**Інженерія програмного забезпечення**

Лабораторна робота №9

*«ШАБЛОНИ, ЩО ПОРОДЖУЮТЬ. ШАБЛОНИ ABSTRACT FACTORY, BUILDER»*

Виконав:

студент гр.ІО-31

Долинний О.В.

Перевірив:

Антонюк А.І.

Київ 2014

**Лабораторна робота №9**

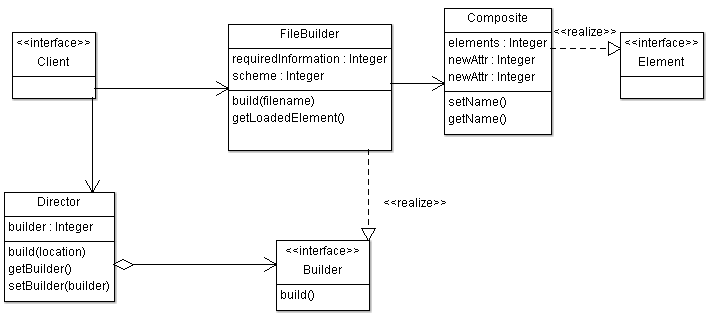
**Мета**: Вивчення шаблонів, що породжують. Отримання базових навичок з застосування шаблонів Abstract Factory та Builder.

1. ***Варіант завдання***

3110 mod 11 = 10

10. Визначити специфікації класів для подання елементів векторного графічного редактору (примітив і композит). Реалізувати можливість побудови композитного зображення на основі завантаженого файлу-специфікації.

1. ***Діаграма класів***

****

1. ***Результати тестування***

Reading required information from file

c:/sample.dat: building...

labwork9: Well done!

1. ***Програмний код***

**package** com.lab111.labwork9;

/\*\*

\* In this labwork I used pattern 'Builder'

\*

\* **@author** Dolinniy

\* **@version** 9.10

\*/

**public** **class** Client {

**public** **static** **void** main(String[] args) {

FileBuilder builder = **new** FileBuilder();

builder.build("labwork9");

Director director = **new** Director(builder);

director.build("c:/sample.dat");

Composite scheme0 = builder.getLoadedElement();

System.***out***.println("Scheme built: " + scheme0.getName());

}

}

**package** com.lab111.labwork9;

/\*\*

\* Class that rules of builder.

\*

\* **@author** Dolinniy

\* **@version** 9.10

\*/

**public** **class** Director {

/\*\*

\* Current builder.

\*/

**private** Builder builder;

/\*\*

\* Constructor

\*

\* **@param** builder

\* builder

\*/

**public** Director(Builder builder) {

**this**.builder = builder;

}

/\*\*

\* Launch constructing of object.

\*

\* **@param** location

\*/

**public** **void** build(String location) {

System.***out***.println(location + ": building...");

builder.build(location);

}

/\*\*

\* Get current builder

\*

\* **@return** builder

\*/

**public** Builder getBuilder() {

**return** builder;

}

/\*\*

\* Define current builder

\*

\* **@param** builder

\*/

**public** **void** setBuilder(Builder builder) {

**this**.builder = builder;

}

}

**package** com.lab111.labwork9;

/\*\*

\* Defines interface for builder.

\*

\* **@author** Dolinniy

\* **@version** 9.10

\*/

**public** **interface** Builder {

**void** build(String location);

}

**package** com.lab111.labwork9;

/\*\*

\* Class that gets information about building from external file.

\*

\* **@author** Dolinniy

\* **@version** 9.10

\*/

**public** **class** FileBuilder **implements** Builder {

/\*\*

\* Information from external file.

\*/

**private** String requiredInformation;

/\*\*

\* Built elements

\*/

**private** Composite scheme;

@Override

**public** **void** build(String fileName) {

System.***out***

.println("Reading required information from file " + fileName);

requiredInformation = "[sample required information]";

}

/\*\*

\* Getting built elemenet

\*

\* **@return** element

\*/

**public** Composite getLoadedElement() {

**return** scheme;

}

}

**package** com.lab111.labwork9;

/\*\*

\* Class that composes singular elements.

\*

\* **@author** Dolinniy

\* **@version** 9.10

\*/

**public** **class** Composite **implements** Element {

/\*\*

\* Elements

\*/

**public** Element[] elements;

**public** String name;

**public** **void** setName (String names) {

name = names;

}

**public** String getName() {

**return** name;

}

}

**package** com.lab111.labwork9;

/\*\*

\* Defines interface that analyze if element is difficult object or not.

\*

\* **@author** Dolinniy

\* **@version** 9.10

\*/

**public** **interface** Element {

}

**Висновок**: У ході лабораторної роботи я вивчив необхідні паттерни та реалізував один з них на прикладі завдання лабораторної роботи.