Розробник:

Огонькова Наталія Максимівна

Група КІТ-118б

Варіант №11

**Лабораторна робота № 15**

Колекції в Java

**Мета:**

* Ознайомлення з бібліотекою колекцій Java SE.
* Використання колекцій для розміщення об'єктів розроблених класів.

**Вимоги:**

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.
6. ОПИС ПРОГРАМИ
   1. Опис змінних

**private** String name;

**private** String unit;

**private** **int** count;

**private** **int** unit\_price;

**private** String data\_of\_receipt;

**private** String attribute;

**private** String value;

* 1. Ієрархія та структура класів

**class** Lab15 – точка входу в програму

**class** Demo – точка входу в програму у двох режимах

**class** UI – клас с описом двох режимів входу в програму

**class** Until – клас с описом параметризованих методів

**class** Container – клас, що містить функції

**class** Store – клас, що містить дані, введені користувачем

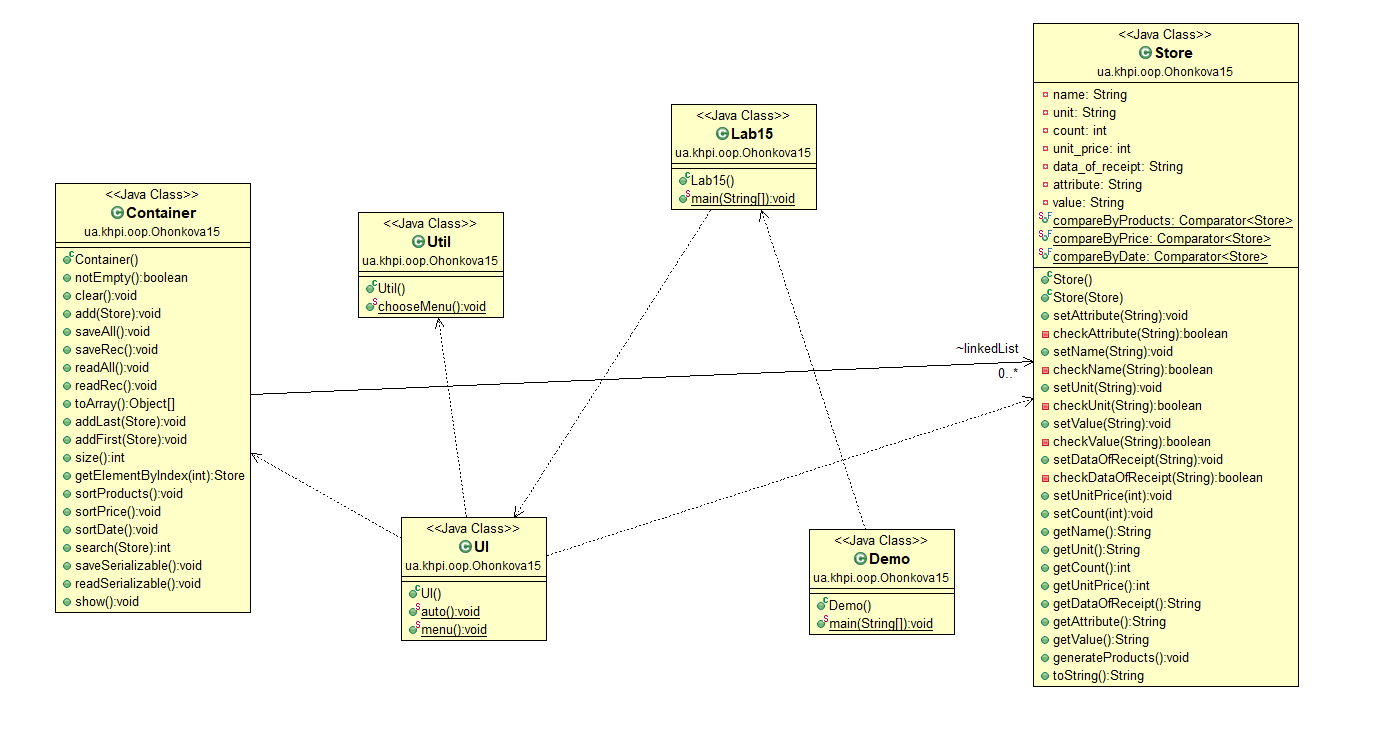


Рисунок 1 – Класи програми та їх зв’язки

* 1. Текст програми

**class** Lab15

package ua.khpi.oop.Ohonkova15;

import java.io.IOException;

public class Lab15 {

public static void main(String[] args) throws IOException {

if (args.length != 0) {

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

UI.*auto*();

