Національний технічний університет України

«Київський політехнічний інститут»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Лабораторна робота №9

З предмету: «Інженерія програмного забезпечення»

номер залікової книжки 2230

Виконав: студент групи ІО-22

Щербина М.В.

Київ 2013р.

1. Повторити шаблони поведінки для проектування ПЗ. Знати загальну характеристику шаблонів поведінки та призначення кожного з них.

2. Детально вивчити шаблони поведінки для проектування ПЗ Abstract Factory та Builder. Для кожного з них:

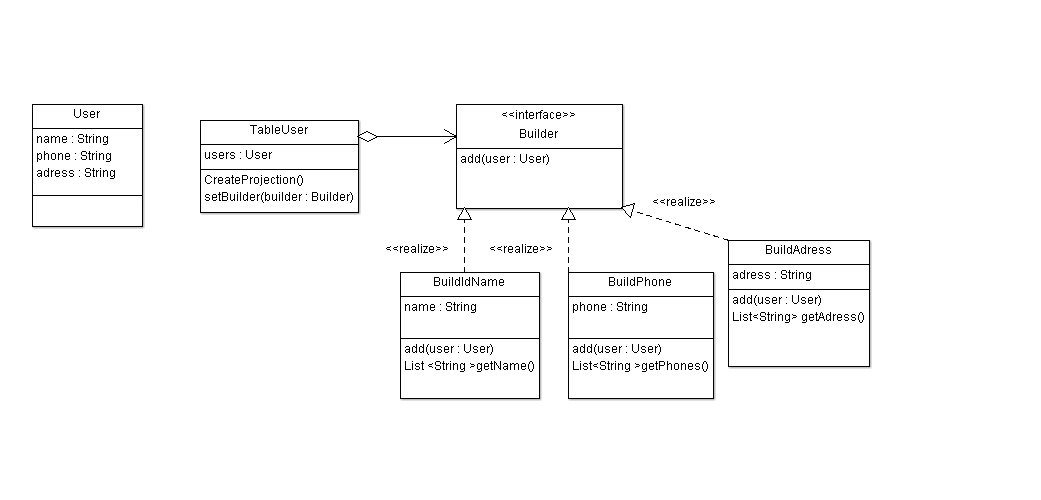
* вивчити Шаблон, його призначення, альтернативні назви, мотивацію, випадки коли його застосування є доцільним та результати такого застосування;
* знати особливості реалізації Шаблону, споріднені шаблони, відомі випадки його застосування в програмних додатках;
* вільно володіти структурою Шаблону, призначенням його класів та відносинами між ними;
* вміти розпізнавати Шаблон в UML діаграмі класів та будувати сирцеві коди Java-класів, що реалізують шаблон.

3. В підготованому проекті (ЛР1) створити програмний пакет com.lab111.labwork9. В пакеті розробити інтерфейси і класи, що реалізують завдання (згідно варіанту) з застосуванням одного чи декількох шаблонів (п.2). В розроблюваних класах 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.

4. 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 8

Визначити специфікації класів для подання реляційної таблиці та будівника проекції таблиць.

****

**package** com.lab111;

**import** java.util.List;

**import** com.lab111.labwork9.\*;

**public** **final** **class** TestMain {

/\*\*

\* Constructor.

\*

\*/

**private** TestMain() {

**super**();

}

/\*\*

\* Invokes at application startup.

\*

\* **@param** args

\* Parameters from command line

\*/

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

User[] users=**new** User[3];

users[0]=**new** User("Коля","343434","Пушкина 1");

users[1]=**new** User("Ваня","23-53-12","Янгеля 10");

users[2]=**new** User("Саня","+388888888","проспект победы 10");

BuildIdName builderName=**new** BuildIdName();

BuildPhone builderPhone=**new** BuildPhone();

BuildAdress builderAddress=**new** BuildAdress();

TableUser table=**new** TableUser(users, builderName);

System.*out*.println("Проекция имен");

table.CreateProjection();

List<String> name=builderName.getNames();

**for** (String string : name) {

System.*out*.println(string);

}

System.*out*.println("Проекция телефонов");

table.setBuilder(builderPhone);

table.CreateProjection();

List<String> phone=builderPhone.getPhones();

**for** (String string : phone) {

System.*out*.println(string);

}

System.*out*.println("Проекция адресов");

table.setBuilder(builderAddress);

table.CreateProjection();

List<String> address=builderAddress.getAdress();

**for** (String string : address) {

System.*out*.println(string);

}

}

}

**package** com.lab111.labwork9;

**public** **class** User {

**public** String name;

**public** String phone;

**public** String adress;

**public** User(String name, String phone, String adress) {

**super**();

**this**.name = name;

**this**.phone = phone;

**this**.adress = adress;

}

}**package** com.lab111.labwork9;

**public** **class** TableUser {

**public** TableUser(User[] users, Builder builder) {

**this**.users = users;

currentBuilder = builder;

}

**public** User[] users;

**public** Builder currentBuilder;

**public** **void** CreateProjection() {

**for** (**int** i = 0; i < users.length; i++) {

currentBuilder.add(users[i]);

}

}

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

currentBuilder = builder;

}

}**package** com.lab111.labwork9;

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

**public** **void** add(User user);

}**package** com.lab111.labwork9;

**import** java.util.ArrayList;

**import** java.util.List;

**public** **class** BuildPhone **implements** Builder {

**public** BuildPhone(){

phone=**new** ArrayList<String>();

}

**public** List<String> phone;

**public** **void** add(User user) {

phone.add(user.phone);

}

**public** List<String> getPhones() {

**return** phone;

}

}**package** com.lab111.labwork9;

**import** java.util.ArrayList;

**import** java.util.List;

**public** **class** BuildIdName **implements** Builder {

**public** BuildIdName() {

name = **new** ArrayList<String>();

}

**public** List<String> name;

**public** **void** add(User user) {

name.add(user.name);

}

**public** List<String> getNames() {

**return** name;

}

}**package** com.lab111.labwork9;

**import** java.util.ArrayList;

**import** java.util.List;

**public** **class** BuildAdress **implements** Builder {

**public** BuildAdress() {

adress = **new** ArrayList<String>();

}

**public** List<String> adress;

**public** List<String> getAdress() {

**return** adress;

}

@Override

**public** **void** add(User user) {

adress.add(user.adress);

}

}