Тема: Колекції Java.
Мета: Ознайомлення з бібліотекою колекції Java SE. Використання колекції для розміщення об’єктів розроблених класів.

1 ВИМОГИ

1.1 Розробник

Інформація про розробника:
- Когутенко Олександр Олексійович;
- КІТ-119Д;
- 11 варіант.

1.2 Загальне завдання

1. Розробити консольну програму для реалізації завдання обробки даних згідно прикладної області.
2. Для розміщення та обробки даних використовувати контейнери (колекції) і алгоритми з Java Collections Framework.
3. Забезпечити обробку колекції об'єктів: додавання, видалення, пошук, сортування згідно розділу Прикладні задачі л.р. №10.
4. Передбачити можливість довготривалого зберігання даних: 1) за допомогою стандартної серіалізації; 2) не використовуючи протокол серіалізації.
5. Продемонструвати розроблену функціональність в діалоговому та автоматичному режимах за результатом обробки параметрів командного рядка.

2 ОПИС ПРОГРАМИ

2.1 Засоби ООП

Використовуеться наслідування, інтерфейс, поліморфізм.

2.2 Ієрархія та структура класів

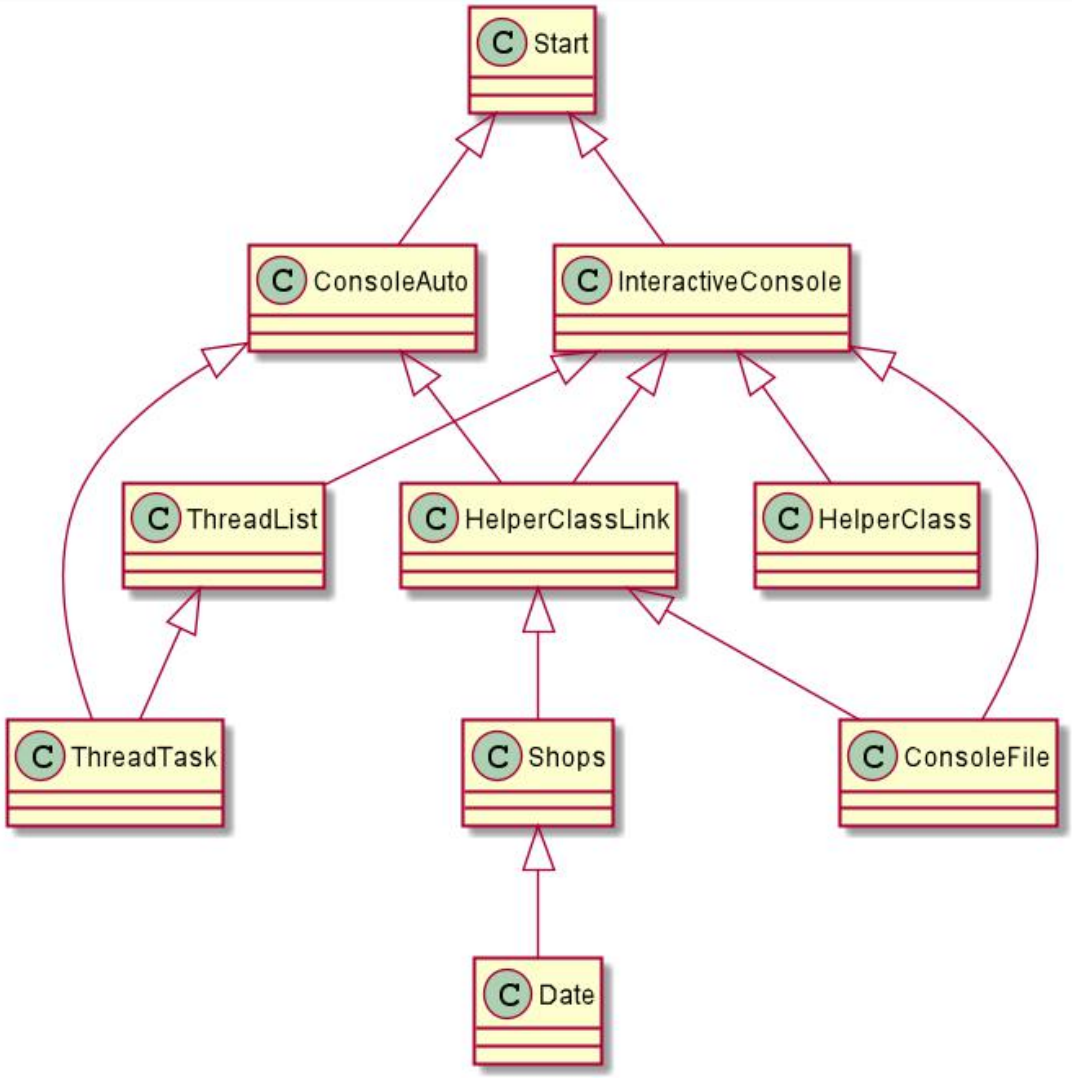

Рисунок 13.1 - ієрархія класів

Використовую 7 классів: InteractiveConsole, Start, Date, Shops, ConsoleFile, HelperClassLink, ConoleAuto, ThreadTask, ThreadList.

- InteractiveConsole клас для налагодженого спілкування програми з користувачем та методами нестандартних протоколів серіалізації.
- Start клас який має точку входу у програму.
- Date клас використовуется для збереження дати.
- Shops клас приклодної галузі.
- ConoleFile клас за допомогою якого користувач може наблюдати за відображенням вмісту каталогів.
- HelperClassLink використовуеться за для реалізації зв’язного списку з методами стандартних методів серіалізації.
- ConsoleAuto класс для автоматичної роботи із списком.
- ThreadTask деяки шаблон контейнеру для
- ThreadList класс який має в собі статичні класи з яких створюється потоки для кожної дії.