} else {

System.*out*.println("Repeat entered with params -auto");

}

} else {

UI.*menu*();

}

}

}

**class** Demo

**package** ua.khpi.oop.Ohonkova15;

**import** java.io.IOException;

**public** **class** Demo {**public** **static** **void** main(String args[]) **throws** IOException {

Lab15.*main*(**new** String[]{"-auto"});

//Lab15.main(args);

}

}

**class** UI

package ua.khpi.oop.Ohonkova15;

import java.io.IOException;

import java.util.Scanner;

import java.util.InputMismatchException;

import ua.khpi.oop.Ohonkova15.Container;

import ua.khpi.oop.Ohonkova15.Store;

import ua.khpi.oop.Ohonkova15.Util;

public class UI {

public static void auto() throws IOException {

Store temp = new Store();

Container container = new Container();

System.out.println(container.size());

container.add(temp);

System.out.println(container.size());

container.readRec();

System.out.println("==================");

container.show();

System.out.println("==================");

System.out.println(container.size());

container.saveSerializable();

System.out.println("==================");

System.out.println(container.getElementByIndex(2));

container.clear();

System.out.println(container.size());

}

public static void menu() throws IOException {

System.out.println("It is menu mode");

Container linkedContainer = new Container();

Store rec1 = new Store();

Scanner scan = new Scanner(System.in);

boolean loop = true;

while (loop) {

Util.chooseMenu();

int choose = scan.nextInt();

switch (choose) {

case 1:

rec1.generateProducts();

linkedContainer.add(rec1);

System.out.println("Done!");

break;

case 2:

if (linkedContainer.notEmpty()) {

linkedContainer.show();

} else {

System.out.println("List is empty!");

}

break;

case 3:

if (linkedContainer.notEmpty()) {

linkedContainer.clear();

System.out.println("Successfully!");

} else {

System.out.println("Array is empty!");

}

break;

case 4:

if (linkedContainer.notEmpty()) {

System.out.println("Your container have data.");

} else {

System.out.println("Your container doesn`t have data.");

}

break;

case 5:

System.out.println("Size of container: " + linkedContainer.size());

break;

case 6:

if (linkedContainer.notEmpty()) {

System.out.print("Entered index from 0 to " + (linkedContainer.size() - 1) + ": ");

int choose1 = scan.nextInt();

if (choose1 > (linkedContainer.size() - 1) || choose1 < 0) {

System.out.println("Repeat enter");

} else {

System.out.println(linkedContainer.getElementByIndex(choose1));

}

break;

}

case 7:

if (linkedContainer.notEmpty()) {

linkedContainer.saveRec();

} else {

System.out.println("Your container is empty");

}

break;

case 8:

if (linkedContainer.notEmpty()) {

linkedContainer.clear();

}

linkedContainer.readRec();

break;

case 9:

if (linkedContainer.notEmpty()) {

System.out.println("What field do you want to sort by?");

System.out.println("1. Sort by products");

System.out.println("2. Sort by price");

System.out.println("3. Sort by date");

int choose2 = scan.nextInt();

boolean loop2 = true;

while (loop2) {

switch (choose2) {

case 1:

linkedContainer.sortProducts();

loop2 = false;

break;

case 2:

linkedContainer.sortPrice();

loop2 = false;

break;

case 3:

linkedContainer.sortDate();

loop2 = false;

break;

default:

System.out.println("Wrong number!");

break;

}

}

} else {

System.out.println("Your container is empty");

}

break;

case 0:

System.out.println("Thanks for working!");

loop = false;

break;

default:

System.out.println("Wrong number!");

break;

}

}

}

}

**class** Until

**package** ua.khpi.oop.Ohonkova15;

**public** **class** Util

{

**public** **static** **void** chooseMenu() {

System.***out***.println();

System.***out***.println("1. Add product.");

System.***out***.println("2. Show all products.");

System.***out***.println("3. Clear container.");

System.***out***.println("4. Check elements in container.");

System.***out***.println("5. Size of container.");

System.***out***.println("6. Get element by index.");

System.***out***.println("7. Save data to file.");

System.***out***.println("8. Read data from file.");

System.***out***.println("9. Sorting data in container.");

