Лабораторна робота №9

Параметризація в Java

Мета: Оволодіння навичками управління введенням/виведенням даних з використанням класів платформи Java SE.

1 Вимоги

1) Створити власний клас-контейнер, що параметризується (Generic Type), на основі зв’язних списків для реалізації колекції doman-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: Абдулаєв І. З.
- Группа: КІТ-119в
- Варіант: 1

2 Опис програми

2.1 Було використано наступні методи

XMLEncoder encoder = new XMLEncoder(new BufferedOutputStream(new FileOutputStream(file))); encoder.writeObject(recruitingAgency); - XML серіалізація

ObjectOutputStream oos = new ObjectOutputStream(new BufferedOutputStream(new FileOutputStream (filenameSerialization))); oos.writeObject(recruitingAgency); - серіалізація ObjectInputStream ois = new ObjectInputStream(new BufferedInputStream(new FileInputStream (filenameDeserialization))); recruitingAgency.clear(); recruitingAgency = (MyContainer<Challanger>) ois.readObject(); - десеріалізація XMLDecoder decoder = new XMLDecoder(new BufferedInputStream(new FileInputStream (filenameDeserialization))); recruitingAgency.clear(); recruitingAgency = (MyContainer<Challanger>) decoder.readObject(); - XML десеріалізація

2.2 Ієрархія класів

Було створено клас Main (головний клас програми), Challanger (клас, що містить всі поля та методи прикладної області), MyContainer (клас контейнер), Node (клас-покажчик на елемент).

