МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Кафедра систем штучного інтелекту

Звіт

Лабораторна робота №4

З дисципліни:

Об’єктно орієнтоване програмування

Виконав

студент групи КН-111

Жигайло Ярослав

Викладач:

Грабовська Н. Р.

Тема роботи

Розробка власних контейнерів. Ітератори. Серіалізація/десеріалізація об’єктів. Бібліотека класів користувача

1.Вимоги

1.1 Розробник

Жигайло Ярослав Олегович

КН-111

7 варіант

1.2 Загальне завдання

Створити власний клас-контейнер, що параметризується (Generic Type), на основі зв’язних списків для реалізації колекції domain-об’єктів з лабораторної роботи №10 (Прикладні задачі. Список №2. 20 варіантів)

1.3 Завдання

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.

Розробити параметризовані методи (Generic Methods) для обробки колекцій об’єктів згідно (Прикладні задачі. Список №2. 20 варіантів)

2.Опис програми

2.1 Засоби ООП

Декомпозиція для розділення завдання між методами і класами, інкапсуляція , розробка Generic Types , створення класів-контейнерів

2.2 Ієрархія і структура класів

Для завдання написано клас Domain об’єктів DomainClient , linked List MyLinkedList , node Node , menu Menu , path getter PathGetter

2.3 Важливі частини коду

Зв’язаний список

```
public class LinkedList<T> implements Iterable<T>{
        Node<T> head;
        private int currentIndex = 0;
          public LinkedList()
          {
                head = new Node<T>();
                head.setData(null);
                head.setNext(null);
          }
          public Node<T> getHead(){return head;}
          public void setHead(Node<T> head) {this.head = head;}

          public void add(T data)
          {
          Node<T> temp = new Node<T>();
          temp.setData(data);
          temp.setNext(null);
```

```
        Node<T> current = head;
                while(current.getNext() !=null)
                {
                        current = current.getNext();
                }
                current.setNext(temp);
        }
        public T get(int index){
    Node<T> temp = head;
    int counter = 0;
    while (temp != null) {
      if (counter == index+1) {
        return (T) temp.getData();
      }
      counter++;
      temp = temp.getNext();
    }
    return null;
}


        public boolean delete(int index)
        {
                if(index<0)
                        return false;
                Node<T> current = head;
                Node<T> current2;
                for(int i = 0 ; i < index ; i++)
```

```
{
        if(current.getNext() != null)
                current = current.getNext();
        else if(current.getNext() == null)
                return false;
}
current2 = current;
if(current.getNext() != null)
{
        current = current.getNext();
        if(current.getNext() == null)
        {
                current2.setNext(null);
                                return true;
        }
        else
        {
                current = current.getNext();
                current2.setNext(current);
                return true;
        }
}
else if(current.getNext() == null)
{
        return false;
}
return false;
```

```
    }

        public int size()
        {
         Node<T> current = head;
         int i = 0;
         while(current.getNext()!=null)
         {
                current = current.getNext();
                i++;
         }
         return i;
        }
         public T getNext() {
                if(get(currentIndex) != null)
   currentIndex++;
   return get(currentIndex);
}
          @Override
            public Iterator<T> iterator() {
               Iterator<T> it = new Iterator<T>() {


                @Override
                public boolean hasNext() {
                   return currentIndex < size()-1;
```

```
        }


        @Override
        public T next() {
         if(get(currentIndex) != null)
         currentIndex++;
            return get(currentIndex);
         }
        @Override
        public void remove() {
           throw new UnsupportedOperationException();
        }
      };
      return it;
    }}
```

Клас Node

```java
public class Node<T> {
    private Node<T> next;
    private T data;

    public Node() {

    }

    public Node<T> getNext() {
        return next;
    }
    public void setNext(Node<T> next) {
        this.next = next;
    }
    public T getData() {
        return data;
    }
    public void setData(T data) {
        this.data = data;
    }
}
```

Висновки

Створив власний клас-контейнер, що параметризується (Generic Type), на основі зв’язних списків для реалізації колекції domain-об’єктів з лабораторної роботи №10 (Прикладні задачі. Список №2. 20 варіантів)