Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут”
Кафедра АСОІУ

# ЗВІТ

про виконання лабораторної роботи
з дисципліни
“Об’єктно-орієнтоване програмування Java”
Тема: Ознайомлення з видами шаблонів проектування ПЗ. Вивчення структурних шаблонів. Отримання базових навичок з застосування шаблонів Composite, Decorator та Proxy.

Прийняв:
Подрубайло О. О.

Виконав:
студент 3-го курсу
гр. ІП-52 ФІОТ
Набоков Е.М

Київ 2017

# 1 ПОСТАНОВКА ЗАДАЧІ

1. Ознайомитись з призначенням та видами шаблонів проектування ПЗ. Вивчити класифікацію шаблонів проектування ПЗ. Знати назви шаблонів, що відносяться до певного класу.
2. Вивчити структурні шаблони проектування ПЗ. Знати загальну характеристику структурних шаблонів та призначення кожного з них.
3. Детально вивчити структурні шаблони проектування Composite, Decorator та Proxy.
4. В підготованому проекті (ЛР1) створити програмний пакет com.lablll.labwork3. В пакеті розробити інтерфейси і класи, що реалізують завдання (згідно варіанту) з застосуванням одного чи декількох шаблонів (п.З). В класах, що розробляються, повністю реалізувати методи, пов'язані з функціюванням Шаблону. Методи, що реалізують бізнес- логіку, закрити заглушками з виводом на консоль інформації про викликаний метод та його аргументи. Приклад реалізації бізнес-методу:

```
void draw(int x, int y){
  System.out.println("Метод draw з параметрами x="+x+" y="+y);
}
```

5. За допомогою автоматизованих засобів виконати повне документування розроблених класів (також методів і полів), при цьому документація має в достатній мірі висвітлювати роль певного класу в загальній структурі Шаблону та особливості конкретної реалізації.

**Варіант 4**. Визначити специфікації класів для подання файлової системи у вигляді дерева об'єктів (файл - листовий об'єкт, каталог - вузловий). Кожний об'єкт має атрибут розміру (для файлу задається в конструкторі, для каталогів обчислюється). Реалізувати бізнес-метод отримання розміру для класу каталогу.