System.***out***.println("0. End of work.");

System.***out***.print("Write your choose there: ");

}

}

**class** Container

package ua.khpi.oop.Ohonkova15;

import ua.khpi.oop.Ohonkova15.Store;

import java.io.BufferedReader;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.FileReader;

import java.io.IOException;

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

import java.io.PrintWriter;

import java.io.Serializable;

import java.util.LinkedList;

public class Container implements Serializable {

LinkedList<Store> linkedList = new LinkedList<>();

public boolean notEmpty() {

return linkedList.size() > 0;

}

public void clear() {

linkedList.clear();

}

public void add(final Store prod) {

addLast(prod);

}

public void saveAll() {

try {

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

if (!file.exists()) {

file.createNewFile();

}

PrintWriter pw = new PrintWriter(file);

System.out.println();

pw.println(linkedList.size());

for (int i = 0; i < linkedList.size(); i++) {

pw.println(getElementByIndex(i));

}

pw.close();

} catch (IOException e) {

System.out.println("Error" + e);

}

}

public void saveRec() {

try {

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

if (!file.exists()) {

file.createNewFile();

}

PrintWriter pw = new PrintWriter(file);

Store temp;

System.out.println();

pw.println(size());

for (int i = 0; i < linkedList.size(); i++) {

temp = getElementByIndex(i);

pw.println(temp.getName());

pw.println(temp.getUnit());

pw.println(temp.getCount());

pw.println(temp.getUnitPrice());

pw.println(temp.getDataOfReceipt());

pw.println(temp.getAttribute());

pw.println(temp.getValue());

}

pw.close();

} catch (IOException e) {

System.out.println("Error" + e);

}

}

public void readAll() {

try (BufferedReader br = new BufferedReader(new FileReader("1.txt"))) {

Object temp;

String line;

line = br.readLine();

int count = Integer.parseInt(line);

for (int i = 0; i < count; i++) {

line = br.readLine();

temp = line;

add((Store) temp);

}

} catch (IOException ex) {

ex.printStackTrace();

}

}

public void readRec() throws IOException {

BufferedReader br = null;

Store temp = new Store();

try {

br = new BufferedReader(new FileReader("1.txt"));

String line;

line = br.readLine();

int count = Integer.parseInt(line);

for (int i = 0; i < count; i++) {

line = br.readLine();

temp.setName(line);

line = br.readLine();

temp.setUnit(line);

line = br.readLine();

temp.setCount(Integer.parseInt(line));

line = br.readLine();

temp.setUnitPrice(Integer.parseInt(line));

line = br.readLine();

temp.setDataOfReceipt(line);

line = br.readLine();

temp.setAttribute(line);

line = br.readLine();

temp.setValue(line);

add(new Store(temp));

}

} catch (IOException ex) {

ex.printStackTrace();

} finally {

br.close();

}

}

public Object[] toArray() {

return linkedList.toArray();

}

public void addLast(final Store obj) {

linkedList.addLast(obj);

}

public void addFirst(final Store obj) {

linkedList.addFirst(obj);

}

public int size() {

return linkedList.size();

}

public Store getElementByIndex(final int index) {

return linkedList.get(index);

}

public void sortProducts() {

linkedList.sort(Store.compareByProducts);

}

public void sortPrice() {

linkedList.sort(Store.compareByPrice);

}

public void sortDate() {

linkedList.sort(Store.compareByDate);

}

public int search(Store obj) {

int index = 0;

for (Store prod : linkedList) {

if (prod == obj) {

return index;

}

}

return -1;

}

public void saveSerializable() {

try {

ObjectOutputStream oo = new ObjectOutputStream(new FileOutputStream("save.ser"));

oo.writeObject(linkedList);

oo.close();

} catch (IOException ioe) {

System.out.println("Error");

}

}

public void readSerializable() {

try {

FileInputStream fileInput = new FileInputStream("save.ser");

ObjectInputStream objectInput = new ObjectInputStream(fileInput);

LinkedList<Store> temp ;

temp = (LinkedList<Store>) objectInput.readObject();

System.out.println(temp);

objectInput.close();

} catch (Exception exc) {

System.out.println("Error");

}

}

public void show() {

int count = 0;

for (Store s : linkedList) {

System.out.println("#" + (++count));

System.out.println(s.toString());

