Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут”
Кафедра АСОІУ

ЗВІТ

про виконання лабораторної роботи
з дисципліни
“Об’єктно-орієнтоване програмування Java”
Тема: Вивчення шаблонів поведінки. Отримання базових навичок з застосування шаблонів Iterator, Mediator та Observer.

Прийняв:
Подрубайло О. О.

Виконав:
студент 3-го курсу
гр. ІП-52 ФІОТ
Набоков Е.М

Київ 2017

1 ПОСТАНОВКА ЗАДАЧІ

1. Вивчити шаблони поведінки для проектування ПЗ. Знати загальну характеристику шаблонів поведінки та призначення кожного з них.
2. Детально вивчити шаблони поведінки для проектування ПЗ – Iterator, Mediator та Observer. 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.labwork5. 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 1. Визначити специфікації класів, які інкапсулюють лінійний список цілих чисел та реалізують можливість звичайного послідовного обходу та послідовного обходу в упорядкованій структурі.

2 UML ДІАГРАММА

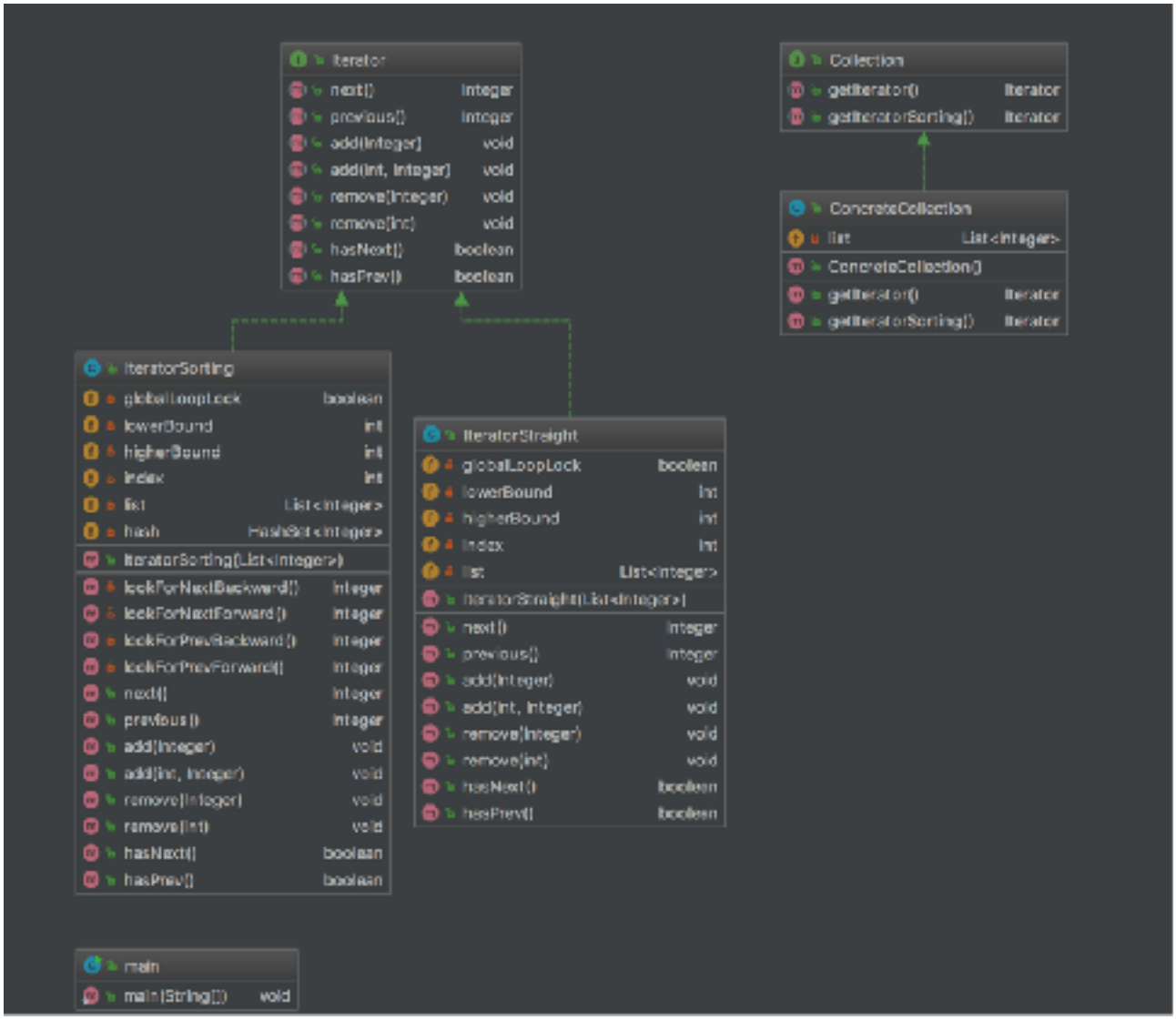

Рисунок 2.1 — схема (поля та конструктори)

