Лабораторна робота №9

Параметризація в Java

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

1 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.

1.1 Розробник

- П.І.Б: Заночкин Є. Д.
- Группа: КІТ-119а
- Варіант: 7

2 ОПИС ПРОГРАМИ

2.1 Засоби ООП:

Scanner inInt, inStr = new Scanner(System.in) – для введення обраних опцій користувачем з клавіатури;

XMLEncoder encoder = new XMLEncoder(new BufferedOutputStream(new FileOutputStream("Lab09.xml"));

encoder.writeObject(container); – нестандартна серіалізація;

XMLDecoder decoder = new XMLDecoder(new BufferedInputStream(new FileInputStream("Lab09.xml")));

container = (ClientList<Client>) decoder.readObject(); – нестандартна десеріалізація;

ObjectOutputStream oos = new ObjectOutputStream(new BufferedOutputStream(newFileOutputStream("Lab09.ser")));

oos.writeObject(container);

oos.flush(); – стандартна серіалізація;

ObjectInputStream ois = new ObjectInputStream(new BufferedOutputStream(new FileInputStream("Lab09.ser")));

container = (ClientList<Client>) ois.readObject(); – стандартна десеріалізація;

2.2 Ієрархія та структура класів

Було створено класи Main (головний клас програми), Client (клас, що містить всі поля та методи прикладної області «Бюро знайомств»), ClientList (клас-контейнер) та Node (клас-покажчик на елемент).

