

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ “ЛЬВІВСЬКА
ПОЛІТЕХНІКА”

Кафедра систем штучного інтелекту



Лабораторна робота N2

З дисципліни

«Об'єктно-орієнтоване програмування»

Виконав:

Студент групи КН-108

Пулик Максим

Викладач:

Грабовська Н.Р.

Мета:

- Набуття навичок розробки власних контейнерів.
- Використання ітераторів.
- Тривале зберігання та відновлення стану об'єктів.
- Ознайомлення з принципами серіалізації/десеріалізації об'єктів.
- Використання бібліотек класів користувача.

1. Вимоги

1. Розробити клас-контейнер, що ітерується для збереження початкових даних Вашого варіанту завдання з попередньої роботи (Прикладні задачі. Список з 1-15 варіантів) у вигляді масиву рядків з можливістю додавання, видалення і зміни елементів.

2. В контейнері реалізувати та продемонструвати наступні методи:

- `String toString()` повертає вміст контейнера у вигляді рядка;
- `void add(String string)` додає вказаний елемент до кінця контейнеру;
- `void clear()` видаляє всі елементи з контейнеру;
- `boolean remove(String string)` видаляє перший випадок вказаного елемента з контейнера;
- `Object[] toArray()` повертає масив, що містить всі елементи у контейнері;
- `int size()` повертає кількість елементів у контейнері;
- `boolean contains(String string)` повертає `true` , якщо контейнер містить вказаний елемент;
- `boolean containsAll(Container container)` повертає `true` , якщо контейнер містить всі елементи з зазначеного у параметрах;
- `public Iterator iterator()` повертає ітератор відповідно до `Interface Iterable` .

3. В класі ітератора відповідно до `Interface Iterator` реалізувати методи:

- `public boolean hasNext()` ;
- `public String next()` ;
- `public void remove()` .

4. Продемонструвати роботу ітератора за допомогою циклів `while` и `for each` .

5. Забороняється використання контейнерів (колекцій) і алгоритмів з `Java Collections Framework` .

6. Реалізувати і продемонструвати тривале зберігання/відновлення розробленого контейнера за допомогою серіалізації/десеріалізації .

7. Обмінятися відкомпільованим (без початкового коду) службовим класом (Utility Class) рішення одного варіанту задачі (Прикладні задачі. Список з 1-15 варіантів) з сусіднім номером. 1 міняється з 2, 2 з 3, 3 з 4, 4 з 5 і т.д. Останній, 15 міняється з 1 варіантом і далі аналогічно. 8. Прогонювати послідовну та вибірккову обробку елементів розробленого контейнера за допомогою власного і отриманого за обміном службового класу. 9. Реалізувати та продемонструвати порівняння, сортування та пошук елементів у контейнері. 10.Розробити консольну програму та забезпечити діалоговий режим роботи з користувачем для демонстрації та тестування рішення.

1.1 Розробник

Пулик Максим, КН-108, номер варіанту індивідуального завдання – 7.

1.2 Задача

Опис програми:

Клас-контейнер , що реалізує згадані вище методи , та клас-ітератор.

2.1 Засоби ООП

Реалізовано три класи та зв'язок їх між собою.

2.2 Ієрархія та структура класів

Main – меню роботи з програмою.

MyContainer – клас-контейнер.

MyIterator – клас-ітератор.

Lab2 [C:\Users\ACER\IdeaProjects\Lab2] - \src\com\company\Main.java [Lab2] - IntelliJ IDEA

