

Лабораторна робота №10

Обробка параметризованих контейнерів

Мета: Розширення функціональності параметризованих класів.

1 ВИМОГИ

1. Розробити параметризовані методи (Generic Methods) для обробки колекцій об'єктів згідно прикладної задачі.
2. Продемонструвати розроблену функціональність (створення, управління та обробку власних контейнерів) в діалоговому та автоматичному режимах.
 - Автоматичний режим виконання програми задається параметром командного рядка -auto. Наприклад, java ClassName -auto.
 - В автоматичному режимі діалог з користувачем відсутній, необхідні данні генеруються, або зчитуються з файлу.
3. Забороняється використання алгоритмів з Java Collections Framework.

1.1 Розробник

- П.І.Б: Абдулаєв І. З.
- Група: КІТ-119в
- Варіант: 1

1.2 Завдання

Реалізувати сортування за реєстраційним номером, за сумарним стажем, за середнім значенням необхідної мінімальної зарплатні.

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("filename")));

encoder.writeObject(recuitingAgency); – нестандартна серіалізація;

```
XMLDecoder decoder = new XMLDecoder(new BufferedInputStream(new
FileInputStream("filename")));

container = (MyContainer<Challanger>) decoder.readObject(); –
нестандартна десеріалізація;

oos.writeObject(container);

ObjectInputStream ois = new ObjectInputStream(new
BufferedOutputStream(new FileInputStream("Lab10.ser")));

container = (MyContainer<Challanger>) ois.readObject(); – стандартна
десеріалізація;
```

2.2 Ієрархія та структура класів

Було створено класи Main (головний клас програми), Challenger (клас, що містить всі поля та методи прикладної області «Кадрове агенство»), MyConatainer (клас-контейнер), Node (клас-показчик на елемент) та 3 класи, що реалізують інтерфейс Comparator для сортування за певними критеріями.

2.3 Важливі фрагменти програми

Class Main

```
package ua.khpi.oop.abdulaev10;

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>();
```

```

        for (String str : args) {
            if(str.equals("-a") || str.equals("-auto")) {
                recruitingAgency = auto(recruitingAgency);
                return;
            }
        }
        recruitingAgency = menu(recruitingAgency);
    }

    private static MyContainer<Challenger>
    auto(MyContainer<Challenger> recruitingAgency) {
        System.out.println("Adding elements...");
        String filenameDeserialization =
            "recruitingAgency10.xml";
        try(XMLDecoder decoder = new XMLDecoder(new
            BufferedInputStream(new FileInputStream
            (filenameDeserialization)))){
            recruitingAgency.clear();
            recruitingAgency = (MyContainer<Challenger>)
            decoder.readObject();
        } catch (Exception e){
            System.out.println(e.getMessage());
        }

        System.out.println("Adding was end.\n");

        System.out.println("List in Recruiting Agency:\n");
        if(recruitingAgency.getSize() > 0) {
            for(var element : recruitingAgency) {
                element.print();
            }
        }
        else {
            System.out.println("The recruiting agency is
            empty!\n");
        }

        int orderSort = 1;

        recruitingAgency.sort(new workExperienceComparator(),
        orderSort);
        System.out.println("Data sorted by work experience");

        System.out.println("List in Recruiting Agency:\n");
        if(recruitingAgency.getSize() > 0) {
            for(var element : recruitingAgency) {
                element.print();
            }
        }
    }

```

```

    }

    return recruitingAgency;
}

private static MyContainer<Challanger>
menu(MyContainer<Challanger> recruitingAgency) {
    boolean endprog = false;
    Scanner inInt = new Scanner(System.in);
    Scanner inStr = new Scanner(System.in);
    int menu;
    int menuSort;
    int orderSort;
    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. Sort data");
        System.out.println("7. Serialize data");
        System.out.println("8. 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
challenger: ");
    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 wishes 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);
        Challenger challengerAdd = new
Challenger(id++, education, day, month, year, workExperienceAdd, deman
dsToWorkAdd);

        recruitingAgency.add(challengerAdd);
        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("Challenger
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:
        System.out.println("1. Sort by Registration
Number");
        System.out.println("2. Sort by work
experience");
        System.out.println("3. Sort by demand to min
salary");
        System.out.println("4. Return to menu");
        System.out.println("Enter option: ");
        try
        {
            menuSort = inInt.nextInt();
        }
        catch(java.util.InputMismatchException e)
        {
            System.out.println("Error! Ошибка
ввода.");
            break;
        }
        System.out.println();
        System.out.println("How to sort data?");
        System.out.println("1. Asc");
        System.out.println("2. Desc");
        System.out.println("Enter option: ");
        try
        {
            orderSort = inInt.nextInt();
        }
        catch(java.util.InputMismatchException e)
        {
            System.out.println("Error! Ошибка
ввода.");
            break;
        }
        switch(menuSort) {
            case 1:
                recruitingAgency.sort(new
idComparator(), orderSort);
                System.out.println("Data sorted by
Registration Number\n");