# 2 UML ДІАГРАММА

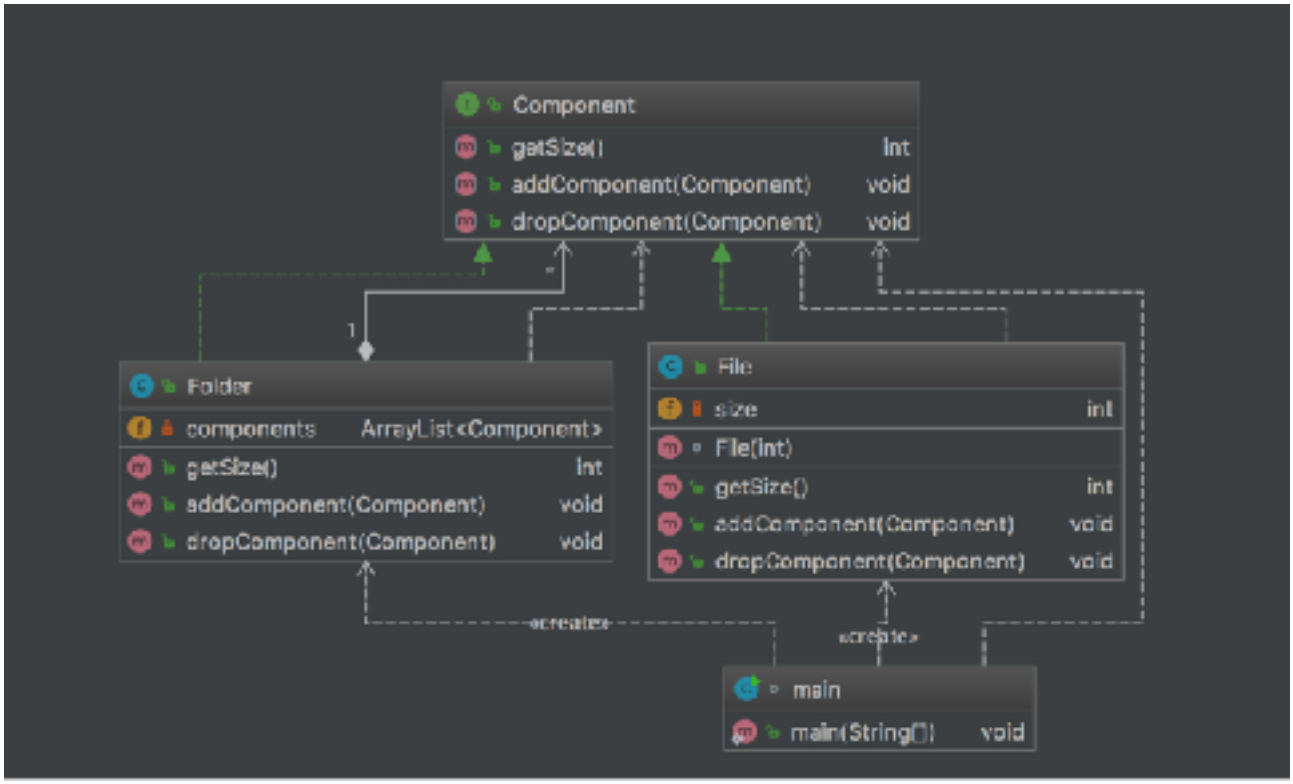


Рисунок 2.1 — схема (поля та конструктори)

# 3 ПРОГРАМНИЙ КОД

## *Component.java*

```
package com.solutions.labwork3;

/**
 * Interface Component.
 * It's common access to handle
 * multiple objects as one object.
 *
 * @author Eduard Nabokov
 * @version 0.1
 * @since 23.10.17
 */

public interface Component {

    /**
     * Calculate size or return given size
     * @return size (int)
     */
    int getSize();

    /**
     * Add component to the list
     * @param obj (Component)
     */
    void addComponent(Component obj);

    /**
     * Drop component from the list
     * @param obj (Component)
     */
    void dropComponent(Component obj);

}
```

## *File.java*

```
package com.solutions.labwork3;

/**
 * Class File represents structure
 * and methods of plain file object.
 * It implements Component interface.
 *
 * @author Eduard Nabokov
 * @version 0.1
 * @since 23.10.17
 */

public class File implements Component {

    private int size;

    File(int size) { this.size = size; }

    /**
     * Return given size of File
     * @return size (int)
     */
    @Override
    public int getSize() {
        return this.size;
    }

    /**
     * Add component to the list
     * @param obj (Component)
     */
    @Override
    public void addComponent(Component obj) {
        System.out.println("It's file. You cannot add anything.");
    }

    /**
     * Drop component from the list
     * @param obj (Component)
     */
    @Override
```

```
    public void dropComponent(Component obj) {
        System.out.println("It's file. You cannot add anything.");
    }
}
```

## *Folder.java*

```
package com.solutions.labwork3;
import java.util.ArrayList;

/**
 * Class Folder represents structure
 * of ordinary folder,
 * that consists of files.
 * It implements Component interface.
 *
 * @author Eduard Nabokov
 * @version 0.1
 * @since 23.10.17
 */

public class Folder implements Component {
    private ArrayList<Component> components = new ArrayList<>();
    /**
     * Calculate size of Folder
     * @return size (int)
     */
    @Override
    public int getSize() {
        int size = 0;
        for(Component component: components) {
            size += component.getSize();
        }
        return size;
    }

    /**
     * Add component to the list
     * @param obj (Component)
     */
    @Override
    public void addComponent(Component obj) { components.add(obj); }
```

```
    /**
     * Drop component from the list
     * @param obj (Component)
     */
    @Override
    public void dropComponent(Component obj) { components.remove(obj); }
}
```

***Main.java***

```
package com.solutions.labwork3;
/**
 * Class Main - starting point for project.
 * Represent usage of implemented
 * classes and methods.
 *
 * @author Eduard Nabokov
 * @version 0.1
 * @since 23.10.17
 */

class main {
    public static void main(String[] args) {
        Component filesystem = new Folder();

        Component folder1 = new Folder();
        filesystem.addComponent(folder1);

        folder1.addComponent(new File(10));
        folder1.addComponent(new File(15));

        Component folder2 = new Folder();
        Component file2 = new File(15);

        file2.addComponent(new File(17));
        folder2.addComponent(file2);
        filesystem.addComponent(folder2);

        filesystem.dropComponent(folder1);

        System.out.println(filesystem.getSize());
    }
}
```