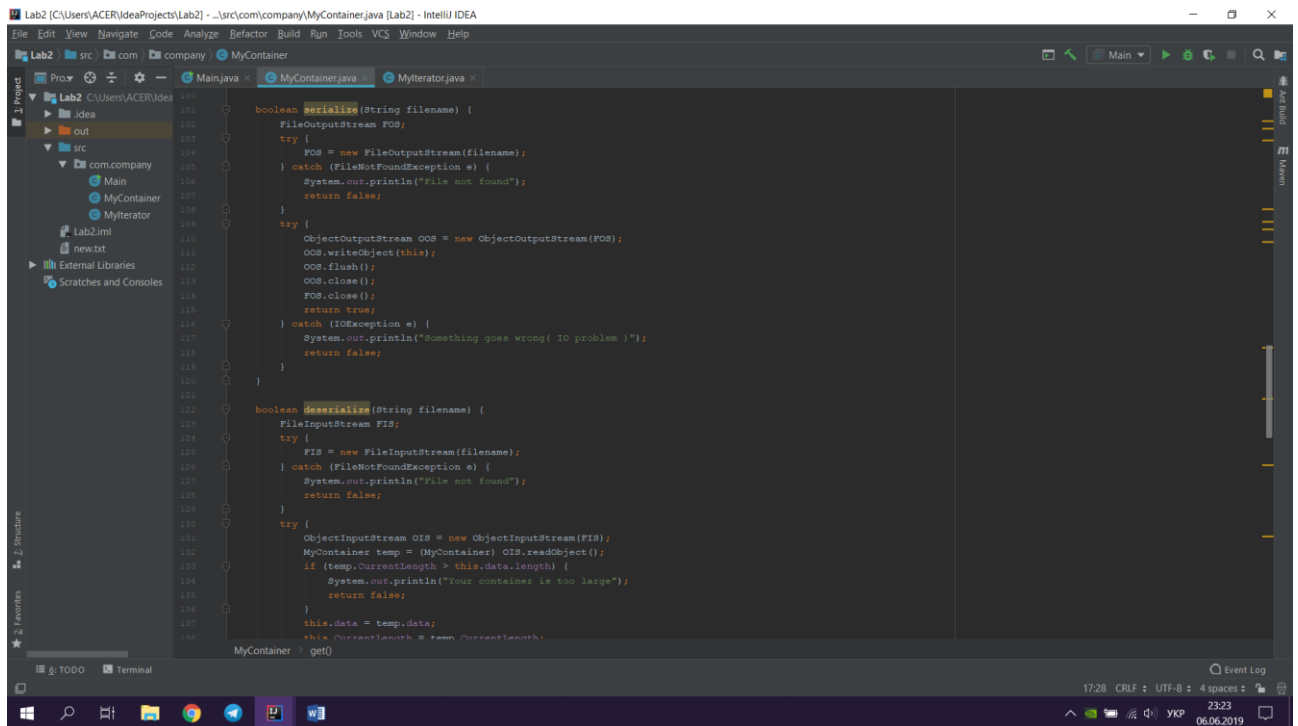
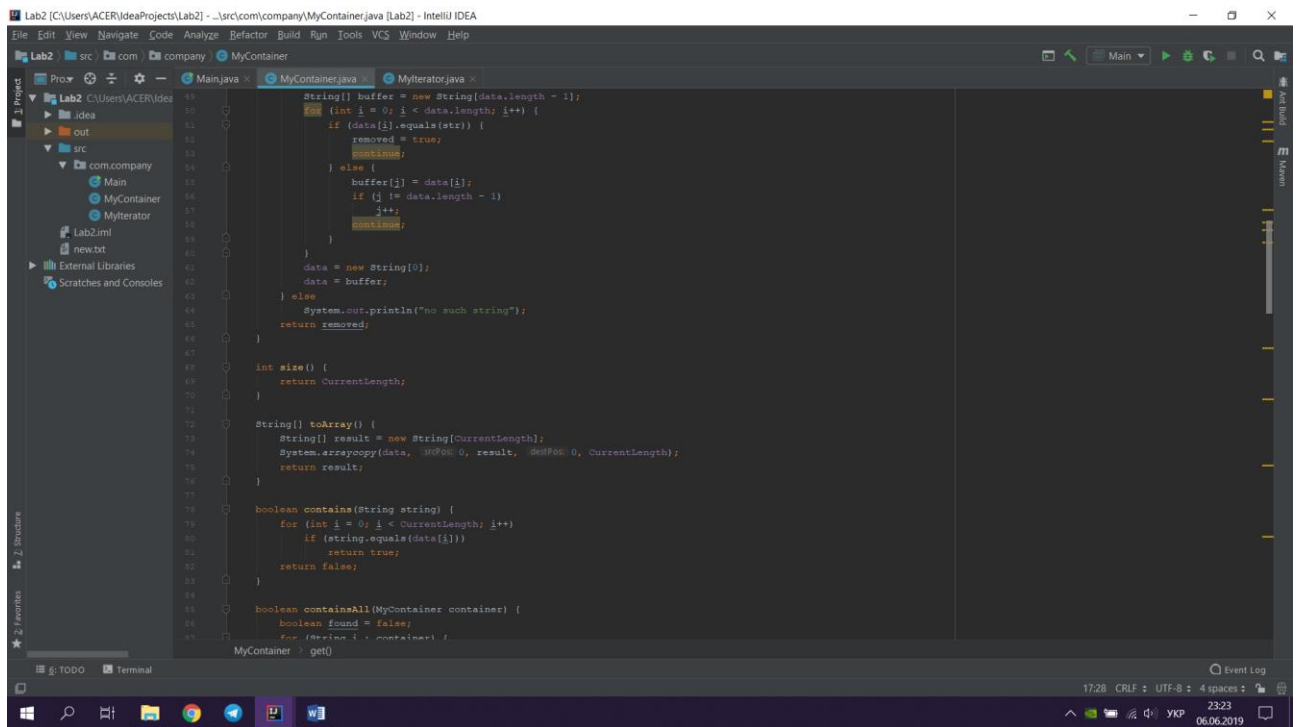
```
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Lab2 src com company Main
Main.java MyContainer.java MyIterator.java
23 System.out.println("Enter command: ");
24 System.out.println("1 - Make fill container");
25 System.out.println("2 - Clear container");
26 System.out.println("3 - Show container");
27 System.out.println("4 - Container menu");
28 System.out.println("5 - Exit");
29 Command = in.nextByte();
30 switch (command) {
31     case 1:
32         System.out.println("Enter string:");
33         in.nextLine();
34         Container.add(in.nextLine());
35         break;
36     case 2:
37         Container.clear();
38         System.out.println("Well done!");
39         break;
40     case 3:
41         for (String s : Container.data)
42             System.out.println(s);
43         break;
44     case 4:
45         System.out.println("Next command, sir: ");
46         System.out.println("1 - Add element");
47         System.out.println("2 - Remove element");
48         System.out.println("3 - Convert to array and iterate through");
49         System.out.println("4 - Current size");
50         System.out.println("5 - Max size");
51         System.out.println("6 - Check string");
52         System.out.println("7 - Check subcontainer");
53         System.out.println("8 - Write to file (serialize)");
54         System.out.println("9 - Read from file (deserialize)");
55         System.out.println("10 - Get element by index");
56         System.out.println("11 - Get element's index");
57         System.out.println("12 - Sort");
58         System.out.println("13 - Iterate through container (foreach)");
59         System.out.println("14 - Iterate through container (while)");
60         System.out.println("15 - Return");
61         Command = in.nextByte();
62 }
63 Main main()
```

