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

Лабораторна робота №4

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

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

Київ 2014 р.

Завдання

1. Закріпити призначення шаблонів проектування ПЗ, їх класифікацію. Знати назву і коротку характеристику кожного з шаблонів, що відносяться до певного класу.

2. Повторити структурні шаблони проектування ПЗ. Знати загальну характеристику структурних шаблонів та призначення кожного з них.

3. Детально вивчити структурні шаблони проектування Flyweight, Adapter, Bridge, Facade. 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.

4. В підготованому проекті (ЛР1) створити програмний пакет com.lab111.labwork4. В пакеті розробити інтерфейси і класи, що реалізують завдання (згідно варіанту) з застосуванням одного чи декількох шаблонів (п.3). 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);
}
```

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.

Варіанти (3211 mod 11)

10. Визначити специфікації класів, які подають об'єкти для маніпулювання елементами файлової системи - файлами та директоріями. Інтерфейс файлу містить методи open(String path, boolean createIfNotExist), close() та delete(String path) для відкриття, закриття та видалення файлу (при createIfNotExist==true файл буде створений, якщо він не існує або обрізаний до нульової довжини, якщо існує). Інтерфейс директорії містить методи create(String path), та rmdir(String path) для створення та видалення директорії. Задати підсистему з 3-ох файлів та 2-х директорій. Забезпечити можливість створення та видалення такої підсистеми через методи create(), destroy() та зміни структури підсистеми без впливу на її користувача.

```java
package lab111.labwork4;

/**
 * Интерфейс для работы с файлами. Содержит методы open, close, delete.
 *
 * @author Error_404
 */
public interface MyFileIf {

    /**
     *  Открывает файл с расположением path. Если файла не сушествует - создает его, или же
просто
     *  переписывает его и открывает.
     * @param path - расположение файла
     * @param createIfNotExist - существует ли файл
     */
    public void open(String path, boolean createIfNotExist);

    /**
     * Закрывает файл.
     */
    public void close();

    /**
     * Удаляет файл по пути path
     * @param path - путь удаляемого файла
     */
    public void delete(String path);


}
```

```java
package lab111.labwork4;

/**
 * Only workclass.
 * Realise pattern "Facade". Its a client part.
 *
 * @author Error_404
 */
public class WorkClass {
    public static void main(String[] args) {

        //создали проводник
        MyExplorer explorer= new MyExplorer();
        //создали иерархию
        explorer.create();
        System.out.println();
        //удалили ее
        explorer.destroy();

    }
}
```

```java
package lab111.labwork4;

/**
 * Класс для работы с папками, реализующий интерфейс MyDirectoryIf.
 *
 * @author Error_404
 */
public class MyDirectory implements MyDirectoryIf {

    @Override
    public void create(String path) {
```

```
        System.out.println("Directory "+ path+ " created");
    }

    @Override
    public void rmdir(String path) {
        System.out.println("Directory "+ path+ " deleted");
    }
}
```

```
package lab111.labwork4;

/**
 * Интерфейс для работы с папками. Содержит методы create, rmdir.
 *
 * @author Error_404
 */
public interface MyDirectoryIf {

    /**
     * Создает папу по пути path.
     * @param path -путь создаваемой папки
     */
    public void create(String path);

    /**
     * Удаляет папку по пути path.
     * @param path - путь удаляемой папки
     */
    public void rmdir(String path);

}
```

```
package lab111.labwork4;

/**
 * Собственно сам класс композиции.
 * Realise pattern "Facade". Its a farcade part.
 *
 * @author Error_404
 */
public class MyExplorer {

    //объекты файла и папки для последующей манипуляции
    private MyFile file;
    private MyDirectory directory;

    //простой конструктор дял создания проводника
    MyExplorer(){
        this.directory= new MyDirectory();
        this.file= new MyFile();
    }

    //создает нашу иерархию из 3-ох файлов и 2-х директорий
    public void create(){
        directory.create("/folder1/folder2");
        file.open("/folder1/file1.lol", true);
        file.open("/folder1/file2.lol", true);
        file.open("/folder1/folder2/file3.lol", true);
    }

    //удаляет нашу иерархию из 3-ох файлов и 2-х директорий
    public void destroy(){
        file.delete("/folder1/file1.lol");
        file.delete("/folder1/file2.lol");
        file.delete("/folder1/folder2/file3.lol");
```

```
        directory.rmdir("/folder1/folder2");
        directory.rmdir("/folder1");
    }
}
```

```
package lab111.labwork4;

/**
 * Класс для работы с файлами, реализующий интерфейс MyFileIf.
 *
 * @author Error_404
 */
public class MyFile implements MyFileIf{

    @Override
    public void open(String path, boolean createIfNotExist) {
        if (createIfNotExist) System.out.println("File "+ path+ " created");
        else System.out.println("File "+ path+ " rewrited");
        System.out.println("File "+ path+ " opened");
    }

    @Override
    public void close() {
        System.out.println("File closed");
    }

    @Override
    public void delete(String path) {
        System.out.println("File "+ path+ " deleted");
    }
}
```