}

}

}

**class** Store

package ua.khpi.oop.Ohonkova15;

import java.io.Serializable;

import java.util.Comparator;

import java.util.InputMismatchException;

import java.util.Scanner;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

import ua.khpi.oop.Ohonkova15.Store;

public class Store implements Serializable{

private String name;

private String unit;

private int count;

private int unit\_price;

private String data\_of\_receipt;

private String attribute;

private String value;

public Store() {

name = "lemon";

unit = "kg";

count = 50;

unit\_price = 95;

data\_of\_receipt = "20.03.2020";

attribute = "tasty";

value = "fruits";

}

public Store (final Store obj) {

name = obj.name;

unit = obj.unit;

count = obj.count;

unit\_price = obj.unit\_price;

data\_of\_receipt = obj.data\_of\_receipt;

attribute = obj.attribute;

value = obj.value;

}

public void setAttribute(String attribute) {

if (checkAttribute(attribute)) {

this.attribute = attribute;

} else {

throw new InputMismatchException();

}

}

private boolean checkAttribute(final String attribute) {

Pattern pattern = Pattern.compile("[\\s\\w%$+#@^()=!\_\\\\-]\*", Pattern.CASE\_INSENSITIVE);

Matcher matcher = pattern.matcher(attribute);

return matcher.matches();

}

public void setName(String name) {

if (checkName(name)) {

this.name = name;

} else {

throw new InputMismatchException();

}

}

private boolean checkName(final String name) {

Pattern pattern = Pattern.compile("[a-z]\*-?\\s?", Pattern.CASE\_INSENSITIVE);

Matcher matcher = pattern.matcher(name);

return matcher.matches();

}

public void setUnit(String unit) {

if (checkUnit(unit)) {

this.unit = unit;

} else {

throw new InputMismatchException();

}

}

private boolean checkUnit(final String unit) {

Pattern pattern = Pattern.compile("[\\s\\w%$+#@^()=!\_\\\\-]\*", Pattern.CASE\_INSENSITIVE);

Matcher matcher = pattern.matcher(unit);

return matcher.matches();

}

public void setValue(String value) {

if (checkValue(value)) {

this.value = value;

} else {

throw new InputMismatchException();

}

}

private boolean checkValue(final String value) {

Pattern pattern = Pattern.compile("[a-z]\*-?\\s?", Pattern.CASE\_INSENSITIVE);

Matcher matcher = pattern.matcher(value);

return matcher.matches();

}

public void setDataOfReceipt(String data\_of\_receipt) {

if (checkDataOfReceipt(data\_of\_receipt)) {

this.data\_of\_receipt = data\_of\_receipt;

} else {

throw new InputMismatchException();

}

}

private boolean checkDataOfReceipt(final String data\_of\_receipt) {

Pattern pattern = Pattern.compile("[0-9]{2}.[0-9]{2}.[0-9]{4}");

Matcher matcher = pattern.matcher(data\_of\_receipt);

return matcher.matches();

}

public void setUnitPrice(int unit\_price) {

this.unit\_price = unit\_price;

}

public void setCount(int count) {

this.count = count;

}

public String getName() {

return name;

}

public String getUnit() {

return unit;

}

public int getCount() {

return count;

}

public int getUnitPrice() {

return unit\_price;

}

public String getDataOfReceipt() {

return data\_of\_receipt;

}

public String getAttribute() {

return attribute;

}

public String getValue() {

return value;

}

public void generateProducts() {

Scanner scan = new Scanner(System.in);

Scanner scan2 = new Scanner(System.in);

int choose = 0;

System.out.print("Введите название товара: ");

name = scan2.nextLine();

System.out.print("Введите единицу измирения: ");

unit = scan2.nextLine();

System.out.print("Введите количество товара: ");

count = scan.nextInt();

System.out.print("Введите цену за единицу товара: ");

unit\_price = scan.nextInt();

System.out.print("Введите дату получения: ");

data\_of\_receipt = scan2.nextLine();

System.out.print("Свойства: ");

attribute = scan2.nextLine();

System.out.print("Значение: ");

value = scan2.nextLine();

}

@Override

public String toString() {

return "Name = " + name + "\n" +"Unit = " + unit +"\n" + "Count = "

+ count + "\n" +"Unit\_price = " + unit\_price + "\n" +"Data\_of\_receipt = " + data\_of\_receipt + "\n" + "Attribute = "

+ attribute + "\n" +"Value = " + value;

}

public static final Comparator<Store> compareByProducts = Comparator.comparing(Store::getName);

public static final Comparator<Store> compareByPrice = Comparator.comparingInt(Store::getUnitPrice);

public static final Comparator<Store> compareByDate = Comparator.comparing(Store::getDataOfReceipt);

}

1. ВАРІАНТИ ВИКОРИСТАННЯ

Програму можна використовувати для створення контейнеру з даними, що вводить користувач, стосовно теми «магазин».