2.3 Важливі фрагменти програми

Клас Main

```
package ua.khpi.oop.zanochkyn09;
import java.beans.XMLDecoder;
import java.beans.XMLEncoder;
import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
```

```
import java.util.Scanner;
import ua.khpi.oop.zanochkyn07.InfoAboutYourself;
import ua.khpi.oop.zanochkyn07.PartnerRequirements;
public class Main
{
	public static void main(String[] args)
	{
		ClientList<Client> container = new ClientList<Client>();
		String gender;
		String date;
		InfoAboutYourself info;
		PartnerRequirements requirements;
		int ID = 1;
		boolean endCheck = true;
		Scanner inInt = new Scanner(System.in);
		Scanner inStr = new Scanner(System.in);
		while (endCheck)
		{
			System.out.println("Menu:");
			System.out.println("1. Show clients");
			System.out.println("2. Add client");
			System.out.println("3. Remove client");
			System.out.println("4. Change information");
			System.out.println("5. Clear list");
			System.out.println("6. Serialize data");
			System.out.println("7. Deserialize data");
			System.out.println("8. Count elements in a container");
			System.out.println("0. Exit");
			System.out.println("Enter your option:");
			int option = inInt.nextInt();
			System.out.println();
			switch (option)
			{
			case 1:
				if(container.getSize() > 0)
					System.out.println(container.toString());
				else
					System.out.println("Container is empty.\n");
				break;
			case 2:
				System.out.println("Enter gender:");
				gender = inStr.nextLine();
```

```
                System.out.println("Enter registration date:");
                date = inStr.nextLine();
                System.out.println("Enter information about yourself: Name, age, height, eye colour, hobby.");
                info = new InfoAboutYourself(inStr.nextLine(), inInt.nextInt(), inInt.nextInt(), inStr.nextLine(), inStr.nextLine());
                System.out.println("Enter partner requirements: Gender, min age, max age.");
                requirements = new PartnerRequirements(inStr.nextLine(), inInt.nextInt(), inInt.nextInt());
                container.add(new Client(gender, ID++, date, info, requirements));
                System.out.println("\n" + container.toString());
                break;
            case 3:
                System.out.println("Enter client's ID to remove him:");
                int id = inInt.nextInt();
                int size = container.getSize();
                for(int i = 0; i < container.getSize(); i++)
                    if(container.getElement(i).getId() == id)
                    {
                        container.remove(i);
                        break;
                    }
                if(size == container.getSize())
                    System.out.println("There is no such client");
                else
                    System.out.println("Client removed");
                System.out.println();
                break;
            case 4:
                System.out.println("Enter client's ID to change his information:");
                id = inInt.nextInt();
                int index = 0;
                for(index = 0; index < container.getSize(); index++)
                    if(container.getElement(index).getId() == id)
                        break;
                if(index == container.getSize())
                {
                    System.out.println("There is no client with that ID.\n");
                    break;
                }
                boolean endCheck2 = true;
```

```
int option2 = 0;
while(endCheck2)
{
	System.out.println("\n" + container.getElement(index).toString() + "\n");
	System.out.println("Which information you want to change?");
	System.out.println("1. Gender");
	System.out.println("2. ID");
	System.out.println("3. Registration date");
	System.out.println("4. Information about yourself");
	System.out.println("5. Partner requirements");
	System.out.println("6. End of change");
	System.out.println("Enter option:");
	option2 = inInt.nextInt();
	System.out.println();
	switch(option2)
	{
	case 1:
		System.out.println("Enter new gender:");
	container.getElement(index).setClientGender(inStr.nextLine());
		break;
	case 2:
		System.out.println("Enter new ID:");
		container.getElement(index).setId(inInt.nextInt());
		break;
	case 3:
		System.out.println("Enter new registration date:");
	container.getElement(index).setDate(inStr.nextLine());
		break;
	case 4:
		System.out.println("Information about yourself:");
		System.out.println("1. Name");
		System.out.println("2. Age");
		System.out.println("3. Height");
		System.out.println("4. Eye colour");
		System.out.println("5. Hobby");
		System.out.println("6. Change all information");
		System.out.println("Enter option:");
		int option3 = inInt.nextInt();
		System.out.println();
		switch(option3)
		{
```

```
                    case 1:
                        System.out.println("Enter new name:");
container.getElement(index).getInformation().setName(inStr.nextLine());
                        break;
                    case 2:
                        System.out.println("Enter new age:");
container.getElement(index).getInformation().setAge(inInt.nextInt());
                        break;
                    case 3:
                        System.out.println("Enter new height:");
container.getElement(index).getInformation().setHeight(inInt.nextInt());
                        break;
                    case 4:
                        System.out.println("Enter new eye colour:");
container.getElement(index).getInformation().setEyeColour(inStr.nextLine());
                        break;
                    case 5:
                        System.out.println("Enter new hobby:");
container.getElement(index).getInformation().setClientHobby(inStr.nextLine());
                        break;
                    case 6:
                        System.out.println("Enter information about yourself: Name, age, height, eye colour, hobby.");
                        info = new InfoAboutYourself(inStr.nextLine(), inInt.nextInt(), inInt.nextInt(), inStr.nextLine(), inStr.nextLine());
                        container.getElement(index).setInformation(info);
                        break;
                    default:
                        System.out.println("Wrong command.");
                        break;
                    }
                    break;
                case 5:
                    System.out.println("Partner requirements:");
                    System.out.println("1. Gender");
                    System.out.println("2. Min age");
                    System.out.println("3. Max age");
                    System.out.println("4. Change all requirements");
                    System.out.println("Enter option:");
                    option3 = inInt.nextInt();
                    System.out.println();
                    switch(option3)
```

```
                                        {
                                        case 1:
                                                System.out.println("Enter new gender:");
                        container.getElement(index).getRequirements().setPartnerGender(inStr.nextLine());
                                                break;
                                        case 2:
                                                System.out.println("Enter new min age:");
                                container.getElement(index).getRequirements().setMinAge(inInt.nextInt());
                                                break;
                                        case 3:
                                                System.out.println("Enter new max age:");
                                container.getElement(index).getRequirements().setMaxAge(inInt.nextInt());
                                                break;
                                        case 4:
                                                System.out.println("Enter partner requirements: Gender, min age, max age.");
                                                requirements = new PartnerRequirements(inStr.nextLine(), inInt.nextInt(), inInt.nextInt());
                                        container.getElement(index).setRequirements(requirements);
                                                break;
                                        default:
                                                System.out.println("Wrong command.");
                                                break;
                                        }
                                        break;
                                case 6:
                                        endCheck2 = false;
                                        break;
                                default:
                                        System.out.println("Wrong command.");
                                        break;
                                }
                        }
                        break;
                case 5:
                        container.clear();
                        System.out.println("Container cleared.\n");
                        break;
                case 6:
                        System.out.println("Choose the method");
                        System.out.println("1. Standard serialization");
                        System.out.println("2. XML serialization");
```

```
                System.out.println("3. End");
                System.out.println("Enter your option:");
                option2 = inInt.nextInt();
                System.out.println();
                switch(option2)
                {
                case 1:
                        try(ObjectOutputStream oos = new ObjectOutputStream(new BufferedOutputStream(new FileOutputStream("Lab09.ser"))))
                        {
                                oos.writeObject(container);
                                oos.flush();
                                System.out.println("Serialization successful.\n");
                        }
                        catch(Exception ex)
                        {
                                System.out.println(ex.getMessage() + "\n");
                        }
                        break;
                case 2:
                        try(XMLEncoder encoder = new XMLEncoder(new BufferedOutputStream(new FileOutputStream("Lab09.xml"))))
                        {
                                encoder.writeObject(container);
                                System.out.println("Serialization successful.\n");
                        }
                        catch(Exception ex)
                        {
                                System.out.println(ex.getMessage() + "\n");
                        }
                        break;
                case 3:
                        break;
                default:
                        System.out.println("Wrong command.\n");
                        break;
                }
                break;
        case 7:
                System.out.println("Choose the method");
                System.out.println("1. Standard deserialization");
                System.out.println("2. XML deserialization");
```

```
System.out.println("3. End");
System.out.println("Enter your option");
option2 = inInt.nextInt();
System.out.println();
switch(option2)
{
case 1:
try(ObjectInputStream ois = new ObjectInputStream(new BufferedInputStream(new FileInputStream("Lab09.ser"))))
{
container.clear();
container = (ClientList<Client>) ois.readObject();
System.out.println("Deserialization successful.\n");
}
catch(Exception ex)
{
System.out.println(ex.getMessage());
}
break;
case 2:
try(XMLDecoder decoder = new XMLDecoder(new BufferedInputStream(new FileInputStream("Lab09.xml"))))
{
container.clear();
container = (ClientList<Client>) decoder.readObject();
System.out.println("Deserialization successful.\n");
}
catch(IOException ex)
{
System.out.println(ex.getMessage());
}
break;
case 3:
break;
default:
System.out.println("Wrong command.\n");
break;
}
break;
case 8:
```

```
                                        System.out.println("There are " + container.getSize() + " elements in a container\n");
                                        break;
                                case 0:
                                        endCheck = false;
                                        container.clear();
                                        inInt.close();
                                        inStr.close();
                                        break;
                                default:
                                        System.out.println("Wrong command\n");
                                        break;
                                }
                        }
                        System.out.println("End");
                }
        }
```

