НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Кафедра «ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ ТА ПРОГРАМУВАННЯ»

«Об’єктно-орієнтоване програмування ч.2»

Звіт з лабораторної роботи №9
Тема: «Параметризація в Java»

Виконав:
ст. гр. КИТ-118в
Кліщов Б. Р.

Перевірив:
Пугачев Р.В.

Харків – 2020

Мета: Вивчення принципів параметризації в Java. Розробка параметризованих класів та методів.

1 ВИМОГИ

1.1 Розробник

Кліщов Б. Р., КИТ-118в

1.2 Загальне завдання

1.3 Задача

Створити власний клас-контейнер, що параметризується (Generic Type), на основі зв'язних списків для реалізації колекції domain-об'єктів лабораторної роботи №7.

Для розроблених класів-контейнерів забезпечити можливість використання їх об'єктів у циклі foreach в якості джерела даних.

Забезпечити можливість збереження та відновлення колекції об'єктів: 1) за допомогою стандартної серіалізації; 2) не використовуючи протокол серіалізації.

Продемонструвати розроблену функціональність: створення контейнера, додавання елементів, видалення елементів, очищення контейнера, перетворення у масив, перетворення у рядок, перевірку на наявність елементів.

Забороняється використання контейнерів (колекцій) з Java Collections Framework.

2 ОПИС ПРОГРАМИ

2.1 Засоби ООП

Був розроблений domain object, ітеруючий клас контейнер. Також були розроблені дві серіалізації: із використовуванням стандартного протоколу та без нього.

2.2 Ієрархія та структура класів

Див. рис.1.

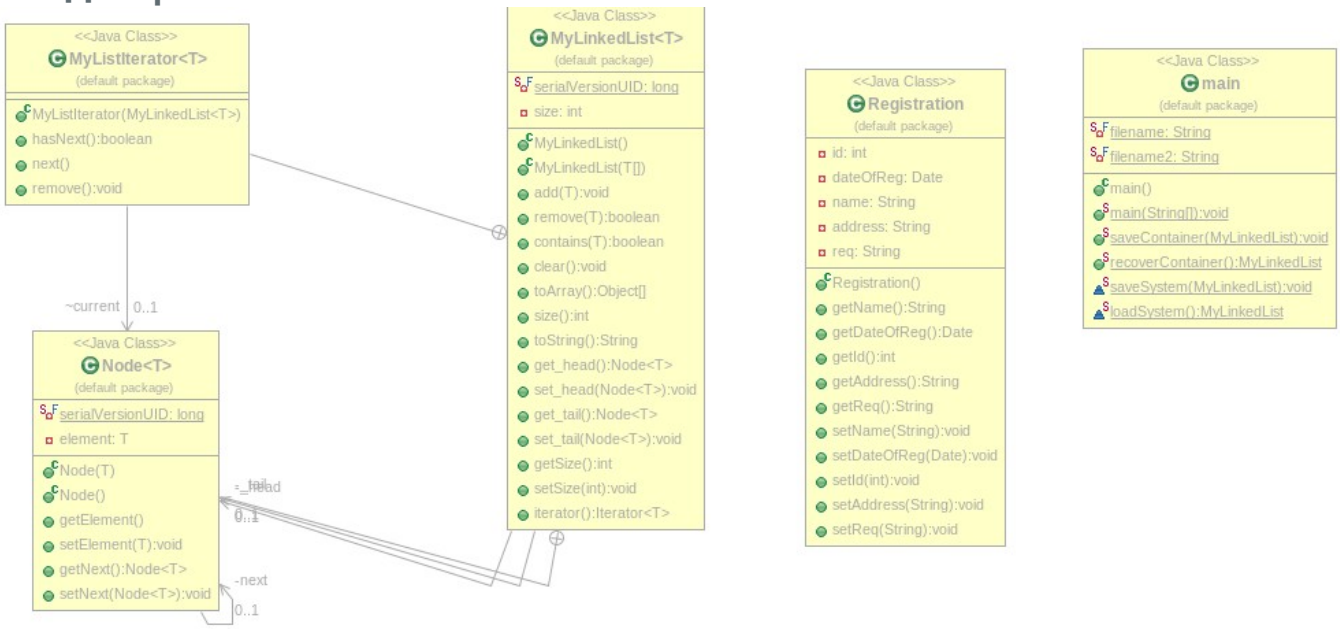

Рисунок 1. Діаграма класів

2.3 Важливі фрагменти програми

```java
public class Client{
    private int id;
    private Date dateOfReg;
    private String name;
    private String address;
    private String req;
```

Рисунок 2. Domain object

```java
public Iterator<T> iterator() {
    return new MyListIterator<T>(this);
}
class MyListIterator<T> implements Iterator<T> {
    Node<T> current;
    public MyListIterator(MyLinkedList<T> list) {
        current = (Node<T>) _head;
    }
    public boolean hasNext() {
        return current != null;
    }
    public T next() {
        T data = current.getElement();
        current = current.getNext();
        return data;
    }
    public void remove() {
        throw new UnsupportedOperationException();
    }
}
```

Рисунок 3. Ітератор у класі-контейнері

```java
public class MyLinkedList<T> implements Serializable, Iterable<T> {
    private static final long serialVersionUID = 1L;
    private Node<T> _head;
    private Node<T> _tail;
    private int size;

    public static class Node<T> implements Serializable {
        private static final long serialVersionUID = 1L;
        private T element;
        private Node<T> next;
```

Рисунок 4. Клас-контейнер та внутрішній клас Node

3 ВАРІАНТИ ВИКОРИСТАННЯ

```
System.out.println("Init: " + list.toString());
saveSystem(list);
list.clear();
System.out.println("After clear: " + list.toString());
list = loadSystem();
System.out.println("After loadSystem(): " + list.toString());
list.add("addStr");
System.out.println("After add 'addStr': " + list.toString());
System.out.println("List is contain 'addStr': " + list.remove("addStr"));
System.out.println("After remove 'addStr': " + list.toString() + "\n");

System.out.println("Saving list to container...");
saveContainer(list);
System.out.println("Complete!!!");
System.out.println("Clearing list...");
list.clear();
System.out.println("Complete!!!");
System.out.println("After clear: " + list.toString());
list = recoverContainer();
System.out.println("After recoverContainer(): " + list.toString());
list.add("addStr2");
System.out.println("After add 'addStr2': " + list.toString());
System.out.println("List is contain 'str2': " + list.remove("str2"));
System.out.println("After removing 'str2': " + list.toString() + "\n");

System.out.println("Using for-each: ");
for(Object s : list) {
    System.out.println(s.toString());
```

Рисунок 5. Тестування

```
Init: [ "str1" "str2" "str3" ]
After clear: [ ]
After loadSystem(): [ "str1" "str2" "str3" ]
After add 'addStr': [ "str1" "str2" "str3" "addStr" ]
List is contain 'addStr': true
After remove 'addStr': [ "str1" "str2" "str3" ]

Saving list to container...
Complete!!!
Clearing list...
Complete!!!
After clear: [ ]
After recoverContainer(): [ "str1" "str2" "str3" ]
After add 'addStr2': [ "str1" "str2" "str3" "addStr2" ]
List is contain 'str2': true
After removing 'str2': [ "str1" "str3" "addStr2" ]

Using for-each:
str1
str3
addStr2
```

Рисунок 6. Результат виконання

ВИСНОВКИ

Були вивчені принципи параметризації в Java, розроблені параметризовані класи-контейнери на основі зв’язних списків та їх методи.