Автор: Кравчук В.П., КІТ101.8а

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.

ЗАВДАННЯ ДО РОБОТИ

Забезпечити можливість збереження і відновлення масива об’єктів рішення завдання лабораторної роботи №7, використовуючи модель Long Term Persistence.

***Ієрархія класів***

final class Laba8 – головний клас. Містить метод main(точку входу у програму) та .

class Functions - допоміжний клас в якому реалізуються функції

class Directory - клас прикладної задачі кадрового агенства

class Container - клас-контейнер, котрий зберігає інформацію агенства

class MyFileVisitor – клас для реалізації відображення вмісту директорії

**ТЕКСТ ПРОГРАМИ**

File Laba8.java:

package laba8;

import org.w3c.dom.\*;

//import org.w3c.dom.Node;

//import org.w3c.dom.NodeList;

import org.xml.sax.SAXException;

import javax.xml.parsers.\*;

//import javax.xml.parsers.DocumentBuilderFactory;

//import javax.xml.parsers.ParserConfigurationException;

import java.io.IOException;

import java.nio.file.\*;

import java.util.HashSet;

import java.util.Scanner;

public class Exe {

public static void exe() throws IOException {

Container contains = new Container();

Directory rec1 = new Directory();

Scanner scan = new Scanner(System.in);

String str = "";

String str3 = "";

boolean loop = true;

while (loop) {

Functions.chooseMenu();

int choose = scan.nextInt();

switch (choose) {

case 1:

rec1.generateDirectory();

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.newInstance();

try {

DocumentBuilder builder = factory.newDocumentBuilder();

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

Node root = document.getDocumentElement();

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

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) {

Functions.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 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;

}

}

}

}

Container.java:

package laba8;

import java.util.Arrays;

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 laba8.Directory;

class Container implements Serializable {

private final int size = 10;

private int count = 0;

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

void add(final Directory temp) {

Directory buff = new Directory(temp);

if (count == mass.length) {

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

mass[count++] = buff;

} else {

mass[count++] = buff;

}

}

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].getStore());

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

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

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

pw.println(mass[i].getWork\_time());

}

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].getStore());

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

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

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

pw.println(mass[i].getWork\_time());

}

pw.close();

} catch (IOException e) {

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

}

}

void read() throws IOException {

BufferedReader br = null;

Directory temp = new Directory();

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.setStore(line);

line = br.readLine();

temp.setAddress(line);

line = br.readLine();

temp.setPhone(line);

line = br.readLine();

temp.setSpecialization(line);

line = br.readLine();

temp.setWork\_time(line);

mass[i] = new Directory(temp);

}

} catch (IOException ex) {

ex.printStackTrace();

} finally {

br.close();

}

}

void show() {

if (count == 0) {

System.out.println("Ошибка! Пустой массив!");

} else {

System.out.println();

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

System.out.println("Магазин #" + (i + 1));

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

System.out.println("Адрес: " + mass[i].getAddress());

System.out.println("Телефон: " + mass[i].getPhone());

System.out.println("Специализация: " + mass[i].getSpecialization());

System.out.println("Время работы: " + mass[i].getWork\_time());

}

}

}

void clear() {

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

mass[i] = null;

}

count = 0;

}

public void remove(final int index) {

int temp = index;

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

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

}

mass[count] = null;

count--;

}

int size() {

return count;

}

}

Directory.java :

**package** laba8;

**import** java.util.Scanner;

**class** Directory {

**private** String store;

**private** String address;

**private** String phone;

**private** String specialization;

**private** String work\_time;

**public** **void** setStore(String store) {

**this**.store = store;

}

**public** **void** setAddress(String address) {

**this**.address = address;

}

**public** **void** setPhone(String phone) {

**this**.phone = phone;

}

**public** **void** setSpecialization(String specialization) {

**this**.specialization = specialization;

}

**public** **void** setWork\_time(String work\_time) {

**this**.work\_time = work\_time;

}

String getSpecialization() {

**return** specialization;

}

String getPhone() {

**return** phone;

}

String getAddress() {

**return** address;

}

String getStore() {

**return** store;

}

String getWork\_time() {

**return** work\_time;

}

Directory() {

store = **null**;

address = **null**;

phone = **null**;

specialization = **null**;

work\_time = **null**;

}

Directory(**final** Directory obj) {

store = obj.store;

address = obj.address;

phone = obj.phone;

specialization = obj.specialization;

work\_time = obj.work\_time;

}

**void** generateDirectory() {

Scanner scan = **new** Scanner(System.***in***);