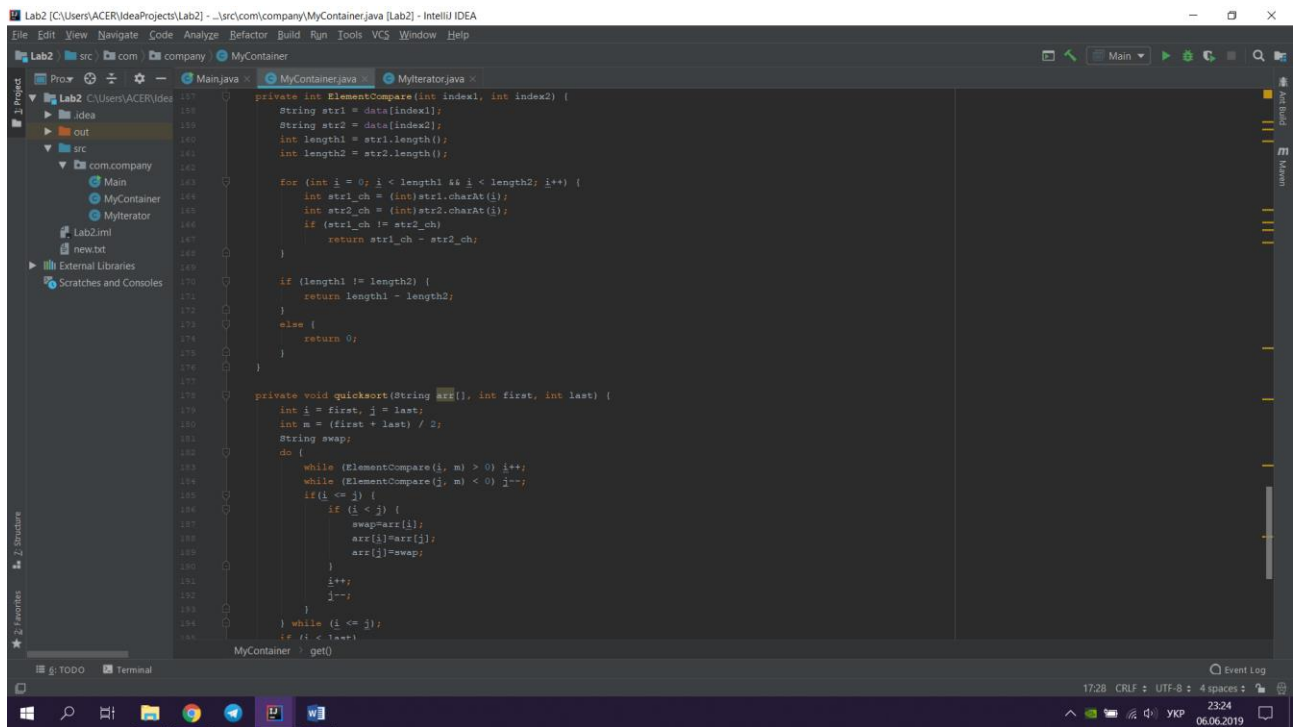
22:57 CRLF UTF-8 4 spaces 23:24 06.06.2019