```



```

        break;
    case 2:
        recruitingAgency.sort(new
workExperienceComparator(), orderSort);
        System.out.println("Data sorted by
work experience\n");
        break;
    case 3:
        recruitingAgency.sort(new
minSalazyComparator(), orderSort);
        System.out.println("Data sorted by
demand to min salary");
        break;
    case 4:

        break;
    default:
        System.out.println("Error! Wrong num
in Sort menu.");
        break;
    }
    break;
case 7:
    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! Ошибка
ввода.");
        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 8:
    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<Challenger>) 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<Challenger>) 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;
    }
}
return recruitingAgency;
}
}

```

Class Node

```

package ua.khpi.oop.abdulaev10;

import java.io.Serializable;

public class Node<T> implements Serializable {

    public T element;
    public Node<T> next;

    private static final long serialVersionUID = -
7470918086342495897L;

    public Node() {

    }

    public Node(T element) {

```

```

        super();
        this.element = element;
    }
}

```

Class MyContainer

```

package ua.khpi.oop.abdulaev10;

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

import ua.khpi.oop.abdulaev07.Challenger;

public class MyContainer<T> implements Iterable<T>, Serializable
{
    private static final long serialVersionUID =
1487028470983100792L;

    public Node<T> head;
    private int size;

    public MyContainer() {
        super();
    }

    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;
            }
        }
    };
};

```

```

    }

    public void sort (Comparator<T> comp, int order) {
        Object[] array = this.toArray();
        Object temp;
        boolean check;

        if (order == 1) {
            do {
                check = false;
                for(int i = 0; size - 1 > i; i++) {
                    if(comp.compare((T)array[i], (T)array[i+1])
== 1) {

                        temp = array[i];
                        array[i] = array[i + 1];
                        array[i + 1] = temp;
                        check = true;
                    }
                }
            } while (check == true);
        }
        else {
            do {
                check = false;
                for(int i = 0; size - 1 > i; i++) {
                    if(comp.compare((T)array[i], (T)array[i+1])
== -1) {

                        temp = array[i+1];
                        array[i+1] = array[i];
                        array[i] = temp;
                        check = true;
                    }
                }
            } while (check == true);
        }

        this.clear();
        for(Object obj : array) {
            this.add((T)obj);
        }
    }
}

class idComparator implements Comparator<Challenger>{
    @Override
    public int compare(Challenger o1, Challenger o2) {
        if(o1.getRegistrationNum() > o2.getRegistrationNum())
            return 1;
        else if (o1.getRegistrationNum() <

```



```

o2.getRegistrationNum())
        return -1;
    else
        return 0;
    }
}

class workExperienceComparator implements
Comparator<Challenger>{
    @Override
    public int compare(Challenger o1, Challenger o2) {
        if(o1.getWorkExperience().getExperience() >
o2.getWorkExperience().getExperience())
            return 1;
        else if (o1.getWorkExperience().getExperience() <
o2.getWorkExperience().getExperience())
            return -1;
        else
            return 0;
    }
}

class minSalazyComparator implements Comparator<Challenger>{
    @Override
    public int compare(Challenger o1, Challenger o2) {
        if(o1.getDemandsToWork().getMinSalary() >
o2.getDemandsToWork().getMinSalary())
            return 1;
        else if (o1.getDemandsToWork().getMinSalary() <
o2.getDemandsToWork().getMinSalary())
            return -1;
        else
            return 0;
    }
}