3 ПРОГРАМНИЙ КОД

ConcreteCollection.java

```
package com.solutions.labwork5;

/**
 * Class ConcreteCollection.
 * It implements IterableCollection.
 * And has a quick sort implementation
 * for sorting Iterator
 *
 * @author Eduard Nabokov
 * @version 0.1
 * @since 15.11.17
 */
public class ConcreteCollection implements IterableCollection {
    private Integer[] list = {10, 3, 1, 2, 0, 7, 4, 10, 8, 2, 1, 0, 15, 9};

    @Override
    public Iterator getIterator(boolean sortingIterator){
        if (sortingIterator){
            this.quickSort(list, 0, list.length - 1);
            System.out.println("Array was sorted. And you've got a sorting iterator");
        }
        else{
            System.out.println("Array was not sorted. And you've got simple iterator");
        }

        return new ConcreteIterator();
    }

    private void quickSort(Integer[] l, int low, int high) {
        if (l == null || l.length == 0)
            return;

        if (low >= high)
            return;
```

```
    int middle = low + (high - low) / 2;
    int pivot = l[middle];

    int i = low, j = high;
    while (i <= j) {
        while (l[i] < pivot) {
            i++;
        }

        while (l[j] > pivot) {
            j--;
        }

        if (i <= j) {
            int temp = l[i];
            l[i] = l[j];
            l[j] = temp;
            i++;
            j--;
        }
    }

    if (low < j)
        quickSort(l, low, j);

    if (high > i)
        quickSort(l, i, high);
}

private class ConcreteIterator implements Iterator {

    int index = 0;

    @Override
    public Object next(){
        if (index >= list.length) {
            throw new IndexOutOfBoundsException("Sorry, it's the highest bound of a list.");
        }

        return list[index++];
    }
```

```
        @Override
        public Object previous() throws IndexOutOfBoundsException {
            if (index < 0) {
                throw new IndexOutOfBoundsException("Sorry, it's the lowest bound of a list.");
            }

            return list[index--];
        }

        @Override
        public boolean hasMore() {
            if (index < list.length && index >= 0) {
                return true;
            }

            if (index == list.length){
                index -= 1;
            }

            if (index == -1) {
                index += 1;
            }

            return false;
        }
    }
}
```

IterableCollection.java

```
package com.solutions.labwork5;

/**
 * Interface IterableCollection.
 * It just create an interface
 * for future class,
 * that will return iterator
 *
```

```
 * @author Eduard Nabokov
 * @version 0.1
 * @since 15.11.17
 */
interface IterableCollection {
    Iterator getIterator(boolean sortingIterator);
}
```

Iterator.java

```
package com.solutions.labwork5;

/**
 * interface Iterator.
 * It define methods like next, previous and hasMore
 *
 * @author Eduard Nabokov
 * @version 0.1
 * @since 15.11.17
 */

interface Iterator {
    Object next();
    Object previous();
    boolean hasMore();
}Main.java

package com.solutions.labwork3;
/**
 * Class Main - starting point for project.
 * Represent usage of implemented
 * classes and methods.
 *
 * @author Eduard Nabokov
 * @version 0.1
 * @since 23.10.17
 */

class main {
    public static void main(String[] args) {
        Component filesystem = new Folder();

        Component folder1 = new Folder();
```

```
        filesystem.addComponent(folder1);

        folder1.addComponent(new File(10));
        folder1.addComponent(new File(15));

        Component folder2 = new Folder();
        Component file2 = new File(15);

        file2.addComponent(new File(17));
        folder2.addComponent(file2);
        filesystem.addComponent(folder2);

        filesystem.dropComponent(folder1);

        System.out.println(filesystem.getSize());
    }
}
```

Main.java

```
package com.solutions.labwork5;

/**
 * Main class
 *
 * @author Eduard Nabokov
 * @version 0.1
 * @since 15.11.17
 */
public class main {
    public static void main(String[] args) {
        ConcreteCollection c = new ConcreteCollection();

        Iterator iter = c.getIterator(false);

        System.out.println("=====FORWARD PASS=====");
        while (iter.hasMore()) {
            System.out.println(iter.next());
        }

        System.out.println("\n=====BACKWARD PASS=====");
        while (iter.hasMore()) {
            System.out.println((iter.previous()));
```

```
        }
    }
}
```