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

Лабораторна робота №8

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

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

Київ 2014 р.

Завдання

1. Вивчити шаблони, що породжують. Знати загальну характеристику шаблонів, що породжують та призначення кожного з них.

2. Детально вивчити шаблони, що породжують - Prototype, Singleton та Factory Method. 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.

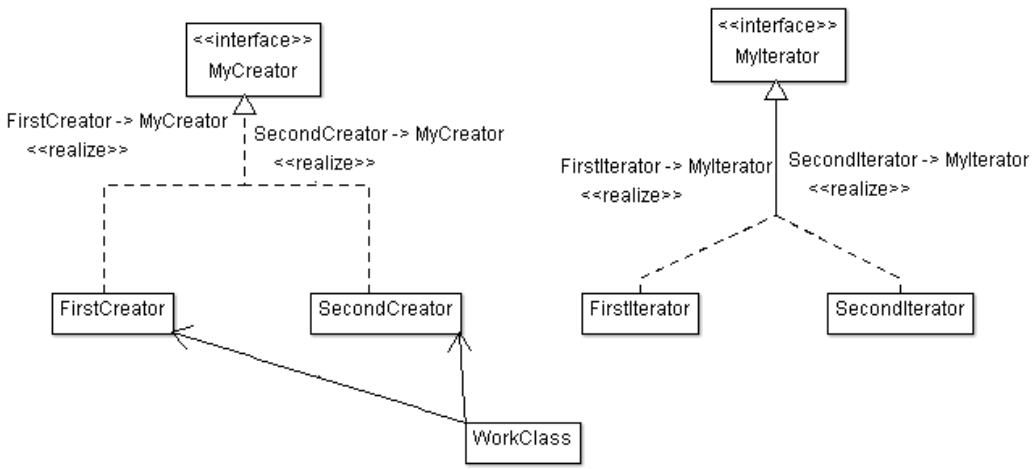
3. В підготованому проекті (ЛР1) створити програмний пакет com.lab111.labwork8. В пакеті розробити інтерфейси і класи, що реалізують завдання (згідно варіанту) з застосуванням одного чи декількох шаблонів (п.2). В розроблюваних класах повністю реалізувати методи, пов'язані з функціюванням Шаблону. Методи, 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 11)

10. Визначити специфікації класів для реалізації агрегату та його ітератору, який реалізує можливість зміни алгоритму пошуку наступного елементу під час виконання програми.



```java
package lab111.labwork8;

/**
 * @author Error_404
 * @version 13.10.2014
 * Realize Factory design pattern. This is the interface for objects which we will create.
 */
public interface MyIterator {
    /**
     * Simple method for iterator.
     */
    public void searchNextEllement();
}
```

```java
package lab111.labwork8;

/**
 * @author Error_404
 * @version 13.10.2014
 * Realize Factory design pattern. This is the first type of objects which we will create.
 */
public class FirstIterator implements MyIterator {
    @Override
    public void searchNextEllement() {
        System.out.println("First iterator search algorithm");
    }
}
```

```java
package lab111.labwork8;

/**
 * @author Error_404
 * @version 13.10.2014
 * Realize Factory design pattern. This is the second type of objects which we will create.
 */
public class SecondIterator implements MyIterator {
    @Override
    public void searchNextEllement() {
        System.out.println("Second iterator search algorithm");
    }
}
```

```java
package lab111.labwork8;

/**
 * @author Error_404
 * @version 13.10.2014
 * Realize Factory design pattern. This is the interface for our factory.
 */
public interface MyCreator {
    /**
     * Create our new object.
     * @return object we must have.
     */
    MyIterator createIterator();
}
```

```java
package lab111.labwork8;
```

```java
/**
 * @author Error_404
 * @version 13.10.2014
 * Realize Factory design pattern. This is the first factory creator.
 */
public class FirstCreator implements MyCreator {
    @Override
    public MyIterator createIterator() {
        return new FirstIterator();
    }
}
```

```java
package lab111.labwork8;

/**
 * @author Error_404
 * @version 13.10.2014
 * Realize Factory design pattern. This is the first factory creator.
 */
public class SecondCreator implements MyCreator {
    @Override
    public MyIterator createIterator() {
        return new SecondIterator();
    }
}
```

```java
package lab111.labwork8;

import java.util.Random;

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

        //create massive for some experiments
        Object[] mas=  new Object[20];
        MyCreator mc;

        //lets try our Factory in work
        for (Object ob: mas){
            if (new Random().nextInt()>0.25) mc= new FirstCreator();
            else mc= new SecondCreator();
            mc.createIterator().searchNextEllement();
        }
    }
}
```