**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**Національний Технічний Університет України**

**«Київський Політехнічний Інститут»**

*Факультет інформатики та обчислювальної техніки*

*Кафедра обчислювальної техніки*

Лабораторна робота №6

*з дисципліни «Інженерія програмного забезпечення»*

*на тему: «Шаблони поведінки - Шаблони Strategy, Chain of Responsibility та Visitor»*

ВИКОНАВ:

студент ІІ курсу ФІОТ

групи ІО-71

Чеботаренко Артем

Залікова - 7127

Варіант – 4

ПЕРЕВІРИВ:

Доцент

к.т.н., с.н.с.

Антонюк А.І.

Київ – 2018

**Мета:** Вивчення шаблонів поведінки. Отримання базових навичок з застосування шаблонів Strategy, Chain of Responsibility та Visitor.

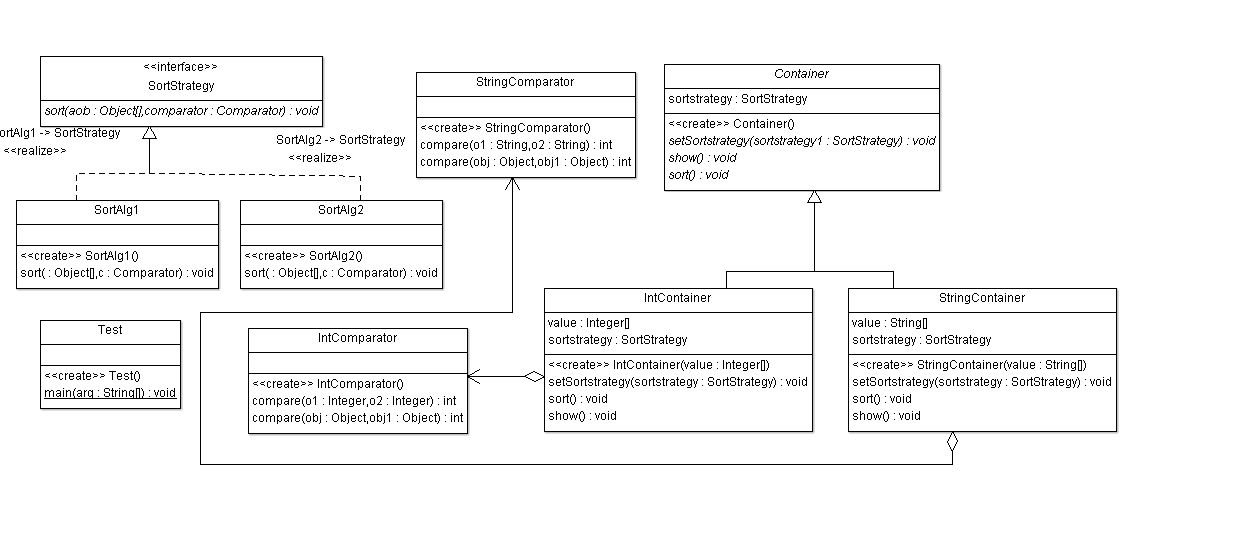
**I. Завдання**

**Визначення варіанту завдання:**

**Номер залікової книжки:** 7127;

**Варіант:** 7127 *mod* 10 = **3** – *Визначити специфікації класів, що реалізують контейнери для цілих чисел та текстових строк з можливістю їх сортування. Забезпечити можливість динамічної зміни алгоритму сортування шляхом зовнішньої параметризації. Реалізація алгоритму сортування має бути незалежною від типу даних, що сортуються.*

**ІI. Діаграма класів**

**

**IIІ. Код програми**

package com.lab111.labwork6;

import java.util.Comparator;

/\*\*

\* Class comparator for int

\* @author Taras Barabash

\* @version 6.3

\*/

public class IntComparator implements Comparator {

/\*\*

\* Constructor of this class

\*/

public IntComparator() {

}

/\*\*

\* Method for compare

\* @param o1

\* @param o2

\* @return

\*/

public int compare(Integer o1, Integer o2) {

if (o1.intValue() > o2.intValue())

return 1;

return o1.intValue() >= o2.intValue() ? 0 : -1;

}

public int compare(Object obj, Object obj1) {

return compare((Integer) obj, (Integer) obj1);

}

}

package com.lab111.labwork6;

import java.io.PrintStream;

/\*\*

\* Class Int container which extends container

\* @author Taras Barabash

\* @version 6.3

\*/

public class IntContainer extends Container {

/\*\*

\* Constructor of this class

\* @param values

\*/

public IntContainer(Integer values[]) {

this.values = new Integer[100];

this.values = values;

}

/\*

\* Setters for sort strategy

\*/

public void setSortstrategy(SortStrategy sortstrategy) {

this.sortstrategy = sortstrategy;

