Розробник:

Огонькова Наталія Максимівна

Група КІТ 101.8б

Варіант №11

**Лабораторна робота № 8**

Основи введення/виведення Java SE

**Мета**:

* Оволодіння навичками управління введенням/виведенням даних з використанням класів платформи Java SE.

**Вимоги:**

1. Забезпечити можливість збереження і відновлення масива об’єктів рішення завдання лабораторної роботи №7.
2. Забороняється використання стандартного протокола серіалізації.
3. Продемонструвати використання моделі Long Term Persistence.
4. Забезпечити діалог з користувачем у вигляді простого текстового меню.
5. При збереженні та відновленні даних забезпечити діалоговий режим вибору директорії з відображенням вмісту і можливістю переміщення по підкаталогах.
6. ОПИС ПРОГРАМИ
   1. Опис змінних

Store rec1 = **new** Store(); //перший елемент класу Store

Store rec2 = **new** Store(); //другий елемент класу Store

Container contains = **new** Container(); //змінна для контейнера

1.2 Ієрархія та структура класів

**class** Lab08 – точка входу в програму

**class** Container – клас, в якому містяться контейнери

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

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

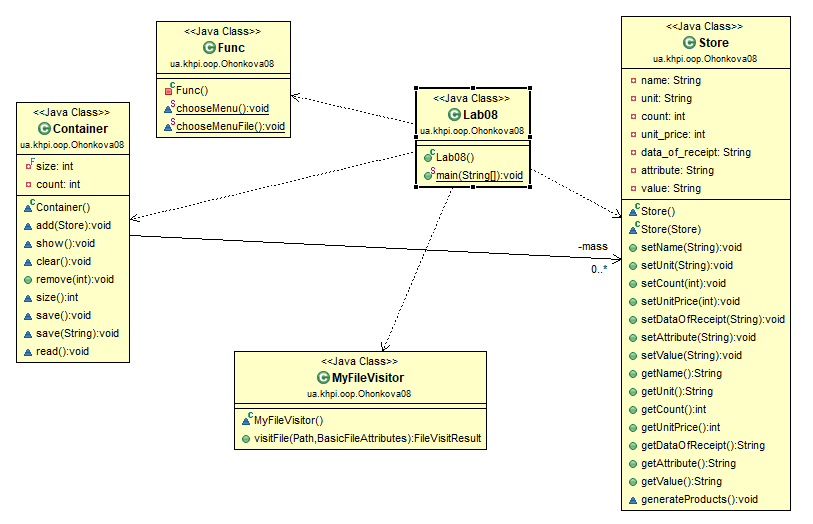


Рисунок 1 – Класи програми

* 1. Текст програми

**class** Lab08

package ua.khpi.oop.Ohonkova08;

import org.w3c.dom.Document;

import org.w3c.dom.Node;

import org.w3c.dom.NodeList;

import org.xml.sax.SAXException;

import javax.xml.parsers.DocumentBuilder;

import javax.xml.parsers.DocumentBuilderFactory;

import javax.xml.parsers.ParserConfigurationException;

import java.io.IOException;//выброс ошибок если что-то не получилось

import java.nio.file.\*;//для классов файл

import java.util.HashSet;

import java.util.Scanner;

public class Lab08 {

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

Container contains = new Container();

Store rec1 = new Store();

Scanner scan = new Scanner(System.in);

String str = "";

String str3 = "";

boolean loop = true;

while (loop) {

Func.chooseMenu();

int choose = scan.nextInt();

switch (choose) {

case 1:

rec1.generateProducts();

contains.add(rec1);

System.out.println("Успешно!");

break;

case 2:

if (contains.size() == 0) {

System.out.println("Ошибка! Массив пустой!");

break;

} else {

contains.show();

break;

}

case 3:

if (contains.size() == 0) {

System.out.println("Ошибка! Массив пустой!");

break;

} else {

System.out.print("Введите позицию удаления от 1 до " + contains.size());

int index = scan.nextInt();

if (index > contains.size() || index < 1) {

System.out.println("Повторите ввод: ");

} else {

contains.remove(index);

System.out.println("Успешно!");

}

}

break;

case 4:

if (contains.size() == 0) {

System.out.println("Ошибка! Массив пустой!");

break;

} else {

contains.clear();

System.out.println("Успешно!");

break;

}

case 5:

DocumentBuilderFactory factory = DocumentBuilderFactory.newDefaultInstance();

try {

DocumentBuilder builder = factory.newDocumentBuilder();

Document document = builder.parse("test.xml");

Node root = document.getDocumentElement();

System.out.println("Список кандидатов: ");

System.out.println();

NodeList vacancies = root.getChildNodes();

for (int i = 0; i < vacancies.getLength(); i++) {

Node vacancy = vacancies.item(i);

if (vacancy.getNodeType() != Node.TEXT\_NODE) {

NodeList bookProps = vacancy.getChildNodes();

for (int j = 0; j < bookProps.getLength(); j++) {

Node bookProp = bookProps.item(j);

if (bookProp.getNodeType() != Node.TEXT\_NODE) {

System.out.println(bookProp.getNodeName() + ":" + bookProp.getChildNodes().item(0).getTextContent());

