Національний технічний університет України «Київський політехнічний інститут»
Кафедра обчислювальної техніки

Лабораторна робота №7

з дісципліни ***«Інженерія програмного забезпечення»***

Виконав:
студент 2 курсу
ФІОТ гр. ІО-32
Довгаль Д.С.
Залікова книжка №3211

Київ 2014 р.

Завдання

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 - Memento, State, Command та Interpreter. 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.

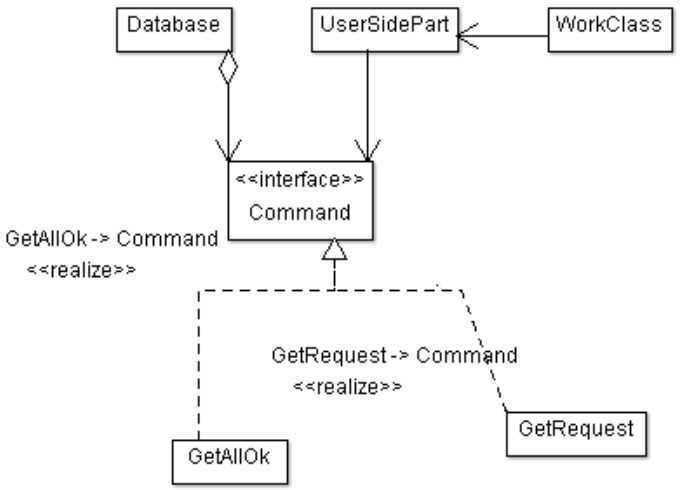
3. В підготованому проекті (ЛР1) створити програмний пакет com.lab111.labwork7. В пакеті розробити інтерфейси і класи, що реалізують завдання (згідно варіанту) з застосуванням одного чи декількох шаблонів (п.2). В розроблюваних класах повністю реалізувати методи, пов'язані з функціюванням Шаблону. Методи, що реалізують бізнес-логіку закрити заглушками з виводом на консоль інформації про викликаний метод та його аргументи. Приклад реалізації бізнес-методу:

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

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.

Варіанти (3211 mod 8)

3. Визначити специфікації класів, які подають операції над таблицю в БД. Реалізувати механізм організації транзакцій при виконанні операцій над таблицею.



```java
package lab111.labwork7;

/**
 * @author Error_404
 * @version 13.10.2014
 * Realize Command design pattern. This is the Command interface.
 */
public interface Command {

    /**
     * Part of realization Command design pattern. This method we will use on
     * each concrete Command.
     */
    void doSmth();
}
```

```java
package lab111.labwork7;

/**
 * @author Error_404
 * @version 13.10.2014
 * Realize Command design pattern. This is the concrete Command part,
 * which implements Command interface. Command which get request from DB.
 */
public class GetRequest implements Command {

    /**
     * Object, which method we will use in this command.
     */
    UserSidePart userSidePart;

    /**
     * Simple constructor
     * @param userSidePart Object, which method we will use in this command.
     */
    GetRequest(UserSidePart userSidePart){
        this.userSidePart= userSidePart;
    }

    @Override
    public void doSmth() {
        userSidePart.request();
    }
}
```

```java
package lab111.labwork7;

/**
 * @author Error_404
 * @version 13.10.2014
 * Realize Command design pattern. This is the concrete Command part,
 * which implements Command interface. Command which say "OK".
 */
public class GetAllOk implements Command {

    /**
     * Object, which method we will use in this command.
     */
    UserSidePart userSidePart;

    /**
     * Simple constructor
     * @param userSidePart Object, which method we will use in this command.
     */
    GetAllOk(UserSidePart userSidePart){
```

```
        this.userSidePart= userSidePart;
    }

    @Override
    public void doSmth() {
        userSidePart.sayOK();
    }
}
```

```
package lab111.labwork7;

/**
 * @author Error_404
 * @version 13.10.2014
 * Realize Command design pattern. This is the Receiver part.
 */
public class UserSidePart {

    /**
     * Just first operation, that say "All is ok on Server !".
     */
    public void sayOK(){
        System.out.println("All is ok on Server !");
    }

    /**
     * Just second operation, that say "Hello from DB !".
     */
    public void request(){
        System.out.println("Hello from DB !");
    }
}
```

```
package lab111.labwork7;

/**
 * @author Error_404
 * @version 13.10.2014
 * Realize Command design pattern. This is the Invoker part.
 */
public class Database {

    //two commands for later work
    private Command command1;
    private Command command2;

    Database(Command command1, Command command2){
        this.command1= command1;
        this.command2= command2;
    }

    /**
     * Do action in Command command1
     */
    public void doFirstCommand(){
        command1.doSmth();
    }

    /**
     * Do action in Command command2
     */
    public void doSecondCommand(){
        command2.doSmth();
    }
```

```
}
```

```
package lab111.labwork7;

/**
 * @author Error_404
 * @version 13.10.2014
 * Only workclass.
 */
public class WorkClass {
    public static void main(String[] args) {

        //create receiver
        UserSidePart usp= new UserSidePart();

        //create different commands
        Command getOK= new GetAllOk(usp);
        Command request= new GetRequest(usp);

        //create invoker
        Database db= new Database(getOK, request);

        //lets play !
        db.doFirstCommand();
        db.doSecondCommand();

    }
}
```