2.3 Важливі фрагменти програми

Class Main

```
package ua.khpi.oop.abdulaev09;

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.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.util.Scanner;

import ua.khpi.oop.abdulaev07.Challanger;
import ua.khpi.oop.abdulaev07.DemandsToWork;
import ua.khpi.oop.abdulaev07.WorkExperience;

public class Main {

    public static void main(String[] args) {
        MyContainer<Challanger> recruitingAgency = new
MyContainer<Challanger>();

        boolean endprog = false;
        Scanner inInt = new Scanner(System.in);
        Scanner inStr = new Scanner(System.in);
        int menu;
```

```
        int menuSerialization;
        int menuDeserialization;

        while(!endprog)
        {
            System.out.println("1. Show all challanger");
            System.out.println("2. Add challanger");
            System.out.println("3. Delete chellanger");
            System.out.println("4. Clear list");
            System.out.println("5. Is empty recruiting agency?");
            System.out.println("6. Serialize data");
            System.out.println("7. Deserialize data");
            System.out.println("0. Exit");
            System.out.print("Enter option: ");
            try
            {
                menu =  inInt.nextInt();
            }
            catch(java.util.InputMismatchException e)
            {
                System.out.println("Error! Ошибка ввода.");
                endprog = true;
                menu = 0;
            }
            System.out.println();
            switch(menu)
            {
                case 1:
                    if(recruitingAgency.getSize() > 0) {
                        for(var element : recruitingAgency) {
                            element.print();
                        }
                    }
                    else {
                        System.out.println("The recruiting agency is
empty!\n");
                    }
                    break;
                case 2:
                    String education;
                    int day;
                    int month;
                    int year;
                    String specializationPrevious;
                    int experience;
                    String specializationNext;
                    int minSalary;
                    String conditions;

                    System.out.println("Enter education of challanger: ");
                    try {
                        education = inStr.nextLine();
                    }catch(java.util.InputMismatchException e) {
                        System.out.println("Error! Incorect input!");
                        break;
                    }

                    System.out.println("Enter day of dismissal: ");
                    try {
```

```
            day = inInt.nextInt();
        } catch(java.util.InputMismatchException e) {
            System.out.println("Error! Incorect input!");
            break;
        }

        System.out.println("Enter month of dismissal: ");
        try {
            month = inInt.nextInt();
        } catch(java.util.InputMismatchException e) {
            System.out.println("Error! Incorect input!");
            break;
        }

        System.out.println("Enter year of dismissal: ");
        try {
            year = inInt.nextInt();
        } catch(java.util.InputMismatchException e) {
            System.out.println("Error! Incorect input!");
            break;
        }

        System.out.println("Enter pervious job: ");
        try {
            specializationPrevious = inStr.nextLine();
        } catch(java.util.InputMismatchException e) {
            System.out.println("Error! Incorect input!");
            break;
        }

        System.out.println("Enter experience of working: ");
        try {
            experience = inInt.nextInt();
        } catch(java.util.InputMismatchException e){
            System.out.println("Error! Incorect input!");
            break;
        }

        System.out.println("Enter next job: ");
        try {
            specializationNext = inStr.nextLine();
        } catch(java.util.InputMismatchException e) {
            System.out.println("Error! Incorect input!");
            break;
        }

        System.out.println("Enter min salary: ");
        try {
            minSalary = inInt.nextInt();
        }catch (java.util.InputMismatchException e) {
            System.out.println("Error! Incorect input!");
            break;
        }

        System.out.println("Enter whishes to the next job: ");
        try {
            conditions = inStr.nextLine();
        } catch(java.util.InputMismatchException e){
            System.out.println("Error! Incorect input!");
```

```
                    break;
                }
                int id = recruitingAgency.getSize();

                WorkExperience workExperienceAdd = new
WorkExperience(specializationPrevious, experience);
                DemandsToWork demandsToWorkAdd = new
DemandsToWork(specializationNext,minSalary,conditions);
                Challanger challangerAdd = new
Challanger(id++,education,day,month,year,workExperienceAdd,demandsToWorkAdd);
                recruitingAgency.add(challangerAdd);
                break;
            case 3:
                System.out.println("Enter ID to delete: ");
                int delete = inInt.nextInt();
                boolean isExist = false;
                if(recruitingAgency.getSize() > 0) {
                    for(var element : recruitingAgency) {
                        if(element.getRegistrationNum() == delete) {
                            isExist = true;
                        }
                    }
                    if(isExist) {
                        if(recruitingAgency.delete(delete))
                            System.out.println("Challanger was deleted
successfully.");
                        else
                            System.out.println("Error! Wrong ID.");
                    }
                    else
                        System.out.println("Error! Wrong ID.");
                }
                break;
            case 4:
                recruitingAgency.clear();
                System.out.println("RecruitingAgency is empty now.\n");
                break;
            case 5:
                if(recruitingAgency.isEmpty())
                    System.out.println("Recruiting agency is empty.\n");
                else
                    System.out.println("Recruiting agency is not
empty.");
                break;
            case 6:
                String filenameSerialization;
                String filenameXML;

                System.out.println("1. Serialization");
                System.out.println("2. XML serialization");
                System.out.println("0. Exit serialization");
                try
                {
                    menuSerialization =  inInt.nextInt();
                }
                catch(java.util.InputMismatchException e)
                {
                    System.out.println("Error! Îøèáêà ââîäà.");
                    menuSerialization = 0;
```

```
                    }
                    switch(menuSerialization)
                    {
                        case 1:
                            System.out.println("\nEnter file name: ");
                            filenameSerialization = inStr.nextLine();
                            if (filenameSerialization.indexOf(".ser") == -1)
{
                                filenameSerialization += ".ser";
                            }
                            try(ObjectOutputStream oos = new
ObjectOutputStream(new BufferedOutputStream(new FileOutputStream
(filenameSerialization)))){
                                oos.writeObject(recruitingAgency);
                                System.out.println("Serialization
successful.");
                            } catch (Exception e){
                                System.out.println(e.getMessage());
                            }
                            break;
                        case 2:
                            System.out.print("Enter XML filename: ");
                            filenameXML = inStr.nextLine();
                            if (filenameXML.indexOf(".xml") == -1)
                                filenameXML += ".xml";
                            try(XMLEncoder encoder = new XMLEncoder(new
BufferedOutputStream(new FileOutputStream (filenameXML)))){
                                encoder.writeObject(recruitingAgency);
                                System.out.println("Serialization
successful.");
                            } catch (Exception e){
                                System.out.println(e.getMessage());
                            }
                            break;
                        case 0:
                            break;
                        default:
                            System.out.println("Error! Wrong num in menu.");
                            break;
                    }
                    break;
                case 7:
                    String filenameDeserialization;

                    System.out.println("1. Deserialization");
                    System.out.println("2. XML deserialization");
                    System.out.println("0. Exit deserialization");
                    try
                    {
                        menuDeserialization =  inInt.nextInt();
                    }
                    catch(java.util.InputMismatchException e)
                    {
                        System.out.println("Error! Ошибка ввода.");
                        menuDeserialization = 0;
                    }
                    switch(menuDeserialization)
                    {
                        case 1:
```

```
                                System.out.println("\nEnter file name: ");
                                filenameDeserialization = inStr.nextLine();
                                if (filenameDeserialization.indexOf(".ser") == -
1) {
                                    filenameDeserialization += ".ser";
                                }
                                try(ObjectInputStream ois = new
ObjectInputStream(new BufferedInputStream(new FileInputStream
(filenameDeserialization)))){
                                    recruitingAgency.clear();
                                    recruitingAgency = (MyContainer<Challanger>)
ois.readObject();
                                    System.out.println("Deserialization
successful.");
                                } catch (Exception e){
                                    System.out.println(e.getMessage());
                                }
                                break;
                            case 2:
                                System.out.print("Enter XML filename: ");
                                filenameDeserialization = inStr.nextLine();
                                if (filenameDeserialization.indexOf(".xml") == -
1)
                                    filenameDeserialization += ".xml";
                                try(XMLDecoder decoder = new XMLDecoder(new
BufferedInputStream(new FileInputStream (filenameDeserialization)))){
                                    recruitingAgency.clear();
                                    recruitingAgency = (MyContainer<Challanger>)
decoder.readObject();
                                    System.out.println("Deserialization
successful.");
                                } catch (Exception e){
                                    System.out.println(e.getMessage());
                                }
                                break;
                            case 0:
                                break;
                            default:
                                System.out.println("Error! Wrong num in menu.");
                                break;
                        }
                        break;
                    case 0:
                        endprog = true;
                        inInt.close();
                        inStr.close();
                        break;
                    default:
                        System.out.println("Error! Wrong num in menu.");
                        break;
                }
            }
        }
```

