Лабораторна робота №11

Регулярні вирази. Перевірка даних

Мета: Ознайомлення з принципами використання регулярних виразів для перевірки рядка на відповідність шаблону.

1 ВИМОГИ

1. Продемонструвати ефективне (оптимальне) використання регулярних виразів для перевірки коректності (валідації) даних, що вводяться, перед записом в domain-об'єкти відповідно до призначення кожного поля для заповнення розробленого контейнера:

- при зчитуванні даних з текстового файла в автоматичному режимі;
- при введенні даних користувачем в діалоговому режимі.

1.1 Розробник

- П.І.Б: Абдуллін О. Р.
- Группа: КІТ-119а
- Варіант: 1

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);

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

Pattern pattern = Pattern.compile() – компілює регулярний вираз у шаблон;

Matcher matcher = pattern.matcher(information); – створює matcher, який буде відповідати даному вводу для цього шаблону.

2.2 Ієрархія та структура класів

Було створено класи Main (головний клас програми), Challanger (клас, що містить всі поля та методи прикладної області «Кадрове агенство»), MyConatainer (клас-контейнер), Node (клас-покажчик 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.abdullin11;

import java.beans.XMLDecoder;
import java.beans.XMLEncoder;
import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.util.Scanner;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

import ua.khpi.oop.abdullin07.Challanger;
import ua.khpi.oop.abdullin07.DemandsToWork;
import ua.khpi.oop.abdullin07.WorkExperience;
import ua.khpi.oop.abdullin10.MyContainer;

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<Challanger> auto(MyContainer<Challanger> recruitingAgency) {
		System.out.println("Adding elements...");

		File file = new File("recruitingAgency11.txt");

		try {
			String education;
			int day;
			int month;
			int year;
			String specializationPrevious;
			int experience;
			String specializationNext;
			int minSalary;
			String conditions;
			Scanner reader = new Scanner(file);
			while(reader.hasNextLine()) {
				String data = reader.nextLine();
				Pattern pattern = Pattern.compile("((\\w+(|\\s))*,\\s([1-9]|[12]\\d|3[01])\\.([1-
9]|1[012])\\.((19|20)\\d{2}),\\s" +

"(\\w+.)+,\\s([0-9]|[1-6][0-9]),\\s(\\w+.)+,\\s([1-9]\\d{3,}),\\s(\\w+(\\.|\\s)(\\s|))+)");
				Matcher matcher = pattern.matcher(data);
				if(matcher.matches()) {
```

```
                                        String[] information = data.split(",\\s");
                                        education = information[0];
                                        specializationPrevious = information[2];
                                        experience = Integer.parseInt(information[3]);
                                        specializationNext = information[4];
                                        minSalary = Integer.parseInt(information[5]);
                                        conditions = information[6];
                                        String[] date = information[1].split("\\.");
                                        day = Integer.parseInt(date[0]);
                                        month = Integer.parseInt(date[1]);
                                        year = Integer.parseInt(date[2]);

                                        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);
                                }
                        }
                        reader.close();
                        System.out.println("Adding was end.\n");
                } catch (FileNotFoundException e){
                        e.printStackTrace();
                }

                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;

Pattern patternEducation = Pattern.compile("(\\w+.)+");
Pattern patternDay = Pattern.compile("([1-9]|[12]\\d|3[01])");
Pattern patternMonth = Pattern.compile("([1-9]|1[012])");
Pattern patternYear = Pattern.compile("(19|20)\\d{2}");
Pattern patternSpecialization = Pattern.compile("(\\w+.)+");
Pattern patternExperience = Pattern.compile("[0-9]|[1-6][0-9]");
Pattern patternMinSalary = Pattern.compile("(^[1-9]\\d{3,})");
Pattern patternConditions = Pattern.compile("(\\w+(\\.|\\s)(\\s|))+");

