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

Бойко Андрій

Викладач:

Грабовська Н. Р.

Тема роботи

Використання об’єктно-орієнтованого підходу для розробки об’єкта предметної (прикладної) галузі. Оволодіння навичками управління введенням/виведенням даних з використанням класів Java SE.

1. Вимоги

1.1 Розробник

Бойко Андрій Віталійович

КН-111

3 варіант

1.2 Загальне завдання

Використовуючи об’єктно-орієнтований аналіз, реалізувати класи для представлення сутностей відповідно списку прикладних задач - domain-об’єктів

1.3 Завдання

1. 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)
2. Для розроблених класів-контейнерів забезпечити можливість використання їх об’єктів у циклі foreach в якості джерела даних. 3. Забезпечити можливість збереження та відновлення колекції об’єктів: 1) за допомогою стандартної серіалізації; 2) не використовуючи протокол серіалізації. 4. Продемонструвати розроблену функціональність: створення контейнера, додавання елементів, видалення елементів, очищення контейнера, перетворення у масив, перетворення у рядок, перевірку на наявність елементів. 5. Забороняється використання контейнерів (колекцій) з Java Collections Framework . 6. Розробити параметризовані методи (Generic Methods) для обробки колекцій об’єктів згідно (Прикладні задачі. Список №2. 20 варіантів). 7. Продемонструвати розроблену функціональність (створення, управління та обробку власних контейнерів) в діалоговому та автоматичному режимах. a. Автоматичний режим виконання програми задається параметром командного рядка -auto . Наприклад, java ClassName -auto . b. В автоматичному режимі діалог з користувачем відсутній, необхідні данні генеруються, або зчитуються з файлу.

2.Опис програми

Main

```
package lab_3;

import java.beans.XMLDecoder;
import java.beans.XMLEncoder;
import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.util.ArrayList;

import java.util.Scanner;

import lab_4.LinkedList;
class Menu
	{
	int k;
	public int execute(Scanner in)
	{
		System.out.println("Choose what you want to do:");
		System.out.println("1. Create new person");
		System.out.println("2. Change name of the person");
		System.out.println("3. Change date of the person");
		System.out.println("4. Change numbers");
		System.out.println("5. Change adress");
		System.out.println("6. Change regdate");
		System.out.println("7. name");
		System.out.println("8. date");
		System.out.println("9. number");
		System.out.println("10. adress");
		System.out.println("11. regdate");
		System.out.println("12. Print to the file");
		System.out.println("13. Get from the file");
		System.out.println("14. Choose the person");
		System.out.println("15 . Finish the program");
		k = in.nextInt();

		return k;
	}

	}
public class Main {
```

```
public static void main(String[] args) throws FileNotFoundException {

	LinkedList<AdressBook> list = new LinkedList<AdressBook>();
	ArrayList<Integer> temp=new ArrayList<>();
	ArrayList<Integer> temp1=new ArrayList<>();
	int user = 0;
	int N=0;
	Scanner in = new Scanner(System.in);
	Scanner on = new Scanner(System.in);
	Menu main = new Menu();

	while(true)
	{
	int k = main.execute(in);

	switch (k)
	{
	case 1 :
		list.add(new AdressBook());
	break;

	case 2 :
		System.out.println("Enter the name : ");
			list.get(user).setName(on.nextLine());
	break;

	case 3 :
		System.out.println("Enter the date : ");
			list.get(user).setDate(on.nextLine());
	break;

	case 4 :
		System.out.println("Enter phone number and its place : ");
	int t;
		while(true) {
		t=in.nextInt();
		if(t==0)
		{
			break;
		}
		else {
			temp.add(t);
	}}
		list.get(user).setNumbers(temp);


	break;
```

```
case 5 :
	System.out.println("Enter adress : ");
	list.get(user).setAdress(on.nextLine());
break;

case 6 :
	System.out.println("Enter regdate : ");
	list.get(user).setRegdate(on.nextLine());
break;


case 7:
	System.out.println(list.get(user).getname());
break;

case 8 :
	System.out.println((list.get(user).getDate()));
break;

case 9 :
	temp1=list.get(user).getNumbers();
	System.out.println(temp1);

break;

case 10 :
	System.out.println(list.get(user).getAdress());
break;

case 11 :
	System.out.println(list.get(user).getRegdate());
break;
case 12:
	XMLEncoder encoder;
	try {
		encoder = new XMLEncoder(
			new BufferedOutputStream(
			new FileOutputStream(PathGetter.getPath(in).toString())));
		encoder.writeObject(list);
		encoder.close();
	} catch (FileNotFoundException e) {
		e.printStackTrace();
	}
break;
case 13 :
	XMLDecoder decoder = new XMLDecoder(
		  new BufferedInputStream(
		  new FileInputStream(PathGetter.getPath(in).toString())));
```

```
                        list =(LinkedList<AdressBook>) decoder.readObject();
                        decoder.close();

                        break;

                case 14 :
                        System.out.println("Choose the person from 0 to "+(list.size()-1));
                        user = in.nextInt();
                break;
                case 15 :
                        in.close();
                        on.close();
                        return;

                }
                }

}}
```