Class Node

```
package ua.khpi.oop.abdulaev09;

import java.io.Serializable;
```

```
public class Node<T> implements Serializable {
    public T element;
    public Node<T> next;

    private static final long serialVersionUID = -1143293932421725348L;

    public Node() {

    }

    public Node(T element) {
        super();
        this.element = element;
    }
}
```

Class MyContainer

```
package ua.khpi.oop.abdulaev09;

import java.io.Serializable;
import java.util.Iterator;
import java.util.NoSuchElementException;


public class MyContainer<T> implements Iterable<T>, Serializable {
    private static final long serialVersionUID = 7079327902945633955L;

    public Node<T> head;
    private int size;

    public MyContainer() {
        super();
        this.setSize(0);
    }

    public int getSize() {
        return size;
    }
    public void setSize(int size) {
        this.size = size;
    }

    public T getElement(int id) {
        if(id < 0 || id > size) {
            System.out.println("Error! Wrong ID.");
            return null;
        }
        Node<T> temp = head;
        for(int i = 0; id > i; i++)     {
            temp = temp.next;
        }
        return temp.element;
    }

    public void add(T element) {
        Node<T> tmp = new Node<T>();

        if(head == null) {
```

```
            head = new Node<T>(element);
        }
        else {
            tmp = head;
            while(tmp.next != null) {
                tmp = tmp.next;
            }
            tmp.next = new Node<T>(element);
        }
        size++;
    }

    public boolean delete(int id) {
        Node<T> tmp = head;

        if(head != null) {
            if(id == 0) {
                head = head.next;
            }
            else {
                for(int i = 0; id-1 > i; i++) {
                    tmp= tmp.next;
                }
                if(tmp.next != null) {
                    tmp.next = tmp.next.next;
                }
                else
                    tmp.next = null;
                size--;
            }
            return true;
        }
        else {
            System.out.println("Container is empty!");
            return false;
        }
    }

    public void clear() {
        head = null;
        size = 0;
    }

    public Object[] toArray() {
        Object[] array = new Object[size];
        for(int i = 0; size > i; i++) {
            array[i] = getElement(i);
        }
        return array;
    }

    public String toString() {
        StringBuilder str = new StringBuilder();
        for(T element : this) {
            str.append(element + "\n");
        }
        return str.toString();
    }
```

```
    public boolean isEmpty() {
        if(size == 0)
            return true;
        else
            return false;
    }

    public Iterator<T> iterator() {
        return new Iterator<T>(){
            int index = 0;
            boolean check = false;

            @Override
            public boolean hasNext() {
                return size > index;
            }

            @Override
            public T next() {
                if(index != size) {
                    check = true;
                    return getElement(index++);
                }
                else
                    throw new NoSuchElementException();
            }

            @Override
            public void remove() {
                if(check) {
                    MyContainer.this.delete(index - 1);
                    check = false;
                }
            }

        };
    }
}
```