Lab2 [C:\Users\ACER\IdeaProjects\Lab2] - \src\com\company\MyContainer.java [Lab2] - IntelliJ IDEA

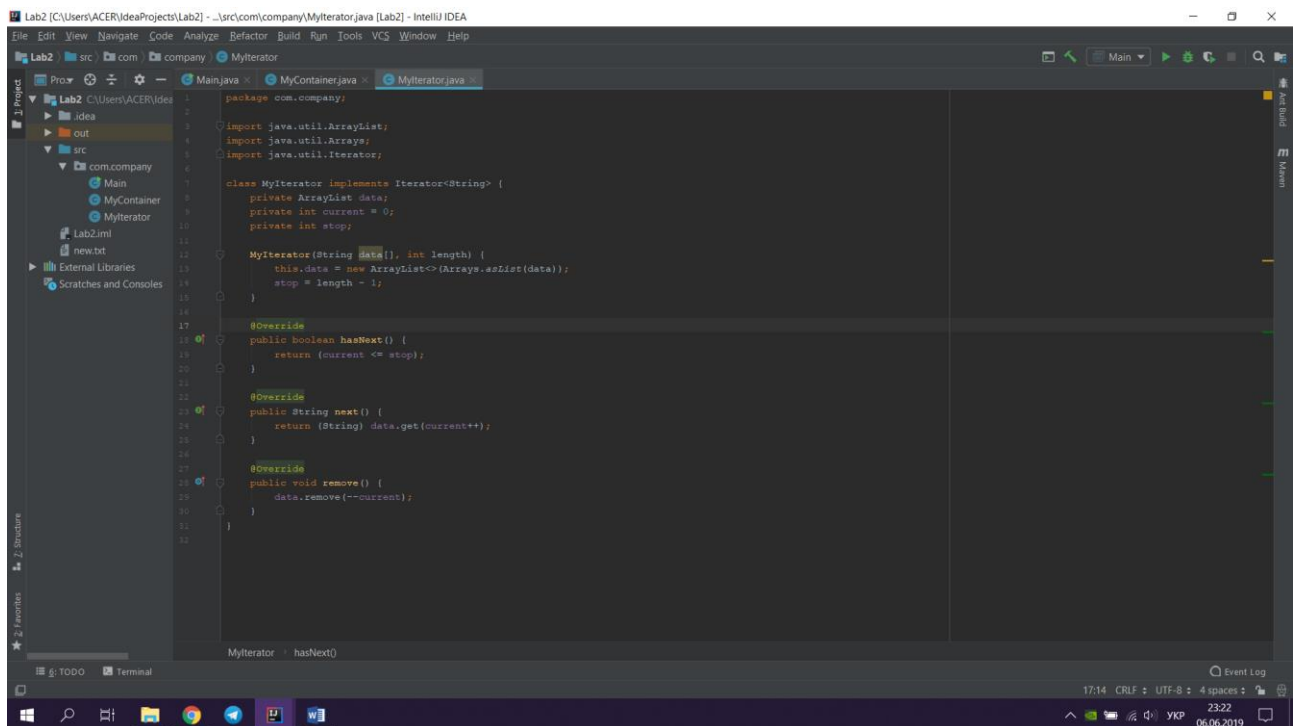
```
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Lab2 src com company MyContainer
Main.java MyContainer.java MyIterator.java
20 private int CurrentLength = 0;
21
22 public void add(String s) {
23     data = Arrays.copyOf(data, newLength() + 1);
24     data[data.length - 1] = s;
25 }
26
27 String get(int index) {
28     if (index >= CurrentLength)
29         return null;
30     else
31         return data[index];
32 }
33
34 public String toString() {
35     if (CurrentLength == 0)
36         return null;
37     StringBuilder result = new StringBuilder();
38     result.append("[ ");
39     for (int i = 0; i < CurrentLength - 1; i++) {
40         result.append(data[i]);
41         result.append(", ");
42     }
43     result.append(data[CurrentLength - 1]);
44     result.append(" ]");
45     return new String(result);
46 }
47
48 void clear() {
49     for (int i = 0; i < CurrentLength; i++) {
50         data[i] = null;
51     }
52     CurrentLength = 0;
53 }
54
55 boolean remove(String str) {
56     boolean removed = false;
57     int j = 0;
58     for (int i = 0; i < data.length; i++) {
59         if (data[i].equals(str)) {
60             removed = true;
61             data[i] = null;
62         }
63     }
64     return removed;
65 }
66
67 MyContainer > get()
```