}

}

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

}

}

} catch (ParserConfigurationException | SAXException | IOException ex) {

ex.printStackTrace(System.out);

}

break;

case 6:

boolean loop2 = true;

while (loop2) {

Func.chooseMenuFile();

int choose2 = scan.nextInt();

switch (choose2) {

case 1:

contains.save();

System.out.println("Успешно!");

break;

case 2:

Files.walkFileTree(Paths.get("."), new HashSet<FileVisitOption>(), 1, new MyFileVisitor());

break;

case 3:

Path path = Paths.get("save.txt").toAbsolutePath();

System.out.println(path.getParent());

break;

case 4:

boolean loop5 = true;

while (loop5) {

int cho = 0;

System.out.println("Введите путь к директории: ");

str = scan.next();

Files.walkFileTree(Paths.get(str), new HashSet<FileVisitOption>(), 1, new MyFileVisitor());

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

System.out.println("Вы желаете сохранить файл в эту директорию?");

System.out.println("1. - Да. 0. - Нет.");

while (true) {

cho = scan.nextInt();

if (cho == 1) {

str3 = str.concat("\\save.txt");

contains.save(str3);

loop5 = false;

break;

} else if (cho == 0) {

break;

} else {

System.out.println("Ошибка повторите ввод: ");

}

}

}

break;

case 5:

System.out.println("Вы вводили собственный путь директории?");

System.out.println("1 - Да. 0 - Нет.");

boolean loop4 = true;

while (loop4) {

int chosik = scan.nextInt();

if (chosik == 1) {

System.out.println("Введите новое имя файла: ");

String str1 = scan.next();

String str4 = str;

str4 = str4.concat("\\");

str4 = str4.concat(str1);

Files.move(Paths.get(str3), Paths.get(str4), StandardCopyOption.REPLACE\_EXISTING);

System.out.println("Успешно!");

loop4 = false;

} else if (chosik == 0) {

System.out.println("Выполните пункт 4");

loop4 = false;

} else {

System.out.println("Ошибка! Повторите ввод: ");

}

}

break;

case 6:

contains.read();

System.out.println("Успешно!");

break;

// case 7:

// XMLEncoder encoder = new XMLEncoder(new BufferedOutputStream(new FileOutputStream("trunk.xml")));

// encoder.writeObject(contains);

// encoder.close();

// break;

// case 8:

// XMLDecoder decoder = new XMLDecoder(new BufferedInputStream(new FileInputStream("trunk.xml")));

// Kontainer contic = (Kontainer) decoder.readObject();

// contic.show();

// decoder.close();

case 0:

System.out.println("Выход!");

loop2 = false;

break;

default:

System.out.println("Ошибка. Неверное число!");

break;

}

}

break;

case 0:

System.out.println("Спасибо за работу!");

loop = false;

break;

default:

System.out.println("Ошибка. Неверное число!");

break;

}

}

}

}

**class** Container

package ua.khpi.oop.Ohonkova08;

import java.io.BufferedReader;

import java.io.File;

import java.io.FileReader;

import java.io.IOException;

import java.io.PrintWriter;

import java.io.Serializable;

import java.util.Arrays;

import ua.khpi.oop.Ohonkova08.Store;

class Container implements Serializable {

/\*\*

\* First size for array.

\*/

private final int size = 10;

/\*\*

\* Array contains all data.

\*/

private int count = 0;

private Store[] mass = new Store[size];

void add(final Store temp){

Store buff = new Store(temp);

if (count == mass.length) {

mass = Arrays.copyOf(mass, mass.length \* 2);

mass[count++] = temp;

} else {

mass[count++] = temp;

}

}

void show() {

if (count == 0) {

System.out.println("Ошибка! Массив пустой!");

} else {

System.out.println();

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

System.out.println("Товар №" + (i+1));

System.out.println("Название: " + mass[i].getName());

System.out.println("Единица измерения: " + mass[i].getUnit());

System.out.println("Количество: " + mass[i].getCount());

System.out.println("Цена за единицу: " + mass[i].getUnitPrice());

System.out.println("Дата получения: " + mass[i].getDataOfReceipt());

System.out.println("Свойства: " + mass[i].getAttribute());

System.out.println("Значение: " + mass[i].getValue());

System.out.println("\n");

}

}

}

void clear() {

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

mass[i] = null;

}

count = 0;

}

public void remove(int index) {

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

mass[index++] = mass[i + 1];

}

count--;

}

int size() {

return count;

}

void save() {

try {

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

if (!file.exists()) {

file.createNewFile();

}

PrintWriter pw = new PrintWriter(file);

System.out.println();