Class Challanger

```
package ua.khpi.oop.abdulaev09;

import java.io.Serializable;

import ua.khpi.oop.abdulaev07.DemandsToWork;
import ua.khpi.oop.abdulaev07.WorkExperience;

public class Challanger implements Serializable {
    private static final long serialVersionUID = -8290634946232397672L;

    private int registrationNum;
    private String education;
    private int dismissalDay;
    private int dismissalMonth;
    private int dismissalYear;
    private DemandsToWork demandsToWork;
    private WorkExperience workExperience;
    /**
     * Конструктор
```

```
     * @param registrationNum ID претндента
     * @param education образование претендента
     * @param dismissalDay день увольнения претендента
     * @param dismissalMonth месяц увольнения претендента
     * @param dismissalYear год увольнения претендента
     * @param workExperience опыт работы претендента
     * @param demandsToWork пожелания к будущей работе
     */
    public Challanger(int registrationNum, String education, int
dismissalDay, int dismissalMonth, int dismissalYear, WorkExperience
workExperience, DemandsToWork demandsToWork ) {
        this.registrationNum = registrationNum;
        this.education = education;
        this.dismissalDay = dismissalDay;
        this.dismissalMonth = dismissalMonth;
        this.dismissalYear = dismissalYear;
        this.workExperience = workExperience;
        this.demandsToWork = demandsToWork;
    }
    public Challanger()
    {
        super();
    }
    /**
     * Геттер ID претендента
     * @return ID претендента
     */
    public int getRegistrationNum() {
        return registrationNum;
    }
    /**
     * Сеттер ID претендента
     * @param registrationNum ID претендента
     */
    public void setRegistrationNum(int registrationNum) {
        this.registrationNum = registrationNum;
    }
    /**
     * Геттер образования претендента
     * @return образование претендента
     */
    public String getEducation() {
        return education;
    }
    /**
     * Сеттер образования претендента
     * @param education Образование претендента
     */
    public void setEducation(String education) {
        this.education = education;
    }
    /**
     * Геттер дня увольнения
     * @return день увольнения
     */
    public int getDismissalDay() {
        return dismissalDay;
    }
    /**
```

```
     * Сеттер дня увольнеия
     * @param dismissalDay день увольнения
     */
    public void setDismissalDay(int dismissalDay) {
        this.dismissalDay = dismissalDay;
    }
    /**
     * Геттер месяца увольнеия
     * @return месяц увольнения
     */
    public int getDismissalMonth() {
        return dismissalMonth;
    }
    /**
     * Сеттер месяца увольнения
     * @param dismissalMonth месяц увольнения
     */
    public void setDismissalMonth(int dismissalMonth) {
        this.dismissalMonth = dismissalMonth;
    }
    /**
     * Геттер года увольнения претендента
     * @return год увольнения
     */
    public int getDismissalYear() {
        return dismissalYear;
    }
    /**
     * Сеттер года увольнения претендента
     * @param dismissalYear год увольнения
     */
    public void setDismissalYear(int dismissalYear) {
        this.dismissalYear = dismissalYear;
    }
    /**
     * Геттер опыта работы претендента
     * @return
     */

    /**
     * Геттер требований к будующей работе
     * @return
     */
    public DemandsToWork getDemandsToWork() {
        return demandsToWork;
    }
    public WorkExperience getWorkExperience() {
        return workExperience;
    }
    public void setWorkExperience(WorkExperience workExperience) {
        this.workExperience = workExperience;
    }
    /**
     * Сеттер требований к будующей работе
     * @param demandsToWork
     */
    public void setDemandsToWork(DemandsToWork demandsToWork) {
        this.demandsToWork = demandsToWork;
    }
```

```
    public void print() {
        System.out.println("ID: " + getRegistrationNum());
        System.out.println("Образование: " + getEducation());
        System.out.println("Дата увольнения: " + getDismissalDay()+"/" +
getDismissalMonth()+"/"+getDismissalYear());
        System.out.println("---Опыт работы---");
        System.out.println("Место предыдущей работы: " +
getWorkExperience().getSpecialization());
        if(getWorkExperience().getExperience() <= 4)
            System.out.println("Стаж: " + getWorkExperience().getExperience()
+ " год(а)");
        else
            System.out.println("Стаж: " + getWorkExperience().getExperience()
+ " лет");
        System.out.println("---Желания по будующей работе---" );
        if(getDemandsToWork().getMinSalary() == 0 &&
getDemandsToWork().getSpecialization() == null &&
getDemandsToWork().getConditions() == null)
            System.out.println("Предендет не имеет никаких желаний по
будующей работе");
        else {
            if(getDemandsToWork().getMinSalary() != 0)
                System.out.println("Желаемая минимальная зарплата: " +
getDemandsToWork().getMinSalary());
            else
                System.out.println("Желаемая минимальная зарплата: Претендент
не имеет пожеланий к этому пункту " );
            if(getDemandsToWork().getSpecialization() != null)
                System.out.println("Желаемая будующая работа: " +
getDemandsToWork().getSpecialization());
            else
                System.out.println("Желаемая будующая работа: Претендент не
имеет пожеланий к этому пункту");
            if(getDemandsToWork().getConditions() != null)
                System.out.println("Желаемые условия будующей работы: " +
getDemandsToWork().getConditions());
            else
                System.out.println("Желаемые условия будующей работы:
Претендент не имеет пожеланий к этому пункту");
        }
        System.out.println("-----------------------------------------");
    }
}
```

