Тема: Обробка парметризованих контейнерів
Мета: Розширення функціональності параметризованих класів

1 ВИМОГИ

1.1 Розробник

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

1.2 Загальне завдання

1. Використовуючи програму рішення завдання лабораторної роботи №9:

2. Розробити параметризовані методи (Generic Methods) для обробки колекцій об'єктів згідно прикладної задачі.
3. 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.

4. Забороняється використання контейнерів (колекцій) з Java Collections Framework.

2 ОПИС ПРОГРАМИ

2.1 Засоби ООП

Використовуеться наслідування, інтерфейс, поліморфізм.

2.2 Ієрархія та структура класів

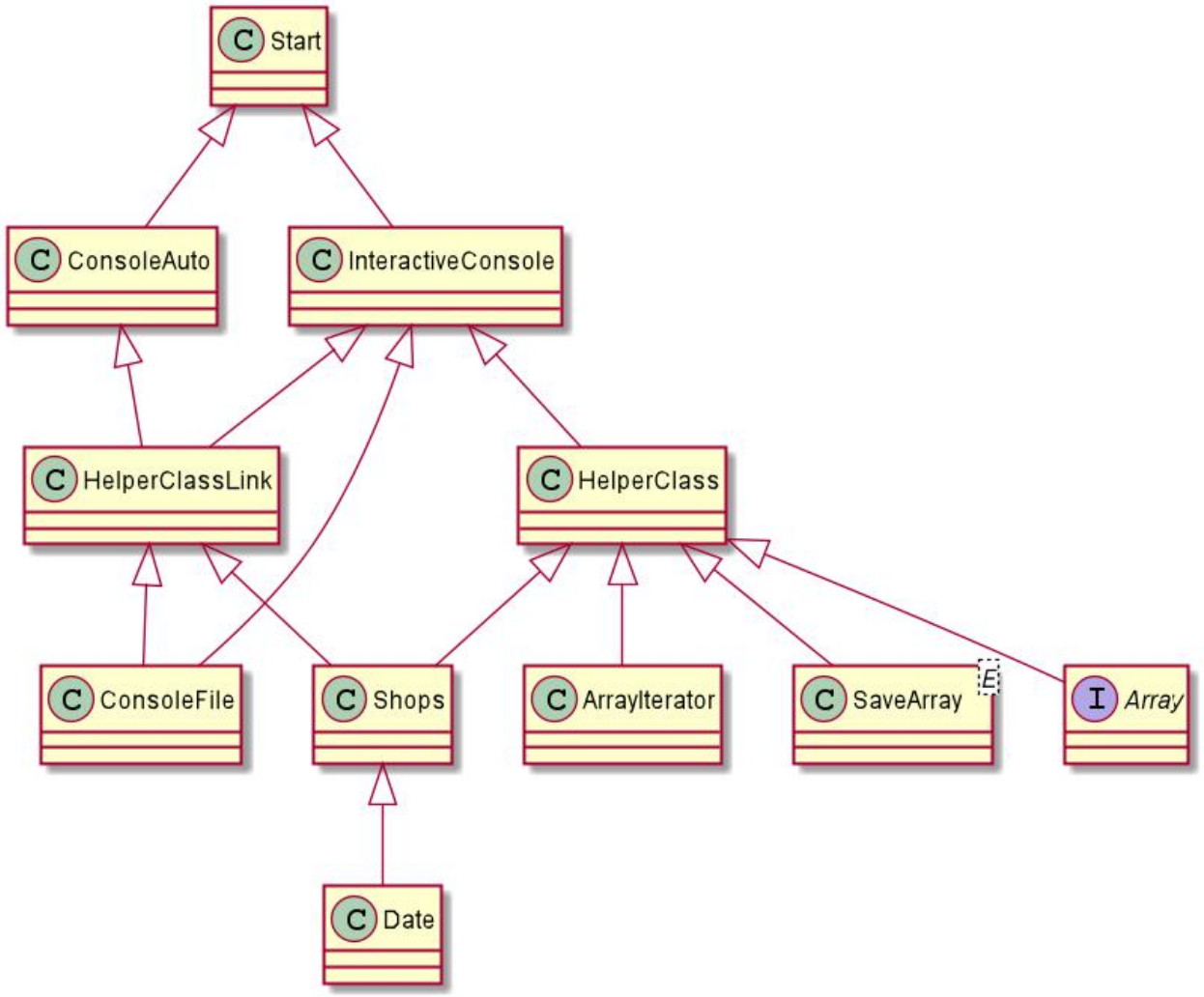

Рисунок 10.1 - иерархія класів

Використовую 11 классів: Array, ArrayIterator, HelperClass, InteractiveConsole, Start, SaveArray, Date, Shops, ConsoleFile, HelperClassLinkб, ConsoleAuto.

