Національний технічний університет України

«Київський політехнічний інститут»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Лабораторна робота №6

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 — Strategy, Chain of Responsibility та Visitor. Для кожного з них:

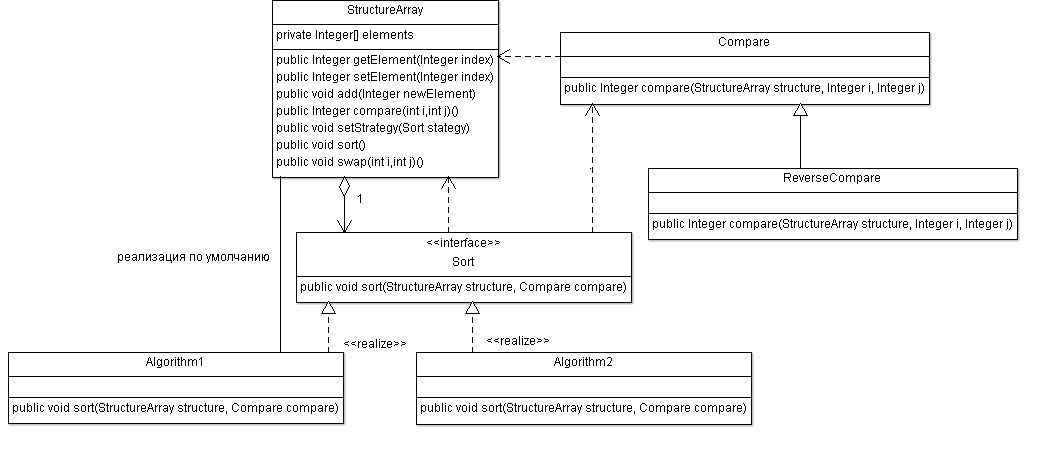
* вивчити Шаблон, його призначення, альтернативні назви, мотивацію, випадки коли його застосування є доцільним та результати такого застосування;
* знати особливості реалізації Шаблону, споріднені шаблони, відомі випадки його застосування в програмних додатках;
* вільно володіти структурою Шаблону, призначенням 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.

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 0

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



**package** com.lab111;

**import** com.lab111.labwork6.\*;

**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) {

Compare compare = **new** Compare();

ReverseCompare reversCompare = **new** ReverseCompare();

Algorithm2 algoriyhm2 = **new** Algorithm2();

StructureArray array = **new** StructureArray();

array.sort(compare);

array.setStrategy(algoriyhm2);

array.sort(reversCompare);

}

}

**package** com.lab111.labwork6;

**public** **class** Algorithm1 **implements** Sort {

**public** **void** sort(StructureArray structure, Compare compare) {

System.*out*.println("алгоритм 1");

System.*out*.println(compare.toString());

}

}

**package** com.lab111.labwork6;

**public** **class** Algorithm2 **implements** Sort {

**public** **void** sort(StructureArray structure, Compare compare) {

System.*out*.println("алгоритм 2");

System.*out*.println(compare.toString());

}

}

**package** com.lab111.labwork6;

**public** **class** Compare {

**public** **int** compare(SortStructure structure, Integer i, Integer j) {

**return** structure.compare(i, j);

}

@Override

**public** String toString() {

**return** "прямое сравнение";

}

}

**package** com.lab111.labwork6;

**public** **class** ReverseCompare **extends** Compare {

**public** **int** compare(SortStructure structure, Integer i, Integer j) {

**return** -1 \* structure.compare(i, j);

}

@Override

**public** String toString() {

**return** "обратное сравнение";

}

}

**package** com.lab111.labwork6;

/\*\*

\* интерфейс стратегий

\*

\*/

**public** **interface** Sort {

**public** **void** sort(StructureArray structure, Compare compare);

}

**package** com.lab111.labwork6;

/\*\*

\* массив

\*

\*/

**public** **class** StructureArray {

**public** StructureArray() {

strategySort = **new** Algorithm1();// реализация по умолчанию;

}

**private** Sort strategySort;

**private** **int**[] elements;

**public** **int** getElement(**int** index) {

**return** elements[index];

}

**public** **void** setElement(**int** index, **int** value) {

elements[index] = value;

}

**public** **void** add(Integer newElement) {

}

**public** **int** compare(**int** i, **int** j) {

**int** result = 0;

**if** (i > j)

result = 1;

**else** **if** (i < j) {

result = -1;

}

System.*out*.println("сравнение в StructureArray");

**return** result;

}

**public** **int** length() {

**return** elements.length;

}

**public** **void** swap(**int** i, **int** j) {

**int** k = elements[i];

elements[i] = elements[j];

elements[j] = k;

}

**public** **void** setStrategy(Sort stategy) {

strategySort = stategy;

}

**public** **void** sort(Compare compare) {

strategySort.sort(**this**, compare);

}

}