

## Лабораторна робота №2.

### 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.

### 1.1 Розробник

Гірник Юрій, КН-108, номер варіанту індивідуального завдання – 6.

### 1.2 Загальна завдання

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<String> 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 і 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. Продемонструвати 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.

### **1.3 Задача**

Ввести текст. З тексту видалити всі символи, крім пропусків, які не є буквами. Пропуски, що повторюються, замінити на одиночні. Між послідовностями літер, де знаходяться розділові знаки, залишити хоча б один пропуск ("a,b,c" -> "a, b, c"). Вивести початковий текст та результат.

## **2 Опис програми**

Дана розроблена програма дозволяє обробляти(відповідно до індивідуального завдання) введений текст, реалізована робота двох можливих режимів роботи: -h – режим при якому на початку програма з'являється певна інформація(інформація про розробника, суть індивідуального завдання, можливі функції консольного меню), -d або ж -debug – спеціальний режим роботи, при якому користувачу виводить на екран проміжковий результат редакування рядка.

Також дана програма зберігає інформацію в створений контейнер з низкою різноманітних функцій(сортування, видалення, серіалізація та десеріалізація, додавання тощо).

## 2.1 Засоби ООП

У лабораторній роботі використано різні класи з багатьма методами. Використано інкапсуляцію даних.

## 2.2 Ієрархія та структура класів

- А) Клас-контейнер MyIterator;
- Б) Клас-ітератор MyIterator;
- В) Клас з автоматичними тестами Tester;
- С) Клас Main з консольною обробкою даних.

## 2.3 Важливі фрагменти програми.

```
public class MyContainer implements Serializable{
    private String[] data;
    private int length = 0;

    MyContainer(){
        data = new String[0];
    }
    @Override
    public String toString(){
        if(data==null){
            return "Container is empty";
        }
        StringBuilder result = new StringBuilder();
        String [] temp = new String[length];
        System.arraycopy(data, 0, temp, 0, length);
        result.append("[");
        for(String a : temp){
            result.append(a);
            if(a != temp[length-1])
                result.append(", ");
        }
        result.append("]");
        return result.toString();
    }

    void add(String string){
        String[] temp = data;
        data = new String[++length];
        System.arraycopy(temp, 0, data, 0, length-1);
        data[length-1] = string;
    }
}
```

```

public boolean serialization() {
    FileOutputStream outputStream;
    try {
        outputStream = new FileOutputStream("D:\\NetBeans 8.0.2\\data.txt");
    } catch (FileNotFoundException ex) {
        System.out.println("Something wrong....");
        return false;
    }
    try {
        try (ObjectOutputStream objectOutputStream = new ObjectOutputStream(outputStream)) {
            objectOutputStream.writeObject(data);
            System.out.println("Success serialization");
        }
        outputStream.close();
        return true;
    } catch (IOException ex) {
        System.out.println("Sorry, but....");
        return false;
    }
}

public boolean deserialization() {
    FileInputStream fileInputStream;
    try {
        fileInputStream = new FileInputStream("D:\\NetBeans 8.0.2\\data.txt");
    } catch (FileNotFoundException ex) {
        System.out.println("Something wrong....");
        return false;
    }
    try {
        ObjectInputStream objectInputStream = new ObjectInputStream(fileInputStream);
        data = (String[]) objectInputStream.readObject();
        length = data.length;
        System.out.println("Success deserialization");
        return true;
    } catch (IOException ex) {
        System.out.println("Sorry, but....");
        return false;
    } catch (ClassNotFoundException ex) {
        System.out.println("Wrong input data....");
        return false;
    }
}
}

```

### 3. Варіанти використання

Дана програма може використовуватись для тривалого зберігання тексту, додавання нового, редагування тощо.

### ВИСНОВКИ

У ході роботи розвинулись навички створення власних класів-контейнерів, створення та використання ітераторів, реалізації тривалого зберігання інформації та відновлення інформації (методи `serialization` та `deserialization`); покращилась обробка консольних команд з лаб.1.