Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут ім. Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Лабораторна робота №5

*з дисципліни «Інженерія програмного забезпечення»*

*на тему: «Шаблони поведінки. Шаблони Iterator, Mediator, Observer»*

ВИКОНАВ:

студент ІІ курсу ФІОТ

групи ІО-71

Чеботаренко Артем

Залікова - 7127

Варіант – 2

ПЕРЕВІРИВ:

Доцент

к.т.н., с.н.с.

Антонюк А.І.

Київ – 2018

**Мета:** Вивчення шаблонів поведінки. Отримання базових навичок з застосування шаблонів Iterator, Mediator та Observer.

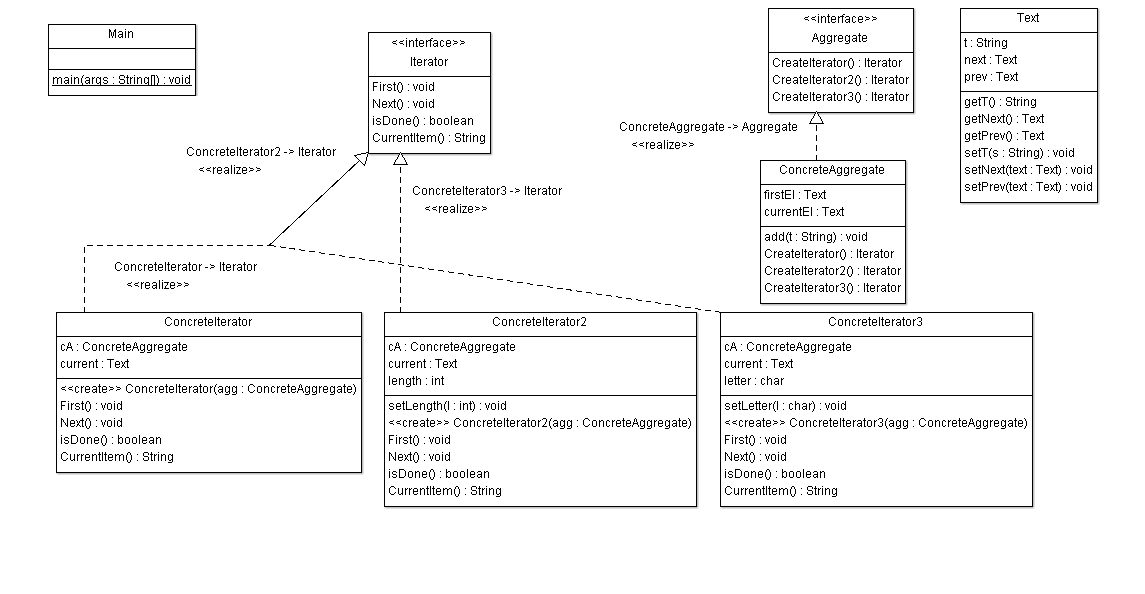
**I. Завдання**

**Визначення варіанту завдання:**

**Номер залікової книжки: 7127**;

**Варіант:** 7127 *mod* 9 = **2** – *Визначити специфікації класів які інкапсулюють лінійний список символьних рядків та реалізують можливість звичайного послідовного обходу та обходу з додатковою фільтрацію агрегату (Наприклад фільтрація по довжині рядка, по його першій літері, тощо).*

**ІI. Діаграма класів**



**IIІ. Код програми**

/\*\*

\* Main class that represents the work of the Iterator design pattern

\*/

package com.lab111.labwork5;

/\*\*

\* @author Taras Barabash

\* @version 5.2

\*/

public class Main {

public static void main(String [] args){

/\*\*

\* A new aggregate is created

\*/

ConcreteAggregate list = new ConcreteAggregate();

/\*\*

\* Filling the list

\*/

list.add("first");

list.add("second");

list.add("third");

list.add("fourth");

list.add("fifth");

list.add("sixth");

list.add("seventh");

list.add("eighth");

list.add("ninth");

