Звіт

Автор: Васильченко С., 1.КІТ101.8б

Дата: 07.11.2019

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.

ОПИС ПРОГРАМИ

***2.1 Опис змінних:***

Kontainer kontain = new Kontainer(); // об’єкт контейнера

Recruitment rec1 = new Recruitment(); // об’єкт класа кадрового агенства

Recruitment rec2 = new Recruitment(); // об’єкт класа кадрового агенства

Scanner scan = new Scanner(System.in); // змінна для активування зчитування з консолі

String str = ""; // змінна строки для використання у програмі

String str3 = ""; // змінна строки для використання у програмі

***2.2 Ієрархія та структура класів.***

final class Lab08 – головний клас. Містить метод main(точку входу у програму) та .

class Functions - допоміжний клас в якому реалізуються функції

class Recruitment - клас прикладної задачі кадрового агенства

class Kontainer - клас-контейнер, котрий зберігає інформацію агенства

class MyFileVisitor – клас для реалізації відображення вмісту директорії

**ТЕКСТ ПРОГРАМИ**

File Lab08.java:

package sergey.Lab08;

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.beans.XMLDecoder;

import java.beans.XMLEncoder;

import java.io.\*;

import java.nio.file.FileVisitOption;

import java.nio.file.Files;

import java.nio.file.Path;

import java.nio.file.Paths;

import java.nio.file.StandardCopyOption;

import java.util.HashSet;

import java.util.Scanner;

final class Lab08 {

private Lab08() {

//default constructor

}

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

Kontainer contains = new Kontainer();

Recruitment rec1 = new Recruitment();

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.generateVacancy();

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!");

}

}

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 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;

}

}

}

}

Kontainer.java:

package sergey.Lab08;

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;

class Kontainer implements Serializable {

/\*\*

\* First size for array.

\*/

private final int size = 10;

/\*\*

\* Array contains all data.

\*/

private int count = 0;

private Recruitment[] massik = new Recruitment[size];

void add(final Recruitment temp) {

Recruitment buff = new Recruitment(temp);

if (count == massik.length) {

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

massik[count++] = buff;

} else {

massik[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(massik[i].getFirm());

pw.println(massik[i].getSpecialty());

pw.println(massik[i].getWorkingConditions());

pw.println(massik[i].getPayment());

pw.println(massik[i].getConfirms());

if (massik[i].getConfirms()) {

pw.println(massik[i].getNeedsSpeciality());

pw.println(massik[i].getExperience());

pw.println(massik[i].getEducation());

}

}

pw.close();

} catch (IOException e) {

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

}

}

void saveInDirectory() {

String str = "";

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

}

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(massik[i].getFirm());

pw.println(massik[i].getSpecialty());

pw.println(massik[i].getWorkingConditions());

pw.println(massik[i].getPayment());

pw.println(massik[i].getConfirms());

if (massik[i].getConfirms()) {

pw.println(massik[i].getNeedsSpeciality());

pw.println(massik[i].getExperience());

pw.println(massik[i].getEducation());

}

}

pw.close();

} catch (IOException e) {

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

}

}

void read() throws IOException {

BufferedReader br = null;

Recruitment temp = new Recruitment();

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.setFirm(line);

line = br.readLine();

temp.setSpecialty(line);

line = br.readLine();

temp.setWorkingConditions(line);

line = br.readLine();

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

line = br.readLine();

temp.setConfirm(Boolean.parseBoolean(line));

if (temp.getConfirms()) {

line = br.readLine();

temp.setNeedsSpeciality(line);

line = br.readLine();

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

line = br.readLine();

temp.setEducation(line);

}

massik[i] = 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("Вакансия #" + (i + 1));

System.out.println("Фирма: " + massik[i].getFirm());

System.out.println("Специальность: " + massik[i].getSpecialty());

System.out.println("Условия работы: " + massik[i].getWorkingConditions());

System.out.println("Оплата: " + massik[i].getPayment());

if (massik[i].getConfirms()) {

System.out.println("Необходимая специальность: " + massik[i].getNeedsSpeciality());

System.out.println("Опыт: " + massik[i].getExperience());

System.out.println("Образование: " + massik[i].getEducation());

}

}

}

}

void clear() {

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

massik[i] = null;

}

count = 0;

}

public void remove(final int index) {

int temp = index;

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

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

}

massik[count] = null;

count--;

}

int size() {

return count;

}

}

Recruitment.java :

package sergey.Lab08;

import java.util.Scanner;

class Recruitment {

private String firm;

private String specialty;

private String workingConditions;

private int payment;

private String needsSpeciality;

private int experience;

private String education;

private boolean confirm = false;

public void setFirm(final String firm) {

this.firm = firm;

}

public void setSpecialty(final String specialty) {

this.specialty = specialty;

