

Звіт

Автор: Водолазський Микола Анатолійович

KIT-118a

Лабораторна робота №11

Регулярні вирази. Перевірка даних

Мета:

- Розширення функціональності параметризованих класів.

Вимоги:

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.
-

ПРИКЛАДНА ЗАДАЧА:

Кадрове агентство. Сортуння за назвою фірми, за назвою запропонованої спеціальності, за вказаною освітою.

ОПИС ПРОГРАМИ

2.1 Опис змінних:

```
LinkedContainer<SecondCreate> stringLinked = new LinkedContainer<>();//  
об'єкт параметризованого контейнера
```

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

2.2 Ієрархія та структура класів.

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

interface iLinked - інтерфейс контейнеру

class SecondCreate - клас прикладної задачі кадрового агенства

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

ТЕКСТ ПРОГРАМИ

File Main.java:

```
import javax.xml.parsers.ParserConfigurationException;
import javax.xml.transform.TransformerException;
import java.io.*;
import java.util.Arrays;
import java.util.Comparator;
import java.util.LinkedList;
import java.util.Scanner;
import java.io.*;
import java.util.concurrent.*;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class Main implements Serializable {

    public static void fileRead() throws IOException, ParserConfigurationException,
TransformerException, InterruptedException, ExecutionException, TimeoutException {
        linkedContainer<SecondCreate> linkedContainer = new
linkedContainer<SecondCreate>();

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

        Scanner scanner = new Scanner(file);

        String line = scanner.nextLine();
        String[] array = line.split(" ");

        String company = null;
        String specialisation = null;
        String workingConditions = null;
        int payment = 0;
        int workingExperience = 0;
        String education = null;
        String English = null;
        String Licence = null;

        for (int i = 0; i < array.length; i++) {
            company = array[0].toString();
            specialisation = array[1].toString();
            workingConditions = array[2].toString();
            payment = Integer.parseInt(array[3]);
            workingExperience = Integer.parseInt(array[4]);
            education = array[5].toString();
            Licence = array[6].toString();
            English = array[7].toString();
        }
    }
}
```

```

regCheck(company,specialisation,workingConditions,payment,workingExperience,education
,Licence, English);

        SecondCreate firstWorker = new SecondCreate(company, specialisation,
workingConditions, payment, workingExperience, education,Licence,English);
        SecondCreate secondCreate = new SecondCreate("epam", "teacher", "good", 100,
1, "none","no","no");
        SecondCreate thirdWorker = new SecondCreate("globalLogic","teacher","10.00-
19.00",300,11,"magistry","yes","yes");

        SecondCreate[] arr = {firstWorker, secondCreate,thirdWorker};

        linkedContainer.addFirst(firstWorker);

        System.out.println("Container size");
        System.out.println(linkedContainer.size());

        serialize(linkedContainer);

        linkedContainer.addLast(secondCreate);
        linkedContainer.addLast(thirdWorker);

        // конец потоков без ограничения по времени

        // Поиск элемента соответствующего заданым критериям
        //textsort(linkedContainer);

    }

    public static void serialize(linkedContainer<SecondCreate> linkedContainer)
throws IOException, ParserConfigurationException, TransformerException{
        XmlRead xmlRead = new XmlRead();
        XmlWrite xmlWrite = new XmlWrite();
        xmlWrite.write(linkedContainer,"XML.xml");

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

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

    }

    public static void regCheck(String company, String specialisation, String
workingConditions, int payment, int workingExperience, String education, String
License, String English)
    {
        if(company.matches("[a-zA-Z0-9]*") == true)
        {
            System.out.println("OK");
        }
        else