/\*\*

\* Creating of a simple iterator, looking over some elements

\* and printing the value of the last reached element

\*/

ConcreteIterator iter = new ConcreteIterator(list);

iter.First();

iter.Next();

iter.Next();

iter.Next();

iter.Next();

System.out.println("One of the elements of a list: " + iter.CurrentItem());

/\*\*

\* Creating of an iterator with a filter by string length, looking over some elements

\* and printing the value of the last reached element

\*/

ConcreteIterator2 iter2 = new ConcreteIterator2(list);

iter2.setLength(7);

iter2.First();

iter2.Next();

System.out.println("One of the elements of a list filtered by string length: " + iter2.CurrentItem());

/\*\*

\* Creating of an iterator with a filter by the first letter, looking over some elements

\* and printing the value of the last reached element

\*/

ConcreteIterator3 iter3 = new ConcreteIterator3(list);

iter3.setLetter('s');

iter3.First();

System.out.println("One of the elements of a list filtered by the first letter: " + iter3.CurrentItem());

}

/\*\*

\* A class that describes an element of the list

\*/

package com.lab111.labwork5;

/\*\*

\* @author Taras Barabash

\* @version 5.2

\*/

public class Text {

/\*\*

\* A value of the element

\*/

private String t;

/\*\*

\* Next element in a list

\*/

private Text next;

/\*\*

\* Previous element in a list

\*/

private Text prev;

/\*\*

\* Returns a value of the element

\*/

public String getT(){

return t;

}

/\*\*

\* Returns the next element in a list

\*/

public Text getNext(){

return next;

}

/\*\*

\* Returns the previous element in a list

\*/

public Text getPrev(){

return prev;

}

/\*\*

\* Sets the value of an element

\*/

public void setT(String s){

this.t = s;

}

/\*\*

\* Sets the next element

\*/

public void setNext(Text text){

this.next = text;

}

/\*\*

\* Sets the previous element

\*/

public void setPrev(Text text){

this.prev = text;

}

}

/\*\*

\* Class that describes a concrete iterator with a filter by the first letter and implements the Iterator interface

\*/

package com.lab111.labwork5;

/\*\*

\* @author Taras Barabash

\* @version 5.2

\*/

public class ConcreteIterator3 implements Iterator {

/\*\*

\* A link to an aggregate

\*/

public ConcreteAggregate cA;

/\*\*

\* A link to a current element

\*/

public Text current;

/\*\*

\* Length of a string by which a list is filtered

\*/

public char letter;

/\*\*

\* Sets the length

\*/

public void setLetter(char l){

this.letter = l;

}

/\*\*

\* A constructor

\* @param agg Aggregate

\*/

public ConcreteIterator3(ConcreteAggregate agg){

cA = agg;

current = cA.firstEl;

}

@Override

public void First() {

current = cA.firstEl;

if (current.getT().charAt(0) != letter && current.getNext() != null) {

do

current = current.getNext();

while (current.getT().charAt(0) != letter && current.getNext() != null);

}

}

@Override

public void Next() {

do

current = current.getNext();

while (current.getT().charAt(0) != letter && current.getNext() != null);

}

@Override

public boolean isDone() {

return current.getNext() == null;

}

@Override

public String CurrentItem() {

return current.getT();

}

}

/\*\*

\* Class that describes a concrete iterator with a filter by length and implements the Iterator interface

\*/

package com.lab111.labwork5;

/\*\*

\* @author Taras Barabash

\* @version 5.2

\*/

public class ConcreteIterator2 implements Iterator {

/\*\*

\* A link to an aggregate

\*/

public ConcreteAggregate cA;

/\*\*

\* A link to a current element

\*/

public Text current;

/\*\*

\* Length of a string by which a list is filtered

\*/

public int length;

/\*\*

\* Sets the length

\*/

public void setLength(int l){

this.length = l;