}

public void setWorkingConditions(final String workingConditions) {

this.workingConditions = workingConditions;

}

public void setPayment(final int payment) {

this.payment = payment;

}

public void setConfirm(final boolean confirm) {

this.confirm = confirm;

}

int getPayment() {

return payment;

}

String getWorkingConditions() {

return workingConditions;

}

String getSpecialty() {

return specialty;

}

String getFirm() {

return firm;

}

Recruitment() {

firm = null;

specialty = null;

workingConditions = null;

payment = 0;

needsSpeciality = null;

experience = 0;

education = null;

confirm = false;

}

Recruitment(final Recruitment obj) {

firm = obj.firm;

specialty = obj.specialty;

workingConditions = obj.workingConditions;

payment = obj.payment;

needsSpeciality = obj.needsSpeciality;

experience = obj.experience;

education = obj.education;

confirm = obj.confirm;

}

void generateVacancy() {

Scanner scan = new Scanner(System.in);

Scanner scan2 = new Scanner(System.in);

int choose = 0;

System.out.print("\nВведите фирму: ");

firm = scan.nextLine();

System.out.print("\nВведите специальность: ");

specialty = scan.nextLine();

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

workingConditions = scan.nextLine();

System.out.print("\nВведите оплату: ");

payment = scan.nextInt();

System.out.println("Желаете добавить дополнительные условия работы? 1 - Да. 0 - Нет");

choose = scan.nextInt();

while (true) {

if (choose == 1) {

System.out.print("\nВведите необходимую специальность: ");

needsSpeciality = scan2.nextLine();

System.out.print("\nНеобходимое образование: ");

education = scan2.nextLine();

System.out.print("\nНеобходимый опыт работы: ");

experience = scan2.nextInt();

confirm = true;

break;

} else if (choose == 0) {

needsSpeciality = null;

experience = 0;

education = null;

break;

} else {

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

}

}

}

public void setExperience(final int experience) {

this.experience = experience;

}

public int getExperience() {

return experience;

}

public void setNeedsSpeciality(final String needsSpeciality) {

this.needsSpeciality = needsSpeciality;

}

public String getNeedsSpeciality() {

return needsSpeciality;

}

public void setEducation(final String education) {

this.education = education;

}

public String getEducation() {

return education;

}

public boolean getConfirms() {

return confirm;

}

@Override

public String toString() {

return "Recruitment{" +

"firm='" + firm + '\'' +

", specialty='" + specialty + '\'' +

", workingConditions='" + workingConditions + '\'' +

", payment=" + payment +

", needsSpeciality='" + needsSpeciality + '\'' +

", experience=" + experience +

", education='" + education + '\'' +

", confirm=" + confirm +

'}'; }}

**Functions.java:**

package sergey.Lab08;

final class Functions {

private Functions() {

//default constructor

}

static void chooseMenu() {

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

System.out.println("1. Добавить вакансию");

System.out.println("2. Показать действующие вакансии");

System.out.println("3. Удалить вакансию");

System.out.println("4. Очистить список вакансий");

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 sergey.Lab08;