```

```

    {
        System.out.println("Rename company");
    }
    if(specialisation.matches("[0-9]*"))
    {
        System.out.println("OK");
    }else{System.out.println("Rename specialisation");}
}

    public static void manual() throws IOException, ClassNotFoundException,
    FileNotFoundException, TransformerException, ParserConfigurationException {
        System.out.println("U have chosen manual mode");
        int choose;

        linkedContainer<SecondCreate> linkedContainer = new linkedContainer<>();
        SecondCreate SecondCreate1 = null;

        do{
            System.out.println("Choose action ");
            Scanner in = new Scanner(System.in);
            System.out.println("1. Create new element");
            System.out.println("2. Add elem ");
            System.out.println("3. Clear container ");
            System.out.println("4. Convert to Array ");
            System.out.println("5. Serialize ");
            System.out.println("6. Deserialize ");
            System.out.println("7. Xml serialize");
            System.out.println("8. Xml deserialize");
            choose = in.nextInt();
            switch (choose) {
                case 1:
                    Scanner din = new Scanner(System.in);
                    Scanner cin = new Scanner(System.in);
                    System.out.println("Enter company name");
                    String company = din.nextLine();
                    if(company.matches("[a-zA-Z0-9]*")==true)
                    {
                        System.out.println("");
                    }else{System.out.println("NOT ok RENAME");company =
din.nextLine();}
                    System.out.println("Enter specialisation");
                    String specialisation=din.nextLine();
                    if(specialisation.matches("[0-9]*")==true)
                    {
                        System.out.println("ok");
                    }else{ System.out.println("NOT ok RENAME"); specialisation =
din.nextLine();}
                    System.out.println("Enter working Conditions");
                    String workingConditions=din.nextLine();
                    System.out.println("Enter payment");
                    int payment=cin.nextInt();
                    System.out.println("Enter working Experience");
                    int workingExperience=cin.nextInt();
                    System.out.println("Enter education");
                    String education=din.nextLine();
                    System.out.println("Enter knowledge of English");
                    String English = cin.nextLine();
                    System.out.println("Enter driving licence");
                    String License = cin.nextLine();
                    SecondCreate1 = new

```

```

SecondCreate(company,specialisation,workingConditions,payment,workingExperience,educa
tion,License,English);
        break;

    case 2:
        System.out.println(linkedContainer.size());
        linkedContainer.addLast(SecondCreate1);
        System.out.println(linkedContainer.size());

        for(SecondCreate tmp : linkedContainer)
        {
            System.out.println(tmp);
        }

        break;
    case 3:
        linkedContainer.clean();
        System.out.println(linkedContainer.size());
        break;
    case 4:
        Object []arr = linkedContainer.toArray().toArray();
        for(int i=0; i<linkedContainer.size();i++)
        {
            System.out.println(arr[i]);
        }
        break;
    case 5:
        ObjectOutputStream objectOutputStream = new
ObjectOutputStream(new FileOutputStream("store.txt"));
        objectOutputStream.writeObject(linkedContainer);
        objectOutputStream.close();

        break;
    case 6:
        /* ObjectInputStream objectInputStream = new ObjectInputStream(new
FileInputStream("store.txt"));
        linkedContainer<SecondCreate> newContainer =
(linkedContainer<SecondCreate>)objectInputStream.readObject();

        for (SecondCreate t : newContainer) {
            System.out.println(t);
        }*/

        break;
    case 7:
        /*XmlWrite xxmlWrite = new XmlWrite();

        xxmlWrite.write(linkedContainer,"XML.xml");*/
        break;
    case 8:
        /*linkedContainer<SecondCreate> newXml = XmlRead.read("XML.xml");

        for(SecondCreate t : newXml )
        {
            System.out.println(t);
        }*/

        break;

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

```

```

    }

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

        if(args[0].equals("-auto"))
        {

            System.out.println("U chose auto mode.");
            System.out.println("1. Reading from file");
            fileRead();

        }else
        {
            manual();
        }
    }
}

```

SecondCreate.java :

```

import java.io.Serializable;

public class SecondCreate implements java.lang.Comparable<SecondCreate>, Serializable
{

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

    public SecondCreate(String company,String specialisation,String
    workingConditions,int payment,int workingExperience,String education,String
    Licence,String English)
    {
        this.company=company;
        this.specialisation=specialisation;
        this.workingConditions=workingConditions;
        this.payment=payment;
        this.workingExperience=workingExperience;
        this.education=education;
        this.Licence=Licence;
        this.English=English;
    }

    public int getPayment()
    {
        return payment;
    }
}

