Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота № 4

з дисципліни «Теорія розробки ПЗ»

Тема: ШАБЛОНИ «SINGLETON», «ITERATOR», «PROXY», «STATE», «STRATEGY»

Виконав:
студент групи ІА-12
Симко Андрій Ігорович
Дата здачі _____
Захищено з балом _____

Перевірив: Колеснік Валерій Миколайович

Київ, 2023

Тема: ШАБЛОНИ «SINGLETON», «ITERATOR», «PROXY», «STATE», «STRATEGY»

Хід роботи

Завдання.

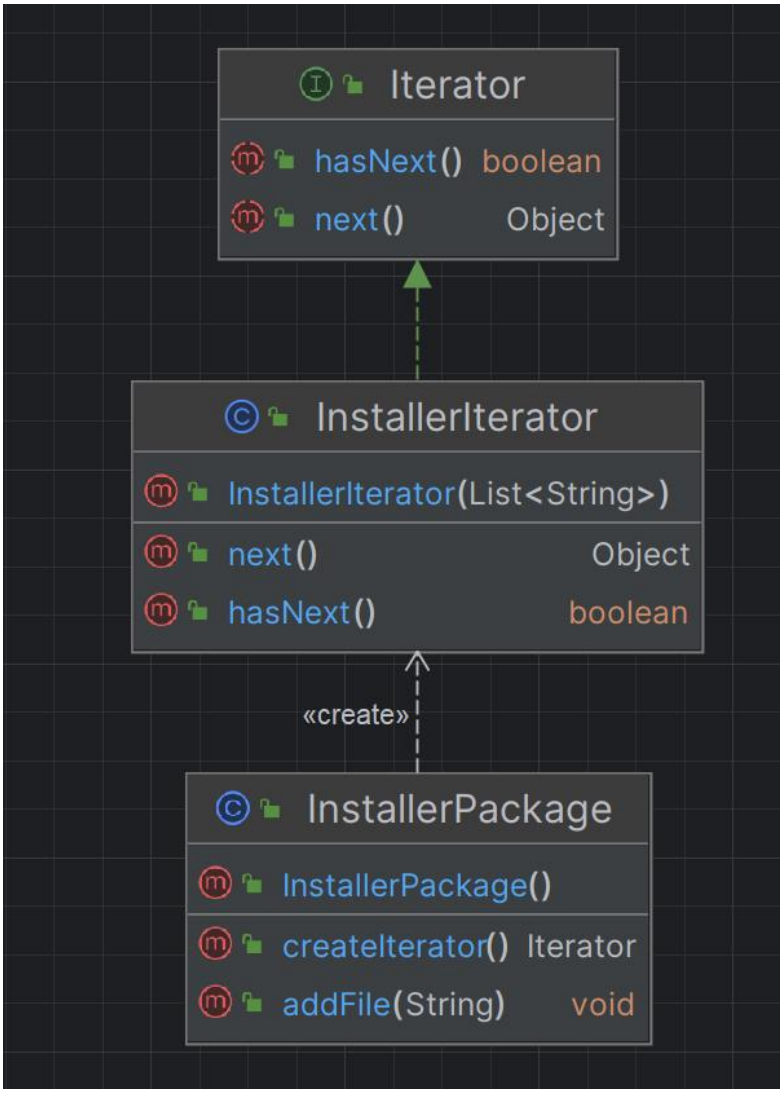
1. Ознайомитися з короткими теоретичними відомостями.

2. Проаналізуйте тему та намалюйте схему прецеденту, що відповідає обраній темі лабораторії.

..25 Installer generator (iterator, builder, factory method, bridge, interpreter, client-server)

Генератор інсталяційних пакетів повинен мати якийсь спосіб налаштування файлів, що входять в установку, установки вікон з інтерактивними можливостями (галочка - створити ярлик на робочому столі; ввести в текстове поле деякі дані, наприклад, ліцензійний ключ і т.д.). Генератор повинен вивести один файл .exe або .msi.

Структура реалізації ітератора



Приклад використання у проекті

```
public class Main {
    public static void main(String[] args) {
        InstallerPackage installerPackage = new InstallerPackage();
        installerPackage.addFile("file1.exe");
        installerPackage.addFile("file2.dll");

        Iterator iterator = installerPackage.createIterator();
        while (iterator.hasNext()) {
            String file = (String) iterator.next();
            System.out.println("Installing: " + file);
        }

    }
}
```

Класи

```
public interface Iterator {
    2 usages  1 implementation
    boolean hasNext();
    1 usage  1 implementation
    Object next();
}
```

```
public class InstallerPackage {
    3 usages
    private List<String> files;

    1 usage
    public InstallerPackage() { this.files = new ArrayList<>(); }

    2 usages
    public void addFile(String file) { files.add(file); }

    1 usage
    public Iterator createIterator() {
        return new InstallerIterator(files);
    }
}
```

```java
public class InstallerIterator implements Iterator {
    2 usages
    private int currentIndex = 0;
    3 usages
    private List<String> files;

    1 usage
    public InstallerIterator(List<String> files) { this.files = files; }

    2 usages
    @Override
    public boolean hasNext() { return currentIndex < files.size(); }

    1 usage
    @Override
    public Object next() {
        if (hasNext()) {
            return files.get(currentIndex++);
        }
        return null;
    }
}
```

Висновок: При виконані лабораторної роботи реалізував патерн ITERATOR.