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

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

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

```
                break;
            }

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

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

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

            System.out.println("Enter next job: ");
            try {
                specializationNext = inStr.nextLine();
                specializationNext = stringRegexCheck(specializationNext,
patternSpecialization);
            } catch(java.util.InputMismatchException e) {
```

```
                    System.out.println("Error! Incorect input!");
                    break;
                }

                System.out.println("Enter min salary: ");
                try {
                    minSalary = inInt.nextInt();
                    minSalary = intRegexCheck(minSalary, patternMinSalary);
                }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();
                    conditions = stringRegexCheck(conditions, patternConditions);
                } 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:
            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<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;
                }
        }
        return recruitingAgency;
}
public static int intRegexCheck(int value, Pattern pattern)
{
        Matcher matcher;
        Scanner in = new Scanner(System.in);
        boolean ready = false;
        do
        {
                matcher = pattern.matcher(Integer.toString(value));
                if(!matcher.matches())
                {
                        System.out.println("You've entered the wrong data. Try again:");
                        value = in.nextInt();
                }
                else
                        ready = true;
        }
        while(!ready);
        return value;
}

public static String stringRegexCheck(String value, Pattern pattern)
{
        Matcher matcher;
        Scanner in = new Scanner(System.in);
```

```
                boolean ready = false;
                do
                {
                        matcher = pattern.matcher(value);
                        if(!matcher.matches())
                        {
                                System.out.println("You've entered the wrong data. Try again:");
                                value = in.nextLine();
                        }
                        else
                                ready = true;
                }
                while(!ready);
                return value;
        }
}
```

Class MyContainer

```
package ua.khpi.oop.abdullin11;

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

import ua.khpi.oop.abdullin07.Challanger;
import ua.khpi.oop.abdullin10.Node;



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<Challanger>{
        @Override
        public int compare(Challanger o1, Challanger o2) {
                if(o1.getRegistrationNum() > o2.getRegistrationNum())
                        return 1;
                else if (o1.getRegistrationNum() < o2.getRegistrationNum())
                        return -1;
                else
                        return 0;
        }
}

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

}
```

3 Результат роботи програми

```
Adding elements...
Adding was end.

List in Recruiting Agency:

ID: 0
Образование: School education
Дата увольнения: 31/5/2020
---Опыт работы---
Место предыдущей работы: HR-manager
Стаж: 4 год(а)
---Желания по будующей работе---
Желаемая минимальная зарплата: 5600
Желаемая будующая работа: Waiter
Желаемые условия будующей работы: Posibility to dose not have buisness trip. Paid vocations. Free dinner. Free cofie.
-----------------------------------------
ID: 1
Образование: Higher education
Дата увольнения: 23/3/2021
---Опыт работы---
Место предыдущей работы: Waiter
Стаж: 3 год(а)
---Желания по будующей работе---
Желаемая минимальная зарплата: 17800
Желаемая будующая работа: Programmer
Желаемые условия будующей работы: Possibility to have buisness trip. Coffie machine in the office. Near home.
-----------------------------------------
Data sorted by work experience
```

Рисунок 11.1 – Результат роботи програми у автоматичкому режимі

```
List in Recruiting Agency:

ID: 1
Образование: Higher education
Дата увольнения: 23/3/2021
---Опыт работы---
Место предыдущей работы: Waiter
Стаж: 3 год(а)
---Желания по будующей работе---
Желаемая минимальная зарплата: 17800
Желаемая будующая работа: Programmer
Желаемые условия будующей работы: Possibility to have buisness trip. Coffie machine in the office. Near home.
-----------------------------------------
ID: 0
Образование: School education
Дата увольнения: 31/5/2020
---Опыт работы---
Место предыдущей работы: HR-manager
Стаж: 4 год(а)
---Желания по будующей работе---
Желаемая минимальная зарплата: 5600
Желаемая будующая работа: Waiter
Желаемые условия будующей работы: Posibility to dose not have buisness trip. Paid vocations. Free dinner. Free cofie.
```