}

/\*

\* Method for sort

\*/

public void sort() {

sortstrategy.sort(values, new IntComparator());

}

/\*

\* Method for print

\*/

public void show() {

for (int i = 0; i < values.length; i++)

System.out.print((new StringBuilder()).append(values[i])

.append(", ").toString());

System.out.println();

}

protected Integer values[];

SortStrategy sortstrategy;

}

package com.lab111.labwork6;

import java.io.PrintStream;

import java.util.Arrays;

import java.util.Comparator;

/\*\*

\* Class Sort Algorithm 1

\* @author Taras Barabash

\* @version 6.3

\*/

public class SortAlg1 implements SortStrategy {

/\*\*

\* Constructor of this class

\*/

public SortAlg1() {

}

/\*

\* method for sorting

\*/

public void sort(Object a[], Comparator c) {

Arrays.sort(a, c);

System.out.println("Sorting by first algorithm");

}

}

package com.lab111.labwork6;

import java.io.PrintStream;

import java.util.Arrays;

import java.util.Comparator;

/\*\*

\* Class Sort Algorithm 2

\* @author Taras Barabash

\* @version 6.3

\*/

public class SortAlg2 implements SortStrategy {

/\*\*

\* Constructor of this class

\*/

public SortAlg2() {

}

/\*

\* method for sorting

\*/

public void sort(Object a[], Comparator c) {

Arrays.sort(a, c);

System.out.println("Sorting by second algorithm");

}

}

package com.lab111.labwork6;

import java.util.Comparator;

/\*\*

\* This is sort strategy class

\* @author Taras Barabash

\* @version 6.3

\*/

public interface SortStrategy

{

/\*\*

\* sort method

\* @param aobj

\* @param comparator

\*/

public abstract void sort(Object aobj[], Comparator comparator);

}

package com.lab111.labwork6;

import java.util.Comparator;

/\*\*

\* Class comparator for sorting String

\* @author Taras Barabash

\* @version 6.3

\*/

public class StringComparator implements Comparator {

public StringComparator() {

}

/\*\*

\* @param o1

\* @param o2

\* @return compare

\*/

public int compare(String o1, String o2) {

return o1.compareTo(o2);

}

/\* (non-Javadoc)

\* @see java.util.Comparator#compare(java.lang.Object, java.lang.Object)

\*/

public int compare(Object obj, Object obj1) {

return compare((String) obj, (String) obj1);

}

}

package com.lab111.labwork6;

import java.io.PrintStream;

/\*\*

\* The class stringcontainer which extends container

\* @author Taras Barabash

\* @version 6.3

\*/

public class StringContainer extends Container {

/\*\*

\* Constructor of this class

\* @param values

\*/

public StringContainer(String values[]) {

this.values = new String[100];

this.values = values;

}

/\*

\* Seter

\*/

public void setSortstrategy(SortStrategy sortstrategy) {

this.sortstrategy = sortstrategy;

}

/\*

\* method which sorted

\*/

public void sort() {

sortstrategy.sort(values, new StringComparator());

}

/\*

\* method for print

\*/

public void show() {

for (int i = 0; i < values.length; i++)

System.out.print((new StringBuilder(String.valueOf(values[i])))

.append(", ").toString());

System.out.println();

}

protected String values[];

SortStrategy sortstrategy;

}

package com.lab111.labwork6;

/\*\*

\* @author Taras Barabash

\* @version 6.3

\*/

public abstract class Container

{

SortStrategy sortstrategy;

public abstract void setSortstrategy(SortStrategy paramSortStrategy);

public abstract void show();

public abstract void sort();

}

package com.lab111.labwork6;

/\*\*

\* Class Main which tested all

\* @author Taras Barabash

\* @version 6.3

\*/

public class Test {

public Test() {

}

/\*\*

\* main

\* @param args

\*/

public static void main(String args[]) {

Integer a[] = { Integer.valueOf(1), Integer.valueOf(3),

Integer.valueOf(5), Integer.valueOf(2), Integer.valueOf(7) };

String b[] = { "ab", "we", "to" };

StringContainer con2 = new StringContainer(b);

IntContainer con = new IntContainer(a);

con2.setSortstrategy(new SortAlg1());

con.setSortstrategy(new SortAlg2());

con2.sort();

con2.show();

con.sort();

con.show();

}

}

**ІV. Висновок**

Ознайомились з видами шаблонів проектування програмного забезпечення. Розроблена відповідна тестова програма. Результати успішної роботи тестової програми наведені нижче підтверджують правильність обраних рішень.

Sorting by first algorithm

ab, to, we,

Sorting by second algorithm

1, 2, 3, 5, 7,