Клас Client

```
        package ua.khpi.oop.zanochkyn09;
        import java.io.Serializable;
        import ua.khpi.oop.zanochkyn07.InfoAboutYourself;
        import ua.khpi.oop.zanochkyn07.PartnerRequirements;
        public class Client implements Serializable
        {
                private static final long serialVersionUID = 8633968308489911794L;
                /*
                 * Змінні
                 */
                private String gender;
                private int id;
                private String registrationDate;
                private InfoAboutYourself information;
                private PartnerRequirements requirements;
                /*
                 * Конструктори класу
                 */
                public Client(String gender, int id, String date, InfoAboutYourself info, PartnerRequirements requirements)
                {
                        this.gender = gender;
                        this.id = id;
```

```
        this.registrationDate = date;
        this.information = info;
        this.requirements = requirements;
}
public Client()
{
}
/*
 * Геттери та сеттери
 */
public String getClientGender()
{
        return gender;
}
public void setClientGender(String gender)
{
        this.gender = gender;
}
public int getId()
{
        return id;
}
public void setId(int id)
{
        this.id = id;
}
public String getDate()
{
        return registrationDate;
}
public void setDate(String date)
{
        this.registrationDate = date;
}
public InfoAboutYourself getInformation()
{
        return information;
}
public void setInformation(InfoAboutYourself info)
{
        this.information = info;
}
```

```
		public PartnerRequirements getRequirements()
		{
			return requirements;
		}
		public void setRequirements(PartnerRequirements requirements)
		{
			this.requirements = requirements;
		}
		public String toString()
		{
			return "ID - " + id + "\nRegistration date - " + registrationDate + "\nGender - " + gender +
"\n\n" +
					"Information about yourself:\nName - " + getInformation().getName() +
"\nAge - " + getInformation().getAge() +
					"\nHeight - " + getInformation().getHeight() + "\nEye colour - " +
getInformation().getEyeColour() +
					"\nHobby - " + getInformation().getClientHobby() + "\n\n" +
					"Partner requirements:\nGender - " +
getRequirements().getPartnerGender() +
					"\nMin age - " + getRequirements().getMinAge() + "\nMax age - " +
getRequirements().getMaxAge() +
					"\n----------------------------------------";
		}
	}
```