```

```

    public String getSpecialisation()
    {
        return specialisation;
    }

    public String getEducation()
    {
        return education;
    }

    public String getCompany()
    {
        return company;
    }

    public String getWorkingConditions()
    {
        return workingConditions;
    }

    public int getWorkingExperience()
    {
        return workingExperience;
    }

    public String getLicence()
    {
        return Licence;
    }

    public String getEnglish()
    {
        return English;
    }

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

    @Override
    public int compareTo(SecondCreate o) {
        SecondCreate entry = (SecondCreate) o;

        int tmp = company.compareTo(entry.company);
        // this.payment - ((SecondCreate)o).payment;
        return tmp;
    }
}

```

iLinked.java:

```

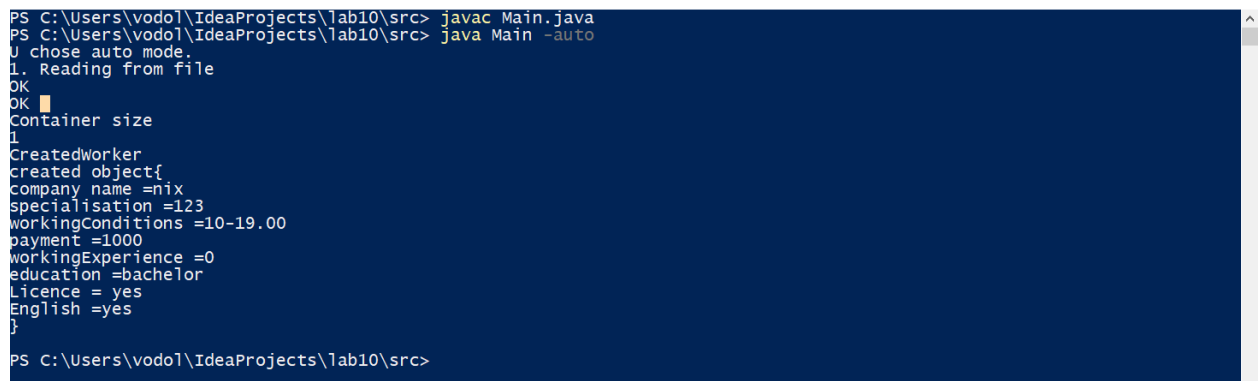
package ua.khpi.oop.vasilchenko09.MyList;

import java.io.Serializable;

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

```

ВАРІАНТИ ВИКОРИСТАННЯ



```

PS C:\Users\vodo1\IdeaProjects\lab10\src> javac Main.java
PS C:\Users\vodo1\IdeaProjects\lab10\src> java Main -auto
I chose auto mode.
1. Reading from file
OK
Container size
1
Created worker
created object{
company name =nix
specialisation =123
workingConditions =10-19.00
payment =1000
workingExperience =0
education =bachelor
Licence = yes
English =yes
}
PS C:\Users\vodo1\IdeaProjects\lab10\src>

```

Рис. 11.1 – Результат роботи програми

Після строки Reading from file йдуть дві строки з написом ОК це означає, що введені данні були перевірені і підпадають під дозволені.


```

PS C:\Users\vodo1\IdeaProjects\lab10\src> javac Main.java
PS C:\Users\vodo1\IdeaProjects\lab10\src> java Main -manual
I have chosen manual mode
Choose action
1. Create new element
2. Add elem
3. Clear container
4. Convert to Array
5. Serialize
6. Deserialize
7. Xml serialize
8. Xml deserialize
1
Enter company name
er
Enter specialisation
456
ok
Enter working Conditions
567
Enter payment
678
Enter working Experience
967
Enter education
5454
Enter knowledge of English
Enter driving licence
78
Choose action
1. Create new element
2. Add elem
3. Clear container
4. Convert to Array
5. Serialize
6. Deserialize
7. Xml serialize
8. Xml deserialize
2
0
1
created object{
company name =er
specialisation =456
workingConditions =567
payment =678

```

Рис. 11.2 – Результат роботи програми

Програму можна використовувати задля створення бази даних. Завдяки параметризації зв'язного списку, базу даних можна використати для будь-яких типів даних. До попереднього завдання була додана можливість перевірки введених користувачем даних.

ВИСНОВКИ

При виконанні лабораторної роботи набуто практичних навичок щодо розробки параметризованих класів. Завдяки цій можливості в JAVA, можливо створювати колекції та інші класи на основі будь-яких типів. Також навчився обробляти параметризовані контейнери, навчився перевіряти введені користувачем дані за допомогою регулярних виразів.