3 Результат роботи програми

```
Enter option: 7

1. Deserialization
2. XML deserialization
0. Exit deserialization
2
Enter XML filename: recruitingAgency.xml
Deserialization successful.
1. Show all challanger
2. Add challanger
3. Delete chellanger
4. Clear list
5. Is empty recruiting agency?
6. Serialize data
7. Deserialize data
0. Exit
Enter option: 1

ID: 0
Образование: Higher education
Дата увольнения: 12/5/2020
---Опыт работы---
Место предыдущей работы: HR-manager
Стаж: 2 год(а)
---Желания по будующей работе---
Желаемая минимальная зарплата: 21900
Желаемая будующая работа: HR-manager
Желаемые условия будующей работы: Coffie machine. Free coffie.
----------------------------------------
ID: 1
Образование: School education
Дата увольнения: 13/12/2017
---Опыт работы---
Место предыдущей работы: Delivery guy
Стаж: 1 год(а)
---Желания по будующей работе---
Желаемая минимальная зарплата: 7800
Желаемая будующая работа: Delivery guy
Желаемые условия будующей работы: Free diner. Discounts in delivery company.
```

Рисунок 9.1 – Результат роботи десеріалізації

```
ID: 2
Образование: Higher education
Дата увольнения: 3/11/2010
---Опыт работы---
Место предыдущей работы: Manager in restaurant
Стаж: 13 лет
---Желания по будующей работе---
Желаемая минимальная зарплата: 14900
Желаемая будующая работа: Manager in restaurant
Желаемые условия будующей работы: Free diner. Free coffie.
----------------------------------------
ID: 3
Образование: Higer education
Дата увольнения: 24/5/2020
---Опыт работы---
Место предыдущей работы: Accountant
Стаж: 12 лет
---Желания по будующей работе---
Желаемая минимальная зарплата: 17850
Желаемая будующая работа: Accountant
Желаемые условия будующей работы: Paid vocations.
----------------------------------------
ID: 4
Образование: Higher education
Дата увольнения: 31/5/2020
---Опыт работы---
Место предыдущей работы: Seller
Стаж: 4 год(а)
---Желания по будующей работе---
Желаемая минимальная зарплата: 12400
Желаемая будующая работа: Seller
Желаемые условия будующей работы: Discounts inside the company. Near underground.
----------------------------------------
ID: 5
Образование: Higer education
Дата увольнения: 12/3/2020
---Опыт работы---
Место предыдущей работы: Teacher
Стаж: 8 лет
---Желания по будующей работе---
Желаемая минимальная зарплата: 9600
Желаемая будующая работа: Teacher
Желаемые условия будующей работы: Paid vocations. Near home.
----------------------------------------
```

Рисунок 9.2 – Результат виводу усіх претендентів

```
Enter option: 2

Enter education of challanger:
Higher education
Enter day of dismissal:
11
Enter month of dismissal:
08
Enter year of dismissal:
2020
Enter pervious job:
Barista
Enter experience of working:
3
Enter next job:
Barista
Enter min salary:
9600
Enter whishes to the next job:
Free coffie. Piad vocations. Discounts inside the company.
```

