Звіт

Водолазський Микола Анатолійович

КІТ-118а

Лабораторна робота №9

ПАРАМЕТРИЗАЦІЯ В JAVA

Мета:

- Вивчення принципів параметризації в Java.
- Розробка параметризованих класів та методів

Вимоги:

1. Створити власний клас-контейнер, що параметризується (Generic Type), на основі

зв’язних списків для реалізації колекції domain-об’єктів лабораторної роботи №7.

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.

ОПИС ПРОГРАМИ

2.1 Опис змінних:

```
linkedContainer<creationClass> linkedContainer = new linkedContainer<creationClass>()
```

// об’єкт параметризованого контейнера

```
creationClass creationClass1 = new
creationClass(company,specialisation,workingConditions,payment,workingExperience,educ
ation)
```

об’єкт класа кадрового агенства

Scanner scan = new Scanner(System.in); // змінна для активування зчитування з консолі

2.2 Ієрархія та структура класів.

Main – головний клас. Містить метод main(точку входу у програму) та методи по роботі з програмою для реалізації індивідуального завдання.

interface iLinked - інтерфейс контенеру

class creationClass - клас прикладної задачі кадрового агенства

class linkedContainer - параметризований клас-контейнер, котрий зберігає інформацію агенства

ТЕКСТ ПРОГРАМИ

File Main.java:

```
import javax.xml.parsers.ParserConfigurationException;
import javax.xml.transform.TransformerException;
import java.util.Iterator;
import java.io.*;
import java.util.LinkedList;
import java.util.Scanner;

public class Main {

    public static void switcher(linkedContainer linkedContainer, creationClass
creationClass1, XmlWrite xmlWrite, XmlRead xmlRead) throws IOException,
ClassNotFoundException, TransformerException, ParserConfigurationException {
        int choose;


        do{
            System.out.println("Choose action ");
            Scanner in = new Scanner(System.in);
            System.out.println("1. Add elem");
            System.out.println("2. Clear container ");
            System.out.println("3. Convert to Array ");
            System.out.println("4. Create new element ");
            System.out.println("5. Serialize ");
            System.out.println("6. Deserialize ");
            System.out.println("7. Xml serialize");
            System.out.println("8. Xml deserialize");
            choose = in.nextInt();
            switch (choose) {
                case 1:
                    System.out.println(linkedContainer.size());
```

```
                    linkedContainer.addLast(creationClass1);
                    System.out.println(linkedContainer.size());
                    Sort sort = new Sort();
                    sort.Sort(linkedContainer);
                    break;
                //   case 2:
                    // fillingClass.delete();
                  //     linkedContainer.removeByIndex(0);
                    //   break;
                case 2:
                    // fillingClass.cleanElement();
                    linkedContainer.clean();
                    System.out.println(linkedContainer.size());
                    break;
                case 3:

                  // linkedContainer.toArray();

                //System.out.println(linkedContainer.getElementByIndex(0));
                    Object []arr = linkedContainer.toArray().toArray();
                    for(int i=0; i<linkedContainer.size();i++)
                    {
                        System.out.println(arr[i]);
                    }
                    break;
                case 4:
                    Scanner din = new Scanner(System.in);
                    Scanner cin = new Scanner(System.in);
                    System.out.println("Enter company name");
                    String company = din.nextLine();
                    System.out.println("Enter specialisation");
                    String specialisation=din.nextLine();
                    System.out.println("Enter working Conditions");
                    String workingConditions=din.nextLine();
                    System.out.println("Enter payment");
                    int payment=cin.nextInt();
                    System.out.println("Enter working Experience");
                    int workingExperience=cin.nextInt();
                    System.out.println("Enter education");
                    String education=din.nextLine();
                    creationClass1 = new
creationClass(company,specialisation,workingConditions,payment,workingExperience,educ
ation);
                    break;
                case 5:
                    ObjectOutputStream objectOutputStream = new
ObjectOutputStream(new FileOutputStream("store.txt"));
                    objectOutputStream.writeObject(linkedContainer);
                    objectOutputStream.close();
                    break;
                case 6:
                    ObjectInputStream objectInputStream = new ObjectInputStream(new
FileInputStream("store.txt"));
                    linkedContainer<creationClass> newTravels =
(linkedContainer<creationClass>) objectInputStream.readObject();
                    objectInputStream.close();

                    for (creationClass t : newTravels) {
                        System.out.println(t);
                    }

                    break;
```

```
                case 7:
                xmlWrite.write(linkedContainer,"XML.xml");

                    break;
                case 8:

                    linkedContainer<creationClass> newXml = XmlRead.read("XML.xml");

                    for(creationClass t : newXml )
                    {
                        System.out.println(t);
                    }

                    //Object []array = newXml.toArray().toArray();
                    //for(int i=0; i<newXml.size();i++)
                    //{
                     //   System.out.println(array[i]);
                    //}

                    break;
                default:

                    break;
            }}while(choose!=9);
    }

    public static void main(String[] args) throws IOException,
ClassNotFoundException, TransformerException, ParserConfigurationException {

        //создание своего линкед листа
        linkedContainer<creationClass> linkedContainer = new
linkedContainer<creationClass>();
        XmlWrite xmlWrite = new XmlWrite();
        XmlRead xmlRead = new XmlRead();
        //создание объекта
        Scanner in = new Scanner(System.in);
        Scanner cin = new Scanner(System.in);
        System.out.println("Enter company name");
        String company = in.nextLine();
        System.out.println("Enter specialisation");
        String specialisation=in.nextLine();
        System.out.println("Enter working Conditions");
        String workingConditions=in.nextLine();
        System.out.println("Enter payment");
        int payment=cin.nextInt();
        System.out.println("Enter working Experience");
        int workingExperience=cin.nextInt();
        System.out.println("Enter education");
        String education=in.nextLine();
        creationClass creationClass1 = new
creationClass(company,specialisation,workingConditions,payment,workingExperience,educ
ation);


        //delete element by index
        switcher(linkedContainer,creationClass1,xmlWrite, xmlRead);

    }
}
```