- Array використовую як інтерфейс для класу контейнеру.
- ArrayIterator використовую як особисту реалізацію ітератора.
- HelperClass допоміжний класс для розрахунків.
- InteractiveConsole клас для налагодженого спілкування програми з користувачем та методами нестандартних протоколів серіалізації.
- Start клас який має точку входу у програму.
- SaveArray клас контейнер який має все необхідні методи маніпулятори.
- 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.

2.3 Важливі фрагменти програми

```
package ua.khpi.oop.kogutenko10;

import java.io.*;
import java.util.HashMap;
import java.util.Random;
import java.util.Scanner;

/**
 * The type Console auto.
 */
public class ConsoleAuto {
    private HelperClassLink<Shops> hlAuto = new HelperClassLink<>();
    /**
     * The Scanner.
     */
    Scanner scanner = new Scanner(System.in);

    /**
     * Start console.
     */
    public void startConsole() {
        System.out.println("Start auto...");
        try {
            deserializationTXT();
            System.out.println("Beginner container:\n" +
                    "----------------------------------------------------------------------------------------
----------------------------------");
            hlAuto.printList();
            System.out.println("----------------------------------------------------------------------
-------------------------------------------------");
            Shops newShop = new Shops(9,
                            "prod_9",
                             "1",
                            130,
                             133,
                                new Date(17,2,2021),
                                new HashMap<String,
String>(){{put("color","new_color");put("weight","10");}});
            System.out.println("new elem is\n" + newShop.toString());
            hlAuto.add(newShop);
            System.out.println("Container after adding\n----------------------------------
-------------------------------------------------------------------------");
            hlAuto.printList();
```

```
            System.out.println("----------------------------------------------------------------------
-------------------------------------------------");
            System.out.println("remove the most expansive in price");
            hlAuto = sort(hlAuto);
            System.out.println("Container after sorting\n----------------------------------
-------------------------------------------------------------------------------");
            hlAuto.printList();
            System.out.println("----------------------------------------------------------------------
-------------------------------------------------");
            hlAuto.remove(hlAuto.size() - 1);
            System.out.println("Container after removing\n---------------------------------
-------------------------------------------------------------------------------");
            hlAuto.printList();
            System.out.println("----------------------------------------------------------------------
-------------------------------------------------");
            serializationTXT();
        }
        finally {
            System.out.println("End auto...");
        }
    }

    private void serializationTXT(){
        File file = new File("D:\\eclips-workspace\\kogutenko-
oleksandr\\src\\ua\\khpi\\oop\\txt10-" + new Random().nextInt() % 20 +