Рисунок 9.3 – Результат додавання претендента

```
----------------------------------------
ID: 6
Образование: Higher education
Дата увольнения: 11/8/2020
---Опыт работы---
Место предыдущей работы: Barista
Стаж: 3 год(а)
---Желания по будующей работе---
Желаемая минимальная зарплата: 9600
Желаемая будующая работа: Barista
Желаемые условия будующей работы: Free coffie. Piad vocations. Discounts inside the company.
```

Рисунок 9.4 – Результат виводу нового претендента

```
1. Show all challanger
2. Add challanger
3. Delete chellanger
4. Clear list
5. Is empty recruiting agency?
6. Serialize data
7. Deserialize data
0. Exit
Enter option: 5

Recruiting agency is not empty.
```

Рисунок 9.5 – Результат перевірки контейнера на наявність елементів

```
1. Show all challanger
2. Add challanger
3. Delete chellanger
4. Clear list
5. Is empty recruiting agency?
6. Serialize data
7. Deserialize data
0. Exit
Enter option: 3

Enter ID to delete:
3
Challanger was deleted successfully.
```

Рисунок 9.6 – Видалення претендента с ID 3

```
----------------------------------------
ID: 1
Образование: School education
Дата увольнения: 13/12/2017
---Опыт работы---
Место предыдущей работы: Delivery guy
Стаж: 1 год(а)
---Желания по будующей работе---
Желаемая минимальная зарплата: 7800
Желаемая будующая работа: Delivery guy
Желаемые условия будующей работы: Free diner. Discounts in delivery company.
----------------------------------------
ID: 2
Образование: Higher education
Дата увольнения: 3/11/2010
---Опыт работы---
Место предыдущей работы: Manager in restaurant
Стаж: 13 лет
---Желания по будующей работе---
Желаемая минимальная зарплата: 14900
Желаемая будующая работа: Manager in restaurant
Желаемые условия будующей работы: Free diner. Free coffie.
----------------------------------------
ID: 4
Образование: Higher education
Дата увольнения: 31/5/2020
---Опыт работы---
Место предыдущей работы: Seller
Стаж: 4 год(а)
---Желания по будующей работе---
Желаемая минимальная зарплата: 12400
Желаемая будующая работа: Seller
Желаемые условия будующей работы: Discounts inside the company. Near underground.
----------------------------------------
```

Рисунок 9.7 – Результат видалення

```
1. Show all challanger
2. Add challanger
3. Delete chellanger
4. Clear list
5. Is empty recruiting agency?
6. Serialize data
7. Deserialize data
0. Exit
Enter option: 6

1. Serialization
2. XML serialization
0. Exit serialization
1

Enter file name:
new
Serialization successful.
```

Рисунок 9.8 – Виконання серіалізації

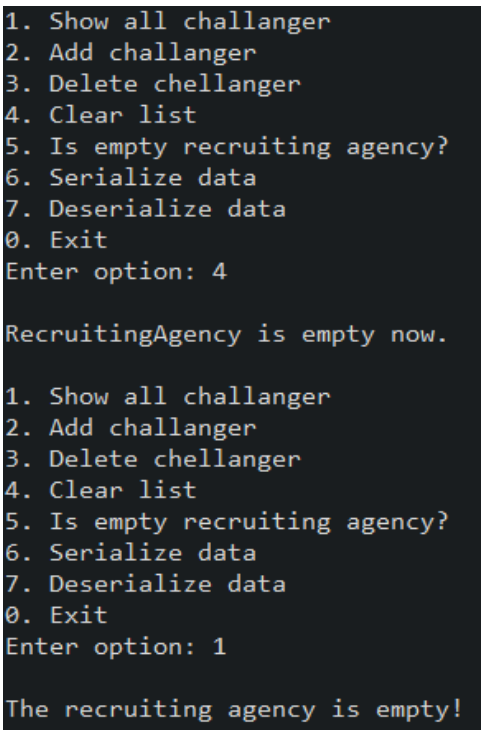

Рисунок 9.9 – Результат очищення контейнера

Висновок: Під час виконання лабораторної роботи було набуто навички роботи з параметризацією в середовищі IntelliJ IDEA.