LinkedContainer.java:

```java
import java.io.Serializable;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.LinkedList;


public class linkedContainer<E> implements ILinked<E>, Iterable<E>, Serializable{
    private int size = 0;
    private Node<E> first;
    private Node<E> last;

    public linkedContainer() {
        last = new Node<E>(first,null, null);
        first = new Node<E>(null, null, last);
    }

    @Override
    public void addLast(E e) {
        Node<E> prev = last;
        prev.setItem(e);
        last = new Node<E>(prev,null, null);
        prev.setNext(last);
        size++;
    }

    @Override
    public void addFirst(E e) {
        Node<E> next = first;
        next.setItem(e);
        first = new Node<E>(null, null, next);
        next.setPrev(first);
        size++;
    }

    @Override
    public int size() {
        return this.size;
    }

    @Override
    public E getElementByIndex(int index) {
        Node<E> target = first.getNext();
        for (int i = 0; i < index; i++) {
            if (target == null) return null;
            target = target.getNext();
        }
        return target.getItem();
    }

    @Override
    public void removeByIndex(int index) {
        Node<E> target = first.getNext();
        for (int i = 0; i < index; i++) {
            if (target == null) return;
            target = target.getNext();
        }
        Node<E> PrevRemoved = target.prev;
        Node<E> NextRemoved = target.next;
        PrevRemoved.next = NextRemoved;
```

```
        NextRemoved.prev = PrevRemoved;
        target.setItem(null);
        target.setPrev(null);
        target.setNext(null);
        size--;
    }

    @Override
    public void clean() {
        Node<E> target = first.getNext();
        for (int i = 0; i < size; i++) {
            target.setItem(null);
            target = target.getNext();
        }
        last = new Node<E>(first,null, null);
        first = new Node<E>(null, null, last);
        size = 0;
    }



    ArrayList<E> toArray(){
        ArrayList<E> result = new ArrayList<E>();

        Node<E> target = first.getNext();
        for (int i = 0; i < size; i++) {
            result.add(target.getItem());
            target = target.getNext();
        }

        return result;
    }

    @Override
    public String toString() {
        StringBuilder builder = new StringBuilder();
        Node<E> target = first.getNext();
        for (int i = 0; i < size; i++) {
            builder.append(target.item.toString());
            target = target.getNext();
        }
        return builder.toString();
    }

    boolean isEmpry(){
        if(first.next == last){
            return true;
        }else {
            return false;
        }
    }

    @Override
    public Iterator<E> iterator() {
        Iterator<E> iterator = new Iterator<E>() {
            int counter = 0;

            @Override
            public boolean hasNext() {
                return counter < size;
            }
```

```
            @Override
            public E next() {
                return getElementByIndex(counter++);
            }
        };
        return iterator;
    }

    private static class Node<E> implements Serializable {
        E item;
        Node<E> next;
        Node<E> prev;

        Node(Node<E> prev, E element, Node<E> next) {
            this.item = element;
            this.next = next;
            this.prev = prev;
        }

        public E getItem() {
            return item;
        }

        public void setItem(E item) {
            this.item = item;
        }

        public Node<E> getNext() {
            return next;
        }

        public void setNext(Node<E> next) {
            this.next = next;
        }

        public Node<E> getPrev() {
            return prev;
        }

        public void setPrev(Node<E> prev) {
            this.prev = prev;
        }
    }
}
```