Рисунок 11.2 – Результат роботи програми у автоматичкому режимі

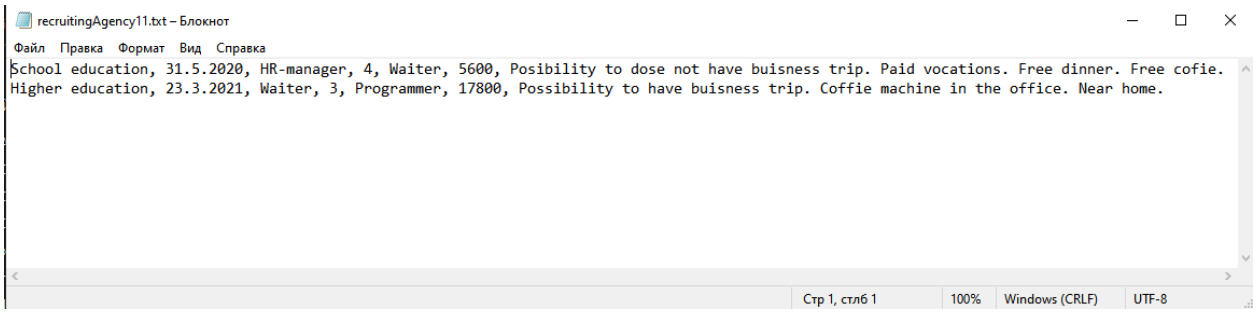

Рисунок 11.3 – Текстовий файл з претендентами

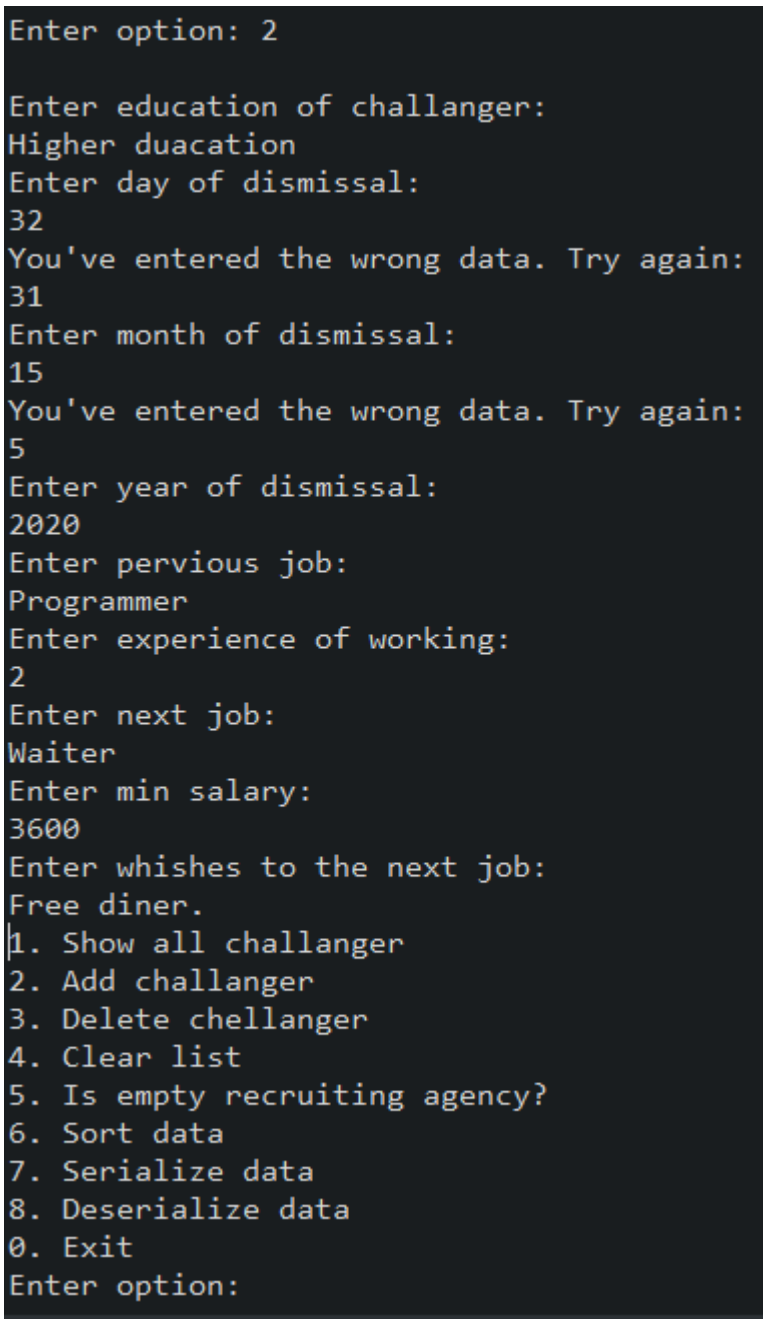

Рисунок 11.4 – Тестування регулярних виразів

Висновок

Під час виконання лабораторної роботи було набуто навички роботи з розробки регулярних виразів та перевірки даних за їх допомогою в середовищі Eclipse IDE.