2.3 Важливі фрагменти програми

Я змінив свій список на LinkedList:

```
package ua.khpi.oop.kogutenko15;

import java.beans.XMLDecoder;
import java.beans.XMLEncoder;
import java.io.*;
import java.util.*;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

/**
 * The type Helper class link.
 *
 * @param <T> the type parameter
 */
public class HelperClassLink<T> {
    List<T> list = new LinkedList<T>();
    HelperClassLink(){  }
    HelperClassLink(Shops[] shops){
        int i = 0;
        for(T item : list) {
            list.add((T)shops[i++]);
        }
    }



    @Override
    public String toString() {
        StringBuilder s = new StringBuilder();
        for (T item : this.list)
            s.append(item + " ");
        return s.toString();
    }

    /**
     * Print list.
     */
    public synchronized void printList() {

        String str = "";
        for (T item : this.list) {
            str += item.toString();
        }
        System.out.println(str);
    }

    public synchronized void serialization(int idS, File file) {
        switch (idS){
            case 1 :
```

```
            serializationXML(file);
            break;
        case 2 :
            serializationBIN(file);
            break;
        case 3 :
            serializationTXT(file);
            break;

    }
}

private synchronized void serializationTXT(File file) {
    try (PrintWriter pw = new PrintWriter(file.getName())) {
        System.out.println("size :" + list.size());
        for (T el : this.list) {
            pw.write(el.toString());
            System.out.print(el.toString());
        }
        System.out.println("End serialization");
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    }
}

/**
 * Serialization xml.
 */
private synchronized void serializationXML(File file) {
    try {
        XMLEncoder encoder = new XMLEncoder(
                new BufferedOutputStream(
                        new FileOutputStream(file)));

        encoder.writeObject(this.list.size());

        for (T shop : this.list) {
            encoder.writeObject(shop);
        }

        encoder.close();
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}

/**
 * Serialization bin.
 */
private synchronized void serializationBIN(File file) {
```

```
    try {
        FileOutputStream fos = new FileOutputStream(file);
        ObjectOutputStream oos = new ObjectOutputStream(fos);

        oos.writeObject(this.list.size());
        System.out.println("size :" + this.list.size());
        for (T el : this.list) {
            oos.writeObject(el);
        }
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}


public void deserialization(int idD, File file){
    switch (idD){
        case 1 :
            deserializationXML(file);
            break;
        case 2 :
            deserializationBIN(file);
            break;
        case 3 :
            deserializationTXT(file);
            break;
    }
}

/**
 * Deserializtion bin.
 */
private synchronized void deserializationBIN(File file) {
    try {
        FileInputStream fis = new FileInputStream(file);///pathname
        ObjectInputStream ois = new ObjectInputStream(fis);
        Integer count = ois.readInt();
        for (int i = 0; i < count; i++) {
            this.list.add((T) ois.readObject());
        }

    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    } catch (ClassNotFoundException e) {
        e.printStackTrace();
    }
}
```

```
/**
 * Deserializtion xml.
 */
private synchronized void deserializationXML(File file) {
    try {
        XMLDecoder decoder = new XMLDecoder(
                new BufferedInputStream(
                        new FileInputStream(file)
                )
        );

        int count = (int) decoder.readObject();

        for (int i = 0; i < count; i++) {
            T shops = (T) decoder.readObject();
            this.list.add(shops);
        }
        decoder.close();

    } catch (FileNotFoundException e) {
        e.printStackTrace();
    }
}

private synchronized void deserializationTXT(File file) {
    try {
        BufferedReader br = new BufferedReader(new InputStreamReader(
                new FileInputStream(file)));
        String line, id = null,
                name = null,
                unit = null,
                count = null,
                price = null,
                date = null,
                description = null;
        Pattern p;
        Matcher m;
        while ((line = br.readLine()) != null) {
            if (line.contains("id:")) {
                //regex
                p = Pattern.compile("[0-9^\\s]+");
                id = line.substring(4, line.indexOf(" | name:")).trim();
                m = p.matcher(id);
                if (!m.matches()) {
                    id = "0";
                }
            }
            if (line.contains("name:")) {
                //regex
                p = Pattern.compile("^[\\w^\\s]{3,15}$");
                name = line.substring(line.indexOf("name: ") + 6, line.indexOf(" | unit:")).trim();
                m = p.matcher(name);
```

```
                if (!m.matches()) {
