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

Лабораторна робота №6

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 — Strategy, Chain of Responsibility та Visitor. 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.

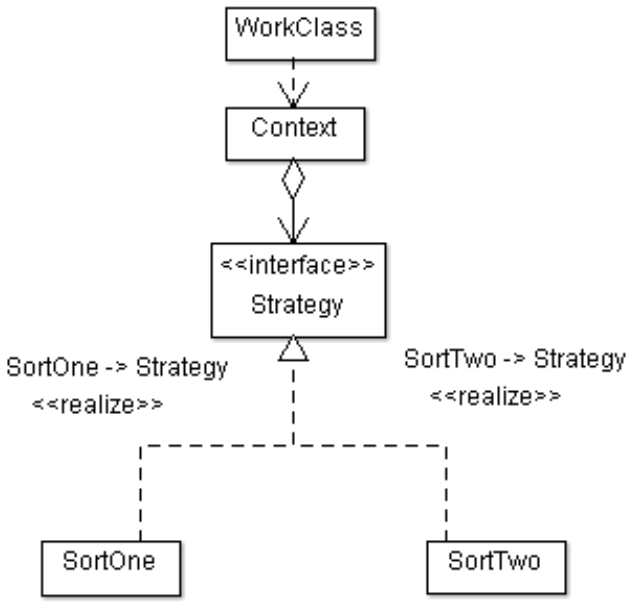
3. В підготованому проекті (ЛР1) створити програмний пакет com.lab111.labwork6. В пакеті розробити інтерфейси і класи, що реалізують завдання (згідно варіанту) з застосуванням одного чи декількох шаблонів (п.2). В розроблюваних класах повністю реалізувати методи, пов'язані з функціюванням Шаблону. Методи, 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 10)

1. 1. Визначити специфікації класу, який містить масив цілих чисел та метод його сортування. Забезпечити можливість динамічної зміни алгоритму та напрямку сортування шляхом зовнішньої параметризації.



```java
package lab111.labwork6;

/**
 * Класс реализующий конкретную стратегию, должен наследовать этот интерфейс
 * Класс контекста использует этот интерфейс для вызова конкретной стратегии
 *
 * @author Error_404
 */
public interface Strategy {

    /**
     * Сотрирует массив по определенному алгоритму
     *
     * @param mas - сортируемый массив
     */
    public void sort(Object[] mas);

}
```

```java
package lab111.labwork6;

/**
 * Реализуем алгоритм с использованием интерфейса стратегии
 *
 * @author Error_404
 */
public class SortOne implements Strategy {

    @Override
    public void sort(Object[] mas) {
        System.out.println("Отсортировано первым");
    }

}
```

```java
package lab111.labwork6;

/**
 * Реализуем алгоритм с использованием интерфейса стратегии
 *
 * @author Error_404
 */
public class SortTwo implements Strategy{

    @Override
    public void sort(Object[] mas) {
        System.out.println("Отсортировано вторым");
    }
}
```

```java
package lab111.labwork6;

/**
 * Класс контекста использующий интерфейс стратегии
 *
 * @author Error_404
 */
public class Context {

    // Хранимый алгоритм
    private Strategy strategy;
```

```
    // Constructor
    public Context() {
    }

    /**
     * Set new strategy
     */
    public void setStrategy(Strategy strategy) {
        this.strategy = strategy;
    }

    /**
     * Do the strategy algorithm
     */
    public void sortByStrategy() {
        strategy.sort(new Object[5]);
    }

}
```

```
package lab111.labwork6;

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

        context.setStrategy(new SortOne());
        context.sortByStrategy();

        context.setStrategy(new SortTwo());
        context.sortByStrategy();
    }
}
```