creationClass.java :

```
import java.io.Serializable;

public class creationClass implements Serializable {

    private String company;
    private String specialisation;
    private String workingConditions;
    private int payment;
    private int workingExperience;
    private String education;

    creationClass()
    {

    }
```

```
    creationClass(String company, String specialisation, String workingConditions,
int payment, int workingExperience, String education)
    {
        this.company=company;
        this.specialisation=specialisation;
        this.workingConditions=workingConditions;
        this.payment=payment;
        this.workingExperience=workingExperience;
        this.education=education;
    }


    public void setCompany(String company)
    {
        this.company=company;
    }
    public String getCompany()
    {
        return company;
    }

    public String getSpetialisation()
    {
        return specialisation;
    }

    public void setSpetialisation(String spetialisation)
    {
        this.specialisation=spetialisation;
    }

    public String getWorkingConditions()
    {
        return workingConditions;
    }

    public void setWorkingConditions(String workingConditions)
    {
        this.workingConditions=workingConditions;
    }

    public int getPayment()
    {
        return payment;
    }

    public void setPayment(int payment)
    {
        this.payment=payment;
    }

    public int getWorkingExperience()
    {
        return workingExperience;
    }

    public void setWorkingExperience(int workingExperience)
    {
        this.workingExperience=workingExperience;
    }
```

```
    public String getEducation()
    {
        return education;
    }

    public void setEducation(String education)
    {
        this.education=education;
    }

    public void Print()
    {
        System.out.println(getCompany());
    }

    @Override
    public String toString() {
        return "created object{" + "\n" +
                "company name =" + company.toString() + "\n" +
                "specialisation =" + specialisation + "\n" +
                "working conditions =" + workingConditions + "\n" +
                "working experience=" + workingExperience + "\n" +
                "education =" + education + "\n" +
                "payment = " + payment + "\n" +
                '}' + "\n";
    }



}
```

Linked.java:

```
package ua.khpi.oop.vasilchenko09.MyList;

import java.io.Serializable;

public interface Linked<T> extends DescendingIterator<T>, Serializable, Iterable<T> {
    void addLast(T obj);
    void addFirst(T obj);
    int size();
    T getElementByIndex(int index);
    void saveAll();
    void saveRec();
    void add(T obj);
    void clear();
    boolean notEmpty();
    void readRec();
    void readAll();
}
```

DescendingIterator.java:

```
package ua.khpi.oop.vasilchenko09.MyList;

import java.util.Iterator;

public interface DescendingIterator<T> {
    Iterator<T> descendingIterator();
}
```