System.***out***.print("\nВведите название магазина: ");

store = scan.nextLine();

System.***out***.print("\nВведите адрес магазина: ");

address = scan.nextLine();

System.***out***.print("\nВведите телефон магазина: ");

phone = scan.nextLine();

System.***out***.print("\nВведите специализацию магазина: ");

specialization = scan.nextLine();

System.***out***.print("\nВведите время работы: ");

work\_time = scan.nextLine();

}

@Override

**public** String toString() {

**return** "Recruitment{" + "store='" + store + '\'' + ", address='" + address + '\'' + ", phone='" + phone + '\''

+ ", specialization=" + specialization + ", work\_time='" + work\_time + '\'' + '}';

}

}

**Functions.java:**

**package** laba8;

**final** **class** Functions {

**private** Functions() {

// default constructor

}

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

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

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

System.***out***.println("2. Показать текущий справочник");

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

System.***out***.println("4. Очистить справочник");

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 необходимый 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: ");

}

}

MyFileVisitor.java:

package laba8;

import java.io.IOException;

import java.nio.file.FileVisitResult;

import java.nio.file.Path;

import java.nio.file.SimpleFileVisitor;

import java.nio.file.attribute.BasicFileAttributes;

class MyFileVisitor extends SimpleFileVisitor<Path> {

@Override

public FileVisitResult visitFile(final Path file, final BasicFileAttributes attrs) throws IOException {

System.out.println(file.getFileName());

return FileVisitResult.CONTINUE;

}

}

**ВАРІАНТИ ВИКОРИСТАННЯ**

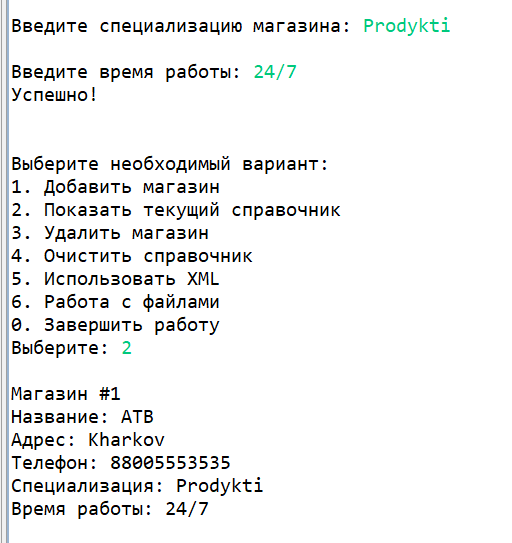
****

Рис. 8.1 – Результат роботи програми

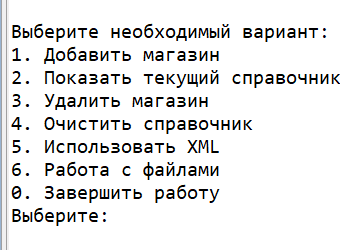


Рис. 8.2 – Результат роботи програми

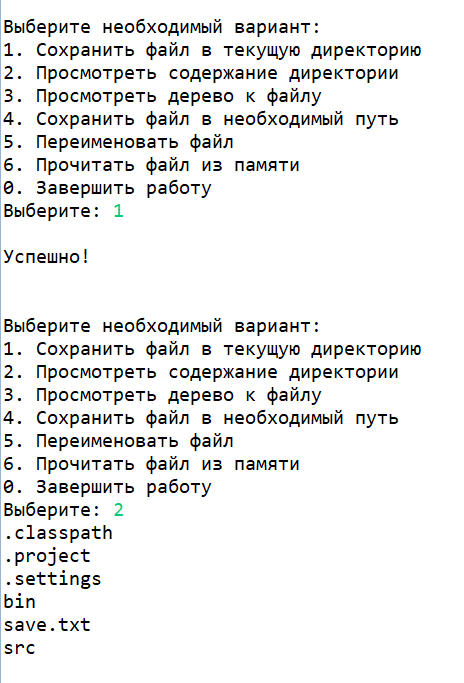


Рис. 8.3 – Результат роботи програми

Програму можна використовувати задля створення бази даних. А саме довідник покупця, в якому представляються різноманітні магазини. Також для вибору доступно багато інших можливостей. Реалізовано текстове меню задля комфортного використання функцій програми.

**ВИСНОВКИ**

При виконанні лабораторної роботи набуто практичних навичок щодо використання моделі Long Term Persistence, яка являє собою зберігання об’єктів в XML-файл та їх зчитування. Завдання виконане! Програма працює успішно!