмного забезпечення

Патерни проектування

Варіант 12 (CI server)

Виконав студент групи ІА-13:

Матусяк М.Ю.

Перевірив:

Мягкий М.Ю.

Київ 2023

Мета роботи: Розібратися в особливостях використання та застосувати в проекті паттерн Command.

Хід роботи:

За завданням необхідно реалізувати шаблон Command. Цей шаблон перетворює звичайний виклик методу в клас. Таким чином дії в системі стають повноправними об'єктами.

Реалізація даного шаблону полягає в наступному: пов'язані з командою поля, властивості, методи і дії виносяться в окремий загальний інтерфейс(Command):

```
interface Command {
   int execute(Project project);
}
```

Кожен команда являє собою окремий клас (CompileCommand, TestCommand), які реалізують загальний інтерфейс Command:

Клас CompileCommand визначає команду яка буде тільки компілювати вхідний проект та повертати результат успушності компіляції.

```
class CompileCommand implements Command {
   @Override
   public int execute(Project project) {
       return switch (project.getLanguage()) {
           case ("Java") -> javaCompile(project);
           case ("C#") -> csCompile(project);
           default -> 0;
       };
   }

   // compiling method for Java
   private int javaCompile(Project project) {
       System.out.println("Compiling for " + project.getLanguage());
       return 1;
       //return new Random().nextInt(2);
   }

   // compiling method for C#
   private int csCompile(Project project) {
       System.out.println("Compiling for " + project.getLanguage());
       return new Random().nextInt(2);
   }
}
```

Клас CompileCommand реалізує допоміжні методи для компіляції в залежності від мови програмування проекту. javaCompile та csCompile для мови програмування Java та C# відповідно.

Також є основний метод execute() який визначає мову програмування проекту та викликає один із допоміжних методів.

Клас TestCommand визначає команду яка буде тільки тестувати вхідний проект та повертати результат успушності тестування.

```
class TestCommand implements Command {
   @Override
   public int execute(Project project) {
       System.out.println("testing");
       return new Random().nextInt(2);
   }
}
```

TestCommand має один метод execute() для тестування проекту.

Висновок: на цій лабораторній роботі був використаний шаблон Command для того щоб при подальшому розширені проекту за потреби можна було легко додавати нові команди створивши клас який реалізує загальний інтерфейс Command. Також цей шаблон дозволяє збирати складні команди з простих.