Клас ClientList

```
	package ua.khpi.oop.zanochkyn09;
	import java.io.Serializable;
	import java.util.Iterator;
	import java.util.NoSuchElementException;
	public class ClientList<T> implements Serializable, Iterable<T>
	{
		private static final long serialVersionUID = 5493313651067238933L;
		public Node<T> head;
		private int size;
		/*
		 * Getter and setter for size
		 */
		public int getSize() { return size; }
		public void setSize(int size) { this.size = size; }
		/*
		 * Method (add) that add a new client into container
```

```
 */
public void add(T el)
{
	Node<T> temp = new Node<T>();
	if(head == null)
		head = new Node<T>(el);
	else
	{
		temp = head;
		while(temp.next != null)
			temp = temp.next;
		temp.next = new Node<T>(el);
	}
	size++;
}
/*
 * Method (remove) that remove a client from container
 */
void remove(int id)
{
	Node<T> temp = head;
	if(head != null)
	{
		if(id == 0)
			head = head.next;
		else
		{
			for(int i = 0; i < id - 1; i++)
				temp = temp.next;
			if(temp.next != null)
				temp.next = temp.next.next;
			else
				temp.next = null;
		}
		size--;
	}
	else
		System.out.println("Container is empty.");
}
/*
 * Method (clear) that clear the container
 */
```

```
void clear()
{
	this.head = null;
	size = 0;
}
/*
 * Method (toArray[]) that return container as an array
 */
public Object[] toArray()
{
	Object[] arr = new Object[size];
	for(int i = 0; i < size; i++)
		arr[i] = getElement(i);
	return arr;
}
/*
 * Method (getElement) that return a specific element from container
 */
public T getElement(int id)
{
	if(id < 0 || id >= size)
	{
		System.out.println("Wrong id.");
		return null;
	}
	Node<T> temp = head;
	for(int i = 0; i < id; i++)
		temp = temp.next;
	return temp.element;
}
/*
 * Method (toString) that return a container as a string
 */
public String toString()
{
	StringBuilder sb = new StringBuilder();
	for(T value : this)
		sb.append(value + "\n");
	return sb.toString();
}
public Iterator<T> iterator()
{
```

```
                return new Iterator<T>()
                {
                        int index = 0;
                        boolean check = false;
                        /*
                         * Method that returns true if the iteration has more elements
                         */
                        @Override
                        public boolean hasNext()
                        {
                                return index < size;
                        }
                        /*
                         * Method that returns the next element in the iteration
                         */
                        @Override
                        public T next()
                        {
                                if (index == size)
                                        throw new NoSuchElementException();
                                check = true;
                                return getElement(index++);
                        }
                        /*
                         * Method that removes from the container the last element returned by this
iterator
                         */
                        @Override
                        public void remove()
                        {
                                if (check)
                                {
                                        ClientList.this.remove(index - 1);
                                        check = false;
                                }
                                else
                                        throw new IllegalStateException();
                        }
                };
        }
}
```

Клас Node

```
package ua.khpi.oop.zanochkyn09;
import java.io.Serializable;
public class Node<T> implements Serializable
{
	private static final long serialVersionUID = -2673405972360871471L;
	public T element;
	public Node<T> next;
	public Node() {}
	public Node(T el)
	{
		super();
		this.element = el;
	}
}
```

3 ВАРІАНТИ ВИКОРИСТАННЯ

У результаті виконання лабораторної роботи було розроблено меню, яке дозволяє користувачу:

1. Вивести усі елементи у консоль (1 команда меню) ;
2. Додати елемент у контейнер (2 команда меню);
3. Видалити елемент з контейнеру (3 команда меню);
4. Редагувати один з елементів (4 команда меню);
5. Очистити контейнер (5 команда меню);
6. Серіалізувати контейнер у файл (6 команда меню);
7. Десеріалізувати контейнер (7 команда меню);
8. Визначити кількість елементів у контейнері (8 команда меню);
9. Закінчити виконання програми (0 команда меню) ;

4 РЕЗУЛЬТАТИ РОБОТИ ПРОГРАМИ

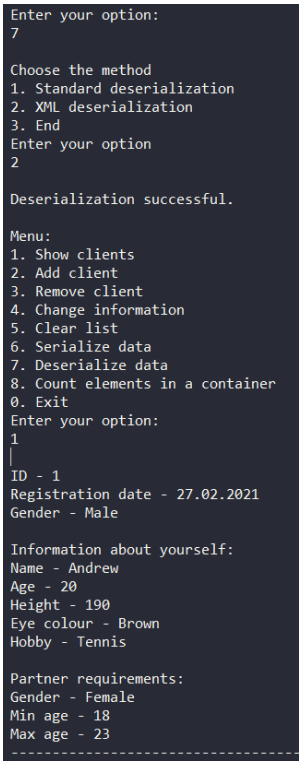

Рисунок 9.1 – Результат роботи програми у середовищі Eclipse

Висновок

Під час виконання лабораторної роботи було набуто навички роботи з параметризацією в середовищі Eclipse IDE.