AdressBook

```
package lab_3;


import java.util.ArrayList;
import java.util.Arrays;
import java.util.Scanner;

public class AdressBook{
        private String name;
        private String date;
        private ArrayList<Integer> numbers;
        private String adress;
        private String regdate;
        Scanner scn=new Scanner(System.in);
        public AdressBook(String name)
        {
                this.name = name;
        }
        public AdressBook()
        {       }
        public void setName(String name)
        {
                this.name = name;
        }

        public void setDate(String date)
```

```
{
	this.date = date;
}
public void setNumbers(ArrayList<Integer> value ) {

	numbers=value;
}

public void setAdress(String adress)
{
	this.adress = adress;
}
public void setRegdate(String regdate)
{
	this.regdate = regdate;
}


public String getname()
 {
	 return name;
 }
 public String getDate()
 {
	 return date;
 }

 public String getAdress()
 {
	return adress;
 }
 public String getRegdate()
 {
	 return regdate;
 }
 public ArrayList<Integer> getNumbers()
 {
	return numbers;
 }

 }
```

PathGetter

```
package lab_3;

import java.io.IOException;
import java.nio.file.*;
import java.util.ArrayList;
```

```
import java.util.Scanner;

public class PathGetter {

   public static Path getPath(Scanner input) {
      Path p = Paths.get(System.getProperty("user.home"));
      while (true) {
         System.out.println(p);
         try (DirectoryStream<Path> stream = Files.newDirectoryStream(p)) {
            int i = 1;
            ArrayList<String> foldersList = new ArrayList<>();
            for (Path file: stream) {
               String check = file.getFileName().toString();
               if (check.contains(".xml") || !check.contains(".")) {
                  System.out.printf("[%d] %s%n", i++, check);
                  foldersList.add(check);
               }
            }
            System.out.printf("~[%d] return%n", i++);
            System.out.printf("~[%d] select XML file or create a new one%n", i);
            System.out.println("chose option or directory: ");
            int option = input.nextInt();
            if (option < 1 || option > i)
               System.err.println("error");
            else if (option < i-1) {
               if (Files.isDirectory(p.resolve(foldersList.get(option-1))))
                  p = p.resolve(foldersList.get(option-1));
               else System.err.println("it isn't a directory");
            } else if (option == i-1)
               p = p.getRoot().resolve(p.getParent());
            else {
               System.out.println("enter filename in format name.xml:");
               input.nextLine();
               String filename = input.nextLine();
               if (filename.contains(".xml")) {
                  return p.resolve(filename);
               } else System.err.println("wrong file format");
            }
         } catch (IOException | DirectoryIteratorException x) {
            System.err.println("error!");
            return null;
         }
      }
   }
}
```

LinkedList

```
package lab_4;

import java.lang.Iterable;
import java.util.Iterator;

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;
}
```

Node

```
package lab_4;


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

На цій лабораторної роботи, я розробив консольну програму на java для реалізації діалогового режиму роботи з користувачем. Вирішив прикладну задачу з використанням масивів , рядків , класів , ітераторів і серіалізаторів.