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**

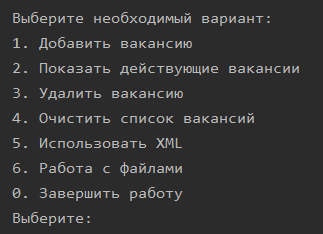
****

Рис. 8.1 – Результат роботи програми

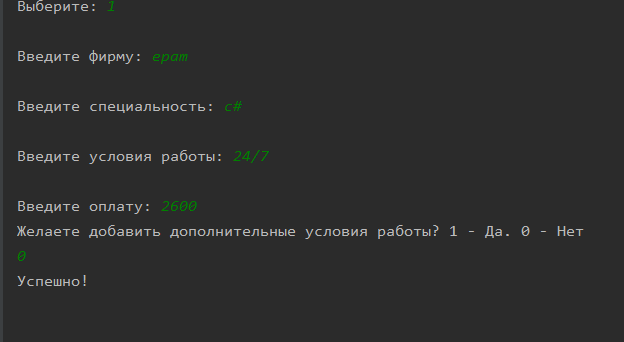


Рис. 8.2 – Результат роботи програми

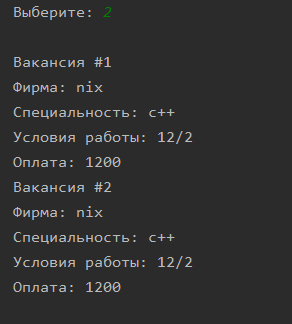


Рис. 8.3 – Результат роботи програми

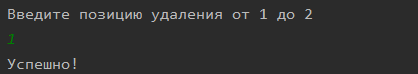
****

Рис. 8.4 – Результат роботи програми

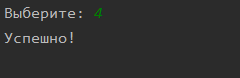
****

Рис. 8.5 – Результат роботи програми

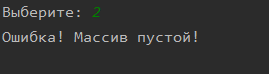
****

Рис. 8.6 – Результат роботи програми

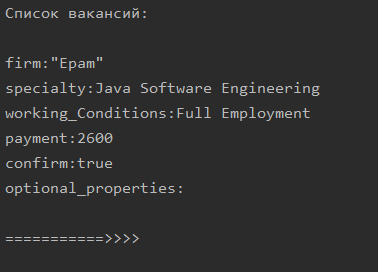
****

Рис. 8.7 – Результат роботи програми

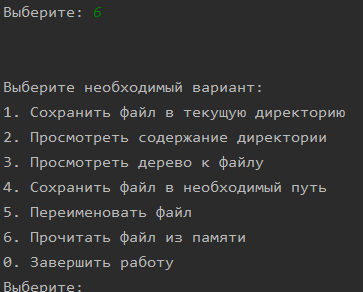
****

Рис. 8.8 – Результат роботи програми

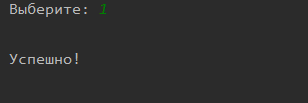
****

Рис. 8.9 – Результат роботи програми

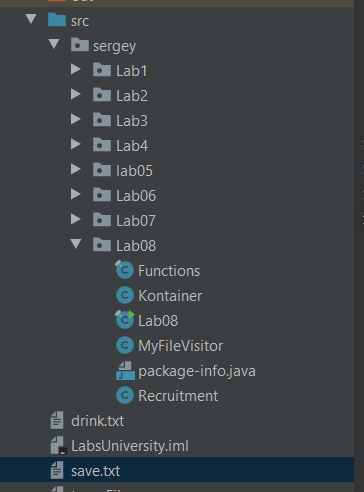
****

Рис. 8.10 – Результат роботи програми

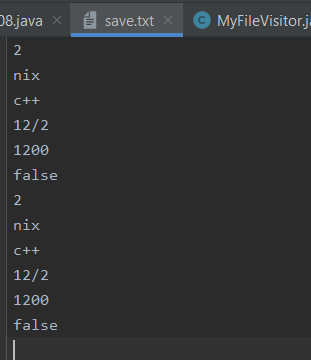
****

Рис. 8.11 – Результат роботи програми

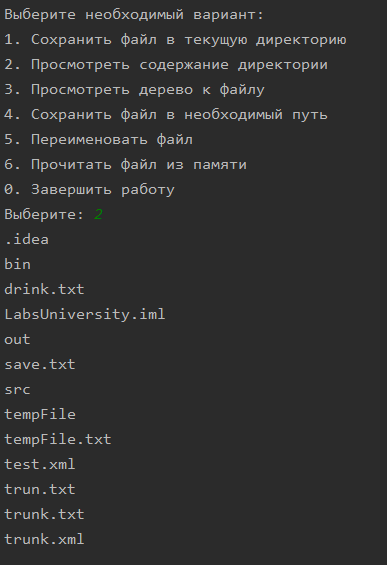
****

Рис. 8.12 – Результат роботи програми

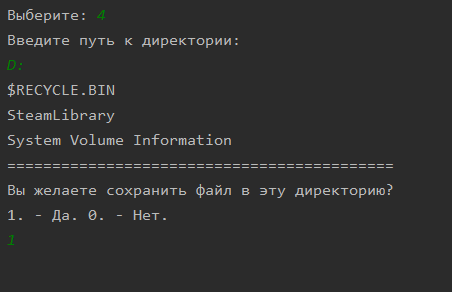
****

Рис. 8.13 – Результат роботи програми

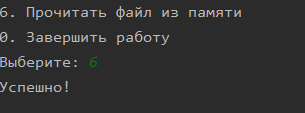
****

Рис. 8.14 – Результат роботи програми

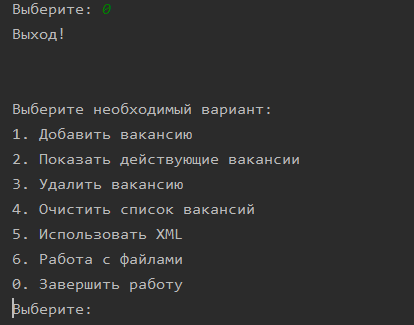
****

Рис. 8.15 – Результат роботи програми

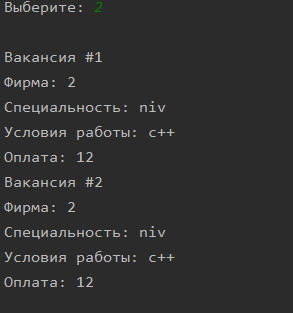
****

Рис. 8.16 – Результат роботи програми

Програму можна використовувати задля створення бази даних. А саме кадрове агенство, в якому представляються різноманітні вакансії. 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!