                    name = "prod";
                }
            }
            if (line.contains("unit:")) {
                //regex
                p = Pattern.compile("kg|l|kg/l");
                unit = line.substring(line.indexOf("unit:") + 6, line.indexOf(" | count: ")).trim();
                m = p.matcher(unit);
                if (!m.matches()) {
                    unit = "obj";
                }
            }
            if (line.contains("count:")) {
                //regex
                p = Pattern.compile("[0-9[^\\n\\t\\f\\r]]+");
                count = line.substring(line.indexOf("count:") + 7, line.indexOf(" | price: ")).trim();
                m = p.matcher(count);
                if (!m.matches()) {
                    count = "0";
                }
            }
            if (line.contains("price")) {
                //regex
                p = Pattern.compile("[0-9[^\\n\\t\\f\\r]]+");
                price = line.substring(line.indexOf("price:") + 7, line.indexOf(" | date: ")).trim();
                m = p.matcher(price);
                if (!m.matches()) {
                    price = "0";
                }
            }
            if (line.contains("date:")) {
                //regex
                p = Pattern.compile("^(?:(?:31(\\/|-|\\.)(?:0?[13578]|1[02]))\\1|(?:(?:29|30)(\\/|-
|\\.)(?:0?[1,3-9]|1[0-2])\\2))(?:(?:1[6-9]|[2-9]\\d)?\\d{2})$|^(?:29(\\/|-|\\.)0?2\\3(?:(?:(?:1[6-9]|[2-
9]\\d)?(?:0[48]|[2468][048]|[13579][26])|(?:(?:16|[2468][048]|[3579][26])00))))$|^(?:0?[1-
9]|1\\d|2[0-8])(\\/|-|\\.)(?:(?:0?[1-9])|(?:1[0-2]))\\4(?:(?:1[6-9]|[2-9]\\d)?\\d{2})$");
                date = line.substring(line.indexOf("date:") + 6, line.indexOf(" | description: "));
                m = p.matcher(date);
                if (!m.matches()) {
                    date = "01/01/2021";
                }
            }
            if (line.contains("description:")) {
                //regex
                description = line.substring(line.indexOf("description:") + 13, line.length() - 1);
            }

            Shops shop = new Shops();
            shop.setId(Integer.parseInt(id));
            shop.setCount(Integer.parseInt(count));
            shop.setName(name);
```

```
            shop.setDate(date);
            shop.setUnit(unit);
            shop.setPrice(Integer.parseInt(price));
            shop.setDescription(description);
            this.list.add((T)shop);
        }
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}

/**
 * To array t [ ].
 *
 * @return the t [ ]
 */
public synchronized T[] toArray() {
    T[] arr = (T[]) new Object[list.size()];
    for (int i = 0; i < list.size(); i++) arr[i] = list.get(i);
    return arr;
}

/**
 * From array.
 *
 * @param array the array
 * @return
 */
public synchronized List<T> fromArray(T[] array) {
    List<T> list = new LinkedList<>();
    for (int i = 0; i < array.length; i++) {
        list.add(array[i]);
    }
    return list;
}

public synchronized HelperClassLink<Shops> sort(HelperClassLink<Shops> list) {
    System.out.println("list before:\n");
    System.out.println("\n---------------------------------\n");
    list.printList();
    System.out.println("\n---------------------------------\n");
    Shops[] shops = list.toArray();
    Integer field;
    Scanner scanner = new Scanner(System.in);
    while (true) {
        System.out.print("Enter field sorted (1 - name; 2 - price; 3 - date\n>>>");
        Pattern p = Pattern.compile("[123]");
        field = scanner.nextInt();
        Matcher m = p.matcher(field.toString());
        if (m.matches()) {
```

```
            break;
        } else {
            System.out.println("Enter info correctly!!!");
        }
    }
    bubbleSort(shops, field);
    return new HelperClassLink<>(shops);
}

public Shops[] bubbleSort(Shops[] array, int field) {
    boolean sorted = false;
    while (!sorted) {
        sorted = true;
        for (int i = 1; i < array.length; i++) {
            if (compare(array[i], array[i - 1], field)) {
                swap(array, i, i - 1);
                sorted = false;
            }
        }
    }
    return array;
}

private boolean compare(Shops a, Shops b, int field) {
    switch (field) {
        case 1:
            return a.getName().compareTo(b.getName()) >= 0;
        case 2:
            return a.getPrice() < b.getPrice();
        case 3:
            return (a.getDate().getYear() > b.getDate().getYear())
                    || (a.getDate().getYear() == b.getDate().getYear() && a.getDate().getMonth() >
b.getDate().getMonth())
                    || (a.getDate().getYear() == b.getDate().getYear() && a.getDate().getMonth() ==
b.getDate().getMonth()
                    && a.getDate().getDay() > b.getDate().getDay());

    }
    return false;
}

private void swap(Shops[] array, int ind1, int ind2) {
    Shops tmp = array[ind1];
    array[ind1] = array[ind2];
    array[ind2] = tmp;
}

}
```

3 ВАРІАНТИ ВИКОРИСТАННЯ

Використання індентичне з використанням 14ої лабороторної роботи.

ВИСНОВКИ

Ознайомився з бібліотекою колекції Java SE. Використував колекції для розміщення об’єктів розроблених класів.