}

/\*\*

\* A constructor

\* @param agg Aggregate

\*/

public ConcreteIterator2(ConcreteAggregate agg){

cA = agg;

current = cA.firstEl;

}

@Override

public void First() {

current = cA.firstEl;

if (current.getT().length() != length && current.getNext() != null) {

do

current = current.getNext();

while (current.getT().length() != length && current.getNext() != null);

}

}

@Override

public void Next() {

do

current = current.getNext();

while (current.getT().length() != length && current.getNext() != null);

}

@Override

public boolean isDone() {

return current.getNext() == null;

}

@Override

public String CurrentItem() {

return current.getT();

}

}

/\*\*

\* Class that describes a concrete iterator and implements the Iterator interface

\*/

package com.lab111.labwork5;

/\*\*

\* @author Taras Barabash

\* @version 5.2

\*/

public class ConcreteIterator implements Iterator {

/\*\*

\* A link to an aggregate

\*/

public ConcreteAggregate cA;

/\*\*

\* A link to a current element

\*/

public Text current;

/\*\*

\* A constructor

\* @param agg Aggregate

\*/

public ConcreteIterator(ConcreteAggregate agg){

cA = agg;

current = cA.firstEl;

}

@Override

public void First() {

current = cA.firstEl;

}

@Override

public void Next() {

if(current.getNext() != null)

current = current.getNext();

}

@Override

public boolean isDone() {

return current.getNext() == null;

}

public String CurrentItem() {

return current.getT();

}

}

/\*\*

\* Class that describes a concrete aggregate and implements the Aggregate inteface

\*/

package com.lab111.labwork5;

/\*\*

\* @author Taras Barabash

\* @version 5.2

\*/

public class ConcreteAggregate implements Aggregate {

/\*\*

\* A link to the first element

\*/

public Text firstEl;

/\*\*

\* A link to the current element

\*/

public Text currentEl;

/\*\*

\* Adds a new element to a list

\* @param t value of a new element

\*/

public void add(String t){

if(firstEl == null){

firstEl = new Text();

firstEl.setT(t);

}

else{

currentEl = firstEl;

while(currentEl.getNext() != null)

currentEl = currentEl.getNext();

Text n = new Text();

currentEl.setNext(n);

n.setPrev(currentEl);

n.setT(t);

}

}

/\*\*

\* Creates an iterator

\*/

@Override

public Iterator CreateIterator() {

return new ConcreteIterator(this);

}

/\*\*

\* Creates an iterator with a filter by length

\*/

@Override

public Iterator CreateIterator2() {

return new ConcreteIterator2(this);

}

/\*\*

\* Creates an iterator with a filter by the first letter

\*/

@Override

public Iterator CreateIterator3() {

return new ConcreteIterator3(this);

}

}

/\*\*

\* Interface for creating an iterator object

\*/

package com.lab111.labwork5;

/\*\*

\* @author Taras Barabash

\* @version 5.2

\*/

public interface Aggregate {

/\*\*

\* Creates an iterator

\*/

public Iterator CreateIterator();

/\*\*

\* Creates an iterator with a filter by length

\*/

public Iterator CreateIterator2();

/\*\*

\* Creates an iterator with a filter by the first letter

\*/

public Iterator CreateIterator3();

}

/\*\*

\* Interface for getting access to the elements and looking over them

\*/

package com.lab111.labwork5;

/\*\*

\* @author Taras Barabash

\* @version 5.2

\*/

public interface Iterator {

/\*\*

\* Sets an iterator on the first element

\*/

public void First();

/\*\*

\* Sets an iterator on the next element

\*/

public void Next();

/\*\*

\* Checks if the end of the list was reached

\*/

public boolean isDone();

/\*\*

\* Returns the value of the current element

\*/

public String CurrentItem();

}

}

**ІV. Висновок**

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

One of the elements of a list: fifth

One of the elements of a list filtered by string length: seventh

One of the elements of a list filtered by the first letter: eighth