17:28 CRLF UTF-8 4 spaces 23:23 06.06.2019





```
187 private int ElementCompare(int index1, int index2) {
188     String str1 = data[index1];
189     String str2 = data[index2];
190     int length1 = str1.length();
191     int length2 = str2.length();
192
193     for (int i = 0; i < length1 && i < length2; i++) {
194         int str1_ch = (int)str1.charAt(i);
195         int str2_ch = (int)str2.charAt(i);
196         if (str1_ch != str2_ch)
197             return str1_ch - str2_ch;
198     }
199
200     if (length1 != length2) {
201         return length1 - length2;
202     }
203     else {
204         return 0;
205     }
206 }
207
208 private void quicksort(String arr[], int first, int last) {
209     int i = first, j = last;
210     int m = (first + last) / 2;
211     String swap;
212     do {
213         while (ElementCompare(i, m) > 0) i++;
214         while (ElementCompare(j, m) < 0) j--;
215         if (i <= j) {
216             if (i < j) {
217                 swap=arr[i];
218                 arr[i]=arr[j];
219                 arr[j]=swap;
220             }
221             i++;
222             j--;
223         }
224     } while (i <= j);
225     if (i < j) {
226         quicksort(arr, i, j);
227     }
228 }
```



```
1 package com.company;
2
3 import java.util.ArrayList;
4 import java.util.Iterator;
5 import java.util.List;
6
7 class MyIterator implements Iterator<String> {
8     private ArrayList data;
9     private int current = 0;
10    private int stop;
11
12    MyIterator(String data[], int length) {
13        this.data = new ArrayList<>(Arrays.asList(data));
14        stop = length - 1;
15    }
16
17    @Override
18    public boolean hasNext() {
19        return (current <= stop);
20    }
21
22    @Override
23    public String next() {
24        return (String) data.get(current++);
25    }
26
27    @Override
28    public void remove() {
29        data.remove(--current);
30    }
31
32 }
```

3. Варіанти використання

Програма може використовуватись для зберігання стрічок та проведення махінацій з ними за допомогою визначених методів.

ВИСНОВКИ

У ході роботи розвинулись навички написання власних контейнерів , власних ітераторів, навички роботи із тривалим зберіганням даних(серіалізація та десеріалізація).