".txt");///pathname
        try (PrintWriter pw = new PrintWriter(new FileOutputStream(file))) {
            System.out.println("size :" + hlAuto.size());
            for (Shops el : hlAuto)
            {
                pw.write(el.toString());
                System.out.print(el.toString());
            }

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

    /**
     * Deserializtion txt.
     */
    private void deserializationTXT(){
        File file = new File("D:\\eclips-workspace\\kogutenko-
oleksandr\\src\\ua\\khpi\\oop\\txt10.txt");///pathname
```

```
        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;
            while ((line = br.readLine()) != null) {
                if (line.contains("id:")) {
                    id = line.substring(4, line.indexOf(" | name:"));
                }
                if (line.contains("name:")) {
                    name = line.substring(line.indexOf("name: ") + 6, line.indexOf(" |
unit:"));
                }
                if (line.contains("unit:")) {
                    unit = line.substring(line.indexOf("unit:") + 6, line.indexOf(" | count:
"));
                }
                if(line.contains("count:")){
                    count = line.substring(line.indexOf("count:") + 7, line.indexOf(" |
price: "));
                }
                if(line. contains("price")){
                    price = line.substring(line.indexOf("price:") + 7, line.indexOf(" | date:
"));
                }
                if(line.contains("date:")){
                    date = line.substring(line.indexOf("date:") + 6, line.indexOf(" |
description: "));
                }
                if(line.contains("description:")){
                    description = line.substring(line.indexOf("description:") + 13,
line.length() - 1);
                }

                Shops shop = new Shops();
                shop.setId(Integer.parseInt(id.trim()));
                shop.setCount(Integer.parseInt(count.trim()));
                shop.setName(name);
                shop.setDate(date);
                shop.setUnit(unit);
```

```
            shop.setPrice(Integer.parseInt(price.trim()));
            shop.setDescription(description);
            hlAuto.add(shop);
        }
    }
    catch(FileNotFoundException e) {e.printStackTrace();}
    catch (IOException e) {e.printStackTrace();    }
    //catch (ClassNotFoundException e) {e.printStackTrace();}
}

/**
 * Sort helper class link.
 *
 * @param list the list
 * @return the helper class link
 */
public HelperClassLink<Shops> sort(HelperClassLink<Shops> list) {
    System.out.println("Function sort\nlist before:\n");
    System.out.println("\n---------------------------------\n");
    list.printList();
    System.out.println("\n---------------------------------\n");
    Shops[] shops = new Shops[list.size()];
    for (int i = 0; i < shops.length; i++) {
        shops[i] = list.get(i);
    }
    bubbleSort(shops, 2);
    return new HelperClassLink<>(shops);
}

private void 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;
            }
        }
    }
}

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

У цій програмі добавлено лише автоматичну обробку тому демонструю лише ЇЇ:

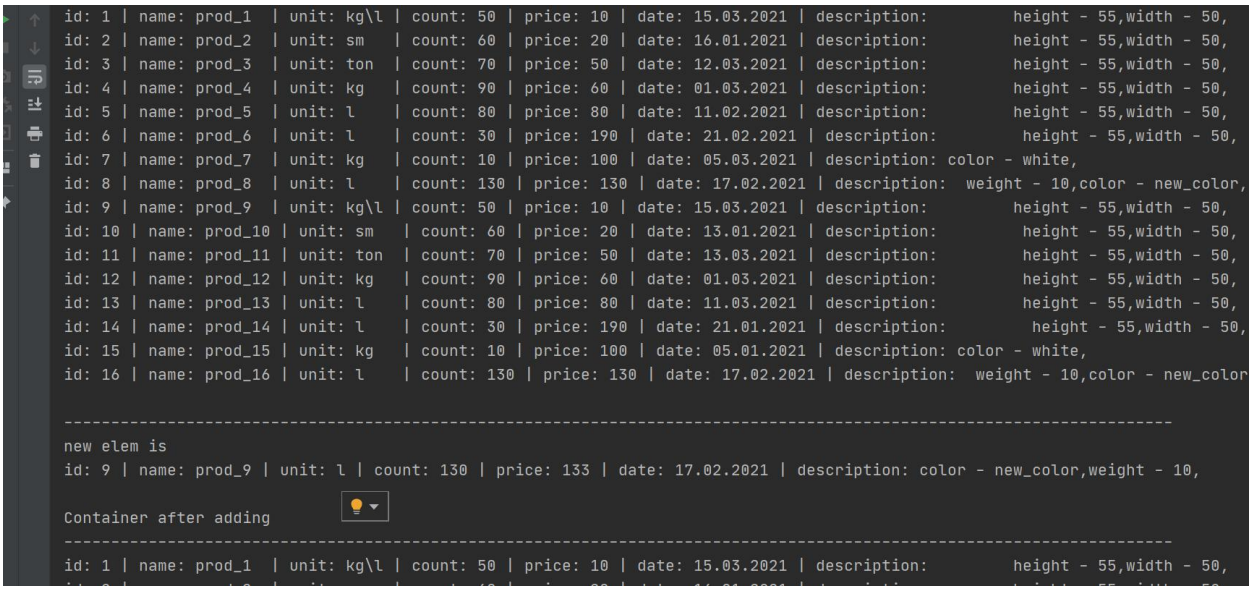

Рисунок 10.2 - автоматична обробка

ВИСНОВКИ

Розширив функціональність параметризованих класів у 9ій лабороторній роботі.