```

Результат роботи програми

```
Enter option: 8
1. Deserialization
2. XML deserialization
0. Exit deserialization
2
Enter XML filename: recruitingAgency10
Deserialization successful.
1. Show all challenger
2. Add challenger
3. Delete challenger
4. Clear list
5. Is empty recruiting agency?
6. Sort data
7. Serialize data
8. Deserialize data
0. Exit
Enter option: 1
ID: 0
Образование: Higher education
Дата увольнения: 13/5/2020
---Опыт работы---
Место предыдущей работы: HR-manager
Стаж: 2 год(а)
---Желания по будущей работе---
Желаемая минимальная зарплата: 32500
Желаемая будущая работа: HR-manager
Желаемые условия будущей работы: Free coffee. Three working day in a week. Two month of vacations.
-----
```

Рисунок 10.1 – Результат роботи десеріалізації

```
ID: 1
Образование: High school
Дата увольнения: 12/12/2012
---Опыт работы---
Место предыдущей работы: Delivery boy
Стаж: 4 год(а)
---Желания по будущей работе---
Желаемая минимальная зарплата: 76000
Желаемая будущая работа: Programmer
Желаемые условия будущей работы: Huge salary
-----
ID: 2
Образование: High school
Дата увольнения: 15/4/2021
---Опыт работы---
Место предыдущей работы: Teacher
Стаж: 24 лет
---Желания по будущей работе---
Желаемая минимальная зарплата: 4600
Желаемая будущая работа: Teacher
Желаемые условия будущей работы: Possibility to have a nap
-----
ID: 3
Образование: Higher education
Дата увольнения: 26/3/2021
---Опыт работы---
Место предыдущей работы: Programmer
Стаж: 10 лет
---Желания по будущей работе---
Желаемая минимальная зарплата: 56800
Желаемая будущая работа: Senior Programmer
Желаемые условия будущей работы: One paid month of vacation.
-----
```

Рисунок 10.2 – Результат виводу усіх претендентів

```
1. Show all challenger
2. Add challenger
3. Delete challenger
4. Clear list
5. Is empty recruiting agency?
6. Sort data
7. Serialize data
8. Deserialize data
0. Exit
Enter option: 5
|
Recruiting agency is not empty.
```

Рисунок 10.3 – Результат перевірки контейнера на наявність елементів

```
Enter option: 6

1. Sort by Registration Number
2. Sort by work experience
3. Sort by demand to min salary
4. Return to menu
Enter option:
3

How to sort data?
1. Asc
2. Desc
Enter option:
2
Data sorted by demand to min salary
```

Рисунок 10.4 – Сортування за бажаною мінімальною зарплатнею

```

ID: 1
Образование: High school
Дата увольнения: 12/12/2012
---Опыт работы---
Место предыдущей работы: Delivery boy
Стаж: 4 год(а)
---Желания по будущей работе---
Желаемая минимальная зарплата: 76000
Желаемая будущая работа: Programmer
Желаемые условия будущей работы: Huge salary
-----
ID: 3
Образование: Higher education
Дата увольнения: 26/3/2021
---Опыт работы---
Место предыдущей работы: Programmer
Стаж: 10 лет
---Желания по будущей работе---
Желаемая минимальная зарплата: 56800
Желаемая будущая работа: Senior Programmer
Желаемые условия будущей работы: One paid month of vocation.
-----
ID: 0
Образование: Higer education
Дата увольнения: 13/5/2020
---Опыт работы---
Место предыдущей работы: HR-manager
Стаж: 2 год(а)
---Желания по будущей работе---
Желаемая минимальная зарплата: 32500
Желаемая будущая работа: HR-manager
Желаемые условия будущей работы: Free coffie. Three working day in a week. Two month of vocations.
-----
ID: 2
Образование: High school
Дата увольнения: 15/4/2021
---Опыт работы---
Место предыдущей работы: Teacher
Стаж: 24 лет
---Желания по будущей работе---
Желаемая минимальная зарплата: 4600
Желаемая будущая работа: Teacher
Желаемые условия будущей работы: Posibility to have a nap

```

Рисунок 10.5 – Результат сортування

```

1. Show all challenger
2. Add challenger
3. Delete challenger
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 10.6 – Виконання серіалізації

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

RecruitingAgency is empty now.

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

The recruiting agency is empty!
```

Рисунок 10.7 – Результат очищення контейнера

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