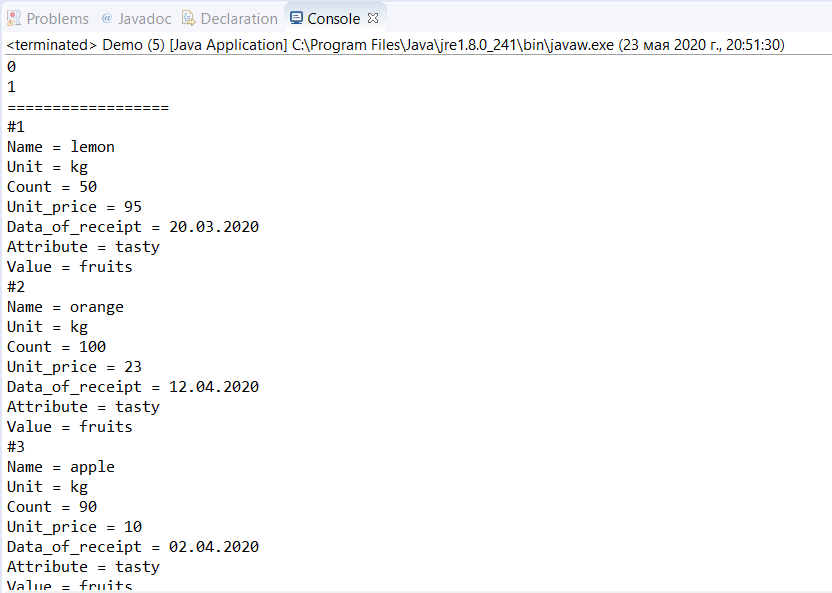


Рисунок 2 – Результат роботи у автоматичному режимі

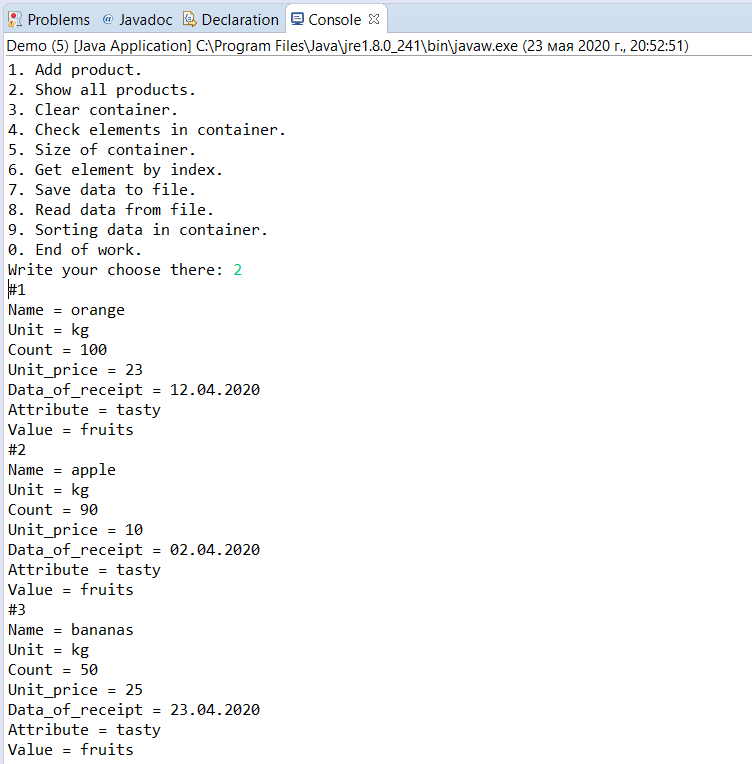


Рисунок 3 – Результат роботи у діалоговому режимі

ВИСНОВОК

При виконанні лабораторної роботи набуто практичних навичок щодо роботи з бібліотекою колекцій Java SE для розміщення об'єктів розроблених класів. Програма працює як в автоматичному режимі, так і в діалоговому. Були використані методи класу LinkedList. Програма виконується без помилок.