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

pw.println(count);

pw.println(mass[i].getName());

pw.println(mass[i].getUnit());

pw.println(mass[i].getCount());

pw.println(mass[i].getUnitPrice());

pw.println(mass[i].getDataOfReceipt());

pw.println(mass[i].getAttribute());

pw.println(mass[i].getValue());

}

pw.close();

} catch (IOException e) {

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

}

}

void save(final String str) {

try {

File file = new File(str);

if (!file.exists()) {

file.createNewFile();

}

PrintWriter pw = new PrintWriter(file);

System.out.println();

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

pw.println(count);

pw.println(mass[i].getName());

pw.println(mass[i].getUnit());

pw.println(mass[i].getCount());

pw.println(mass[i].getUnitPrice());

pw.println(mass[i].getDataOfReceipt());

pw.println(mass[i].getAttribute());

pw.println(mass[i].getValue());

}

pw.close();

} catch (IOException e) {

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

}

}

void read() throws IOException {

BufferedReader br = null;

Store temp = new Store();

try {

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

String line;

line = br.readLine();

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);

mass[i] = new Store(temp);

}

} catch (IOException ex) {

ex.printStackTrace();

} finally {

br.close();

}

}

}

**class** Store

**package** ua.khpi.oop.Ohonkova08;

**import** java.util.Scanner;

**class** Store {

**private** String name;

**private** String unit;

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

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

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

**private** String attribute;

**private** String value;

Store() {

name = **null**;

unit = **null**;

count = 0;

unit\_price = 0;

data\_of\_receipt = **null**;

attribute = **null**;

value = **null**;

}

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** setName(String name) {

**this**.name = name;

}

**public** **void** setUnit(String unit) {

**this**.unit = unit;

}

**public** **void** setCount(**int** count) {

**this**.count = count;

}

**public** **void** setUnitPrice(**int** unit\_price) {

**this**.unit\_price = unit\_price;

}

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

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

}

**public** **void** setAttribute(String attribute) {

**this**.attribute = attribute;

}

**public** **void** setValue(String value) {

**this**.value = value;

}

**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;

}

**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();

}

}

**class** Func

**package** ua.khpi.oop.Ohonkova08;

**final** **class** Func {

**private** Func() {

//default constructor

}

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

System.***out***.println("\n\nВыберите вариант:");

System.***out***.println("1. Добавить товар");

System.***out***.println("2. Показать существующие товары");

System.***out***.println("3. Удалить товар");

System.***out***.println("4. Очистить список товаров");

System.***out***.println("5. Использовать XML");

System.***out***.println("6. Работа с файлами");

System.***out***.println("0. Завершить работу");

System.***out***.print("Выберите: ");

}

**static** **void** chooseMenuFile() {

System.***out***.println("\n\nВыберите вариант:");

System.***out***.println("1. Сохранить файл в текущую директорию");

System.***out***.println("2. Просмотреть содержание директории");

System.***out***.println("3. Просмотреть дерево к файлу");

System.***out***.println("4. Сохранить файл в необходимый путь");

System.***out***.println("5. Переименовать файл");

System.***out***.println("6. Прочитать файл из памяти");

System.***out***.println("0. Завершить работу");

System.***out***.print("Выберите: ");

}

}

1. ВАРІАНТИ ВИКОРИСТАННЯ

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

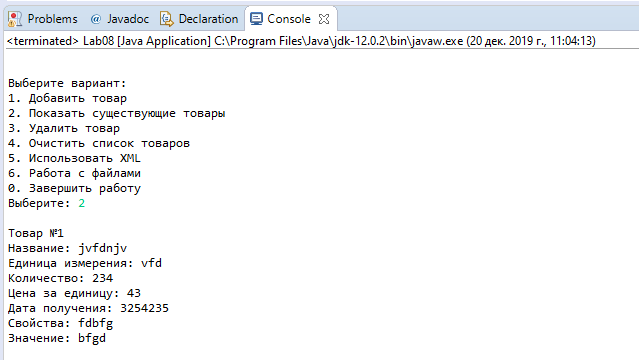


Рисунок 2 – Результат програми

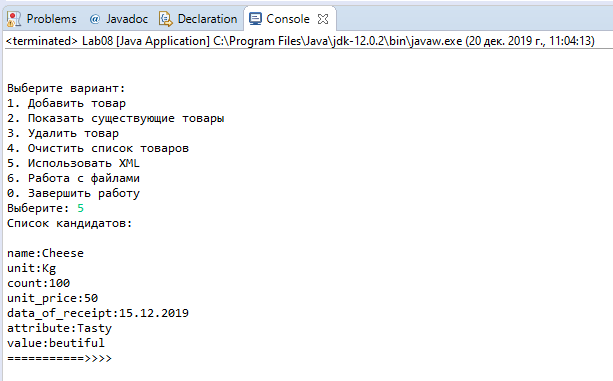


Рисунок 3 – Результат програми

ВИСНОВОК

При виконанні лабораторної роботи набуто практичних навичок, щодо управління введенням/виведенням даних з використанням класів платформи Java SE. Програма виконується без помилок.