ВАРІАНТИ ВИКОРИСТАННЯ

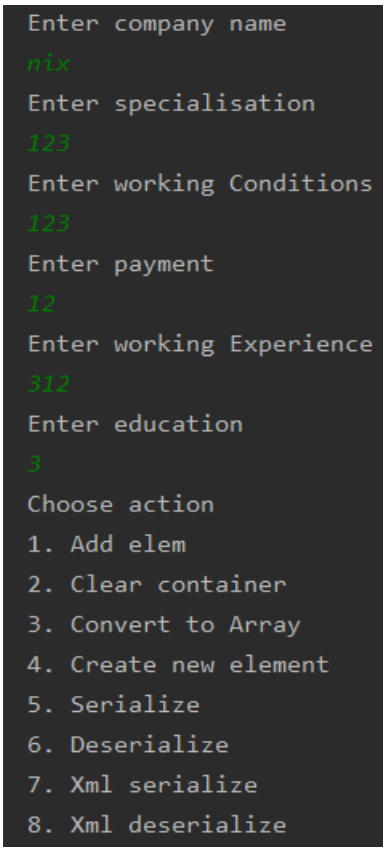

Рис. 9.1 – Результат роботи програми

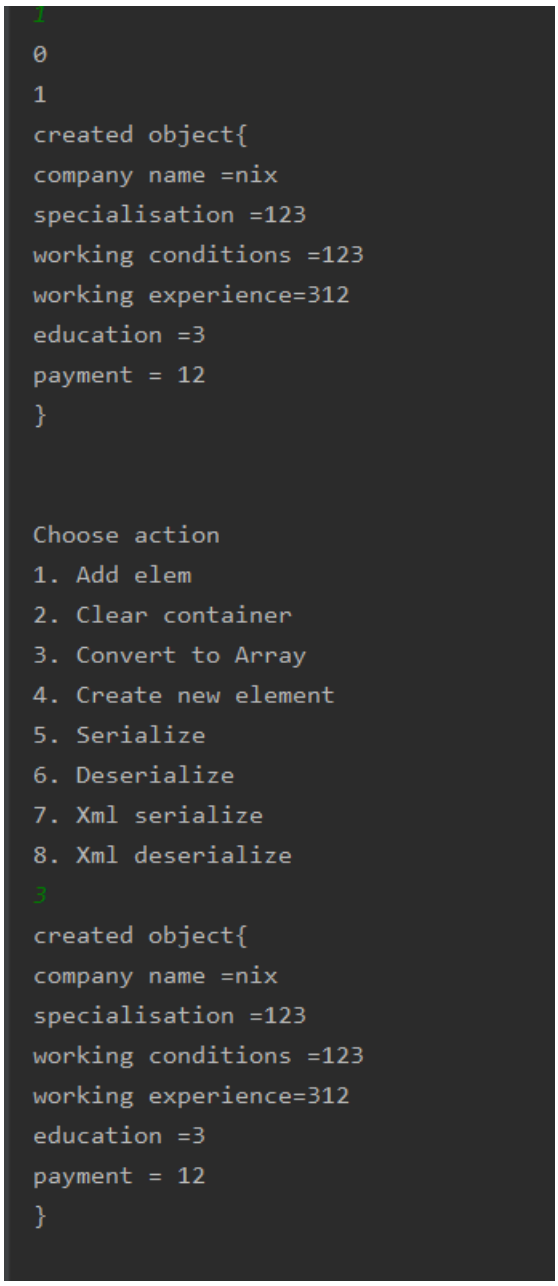

Рис. 9.2 – Результат роботи програми

Програму можна використовувати задля створення бази даних. Завдяки параметризації зв’язного списка, базу даних можна використати для будь-яких типів даних. Переважно у нашому варіанті - кадрове агенство, в якому представляються різноманітні вакансії. Також для вибору доступно багато інших можливостей.

ВИСНОВКИ

При виконанні лабораторної роботи набуто практичних навичок щодо розробки параметризованих класів. Завдяки цієї можливості в JAVA, можливо створювати колекції та інші класи на основі будь-яких типів. Завдання виконане! Програма працює успішно!