Звіт

Автор: Пумня О., КІТ101.8б

Дата: 07.10.2019

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

Опис змінних

SchedulerEvent[] events;	// набір заходів
Scanner in;	// для отримання даних з клавіатури
String choice;	// зберігання вибору користувача

Ієрархія та структура класів

class Pumnya08 – точка входу в програму.

class UI – реалізація діалогового режиму.

ТЕКСТ ПРОГРАМИ

Текст файлу Pumnya08.java

```
package labs.pumnya08;

import labs.pumnya07.SchedulerEvent;
```

```
import java.io.IOException;
import java.text.ParseException;
import java.util.Scanner;

public final class Pumnya08 {
    private Pumnya08() {
    }
    /**
     * Точка входа, главный метод.
     * @param args - аргументы главного метода
     * @throws IOException - при неудачной
     * работе с файлами
     * @throws ParseException - при неудачном
     * парса
     */
    public static void main(final String[] args)
            throws IOException, ParseException {
        SchedulerEvent[] events = new SchedulerEvent[0];
        Scanner in = new Scanner(System.in);
        String choice;
        do {
            UI.mainMenu();
            choice = UI.getChoice();
            switch (choice) {
                case "1":
                    events = UI.addEvent(events);
                    break;
                case "2":
                    System.out.print("Введите индекс"
                            + " удаляемого мероприятия: ");
                    int index = in.nextInt();
                    events = UI.dropEvent(events, index);
                    break;
                case "3":
                    UI.printInfo(events);
                    System.out.println();
                    break;
                case "4":
                    UI.saveToFile(events);
                    break;
                case "5":
                    try {
                        SchedulerEvent[] newEvents = UI.loadFromFile();
                        UI.printInfo(newEvents);
                    } catch (IOException e) {
                        System.out.println(e.toString());
                    }
                    break;
                case "0":
                    System.out.println("Выход...");
                    break;
                default:
                    System.out.println("Введите номер"
                            + " одного из пунктов!\n");
            }
        } while (!choice.equals("0"));
    }
}
```

Текст файлу UI.java

```
package labs.pumnya08;

import labs.pumnya07.SchedulerEvent;

import java.beans.XMLDecoder;
import java.beans.XMLEncoder;
import java.io.BufferedReader;
import java.io.File;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.text.ParseException;
```

```
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public final class UI {
    private UI() {
    }
    /** Для чтения данных с клавиатуры. */
    private static BufferedReader buffer = new BufferedReader(
            new InputStreamReader(System.in));
    /**
     * Главное меню.
     */
    public static void mainMenu() {
        System.out.println("1. Добавление мероприятия.");
        System.out.println("2. Удаление мероприятия.");
        System.out.println("3. Вывод информации.");
        System.out.println("4. Сохранить в файл.");
        System.out.println("5. Загрузить из файла.");
        System.out.println("0. Выход.");
        System.out.print("Введите ваш ответ сюда: ");
    }
    /**
     * Получение ответа диалогового меню.
     * @return ответ
     * @throws IOException при неудачном считывании
     */
    public static String getChoice() throws IOException {
        return buffer.readLine();
    }
    /**
     * Добавление мероприятия.
     * @param events - исходный набор мероприятий
     * @return новый набор мероприятий
     * @throws IOException при неудачном считывании
     * @throws ParseException при неудачном парсинге
     */
    public static SchedulerEvent[] addEvent(final SchedulerEvent[] events)
            throws IOException, ParseException {
        SchedulerEvent[] newEvents = new SchedulerEvent[events.length + 1];
        System.arraycopy(events, 0, newEvents, 0, events.length);
        newEvents[events.length] = SchedulerEvent.generate();
        return newEvents;
    }
    /**
     * Удаление мероприятия.
     * @param events - исходный набор мероприятий
     * @param pos - позиция удаляемого мероприятия
     * @return новый набор мероприятий
     */
    public static SchedulerEvent[] dropEvent(final SchedulerEvent[] events,
                                             final int pos) {
        if (pos >= events.length) {
            System.out.println("Error. Out of bounds.");
            return events;
        } else {
            SchedulerEvent[] newEvents = new SchedulerEvent[events.length - 1];
            int i = 0;
            for (int j = 0; j < events.length; j++) {
                if (j == pos) {
                    continue;
                } else {
                    newEvents[i] = events[j];
                    i++;
                }
            }
            return newEvents;
        }
    }
    /**
     * Вывод информации о мероприятиях.
     * @param events - набор мероприятий
     */
    public static void printInfo(final SchedulerEvent[] events) {
        for (int i = 0; i < events.length; i++) {
            System.out.format("%nМЕРОПРИЯТИЕ %d:%n", i + 1);
            System.out.println(events[i].printInfo());
        }
```

```
    }
    /**
     * Сохранение объекта в файл.
     * @param events - набор мероприятий
     * @throws IOException при неудачном
     * считывании или записи.
     */
    public static void saveToFile(final SchedulerEvent[] events)
            throws IOException {
        final String semi = ";";
        Pattern pattern = Pattern.compile(semi);
        Matcher matcher;
        String path;
        StringBuilder direct = new StringBuilder();
        System.out.print("Введите имя директории: ");
        while (true) {
            System.out.print(direct.toString());
            path = buffer.readLine();
            direct.append(path).append("\\");
            matcher = pattern.matcher(direct.toString());
            if (matcher.find()) {
                break;
            }
            File directory = new File(direct.toString());
            File[] list = directory.listFiles();
            if (list == null) {
                System.out.println("Неверное"
                        + " имя директории!");
                if (direct.length() != 0) {
                    direct.delete(
                            direct.length() - path.length() - 1,
                            direct.length());
                }
                continue;
            }
            System.out.println("---------------");
            for (File it : list) {
                if (it.isDirectory()) {
                    System.out.print(it.getName());
                    System.out.println(" (...)");
                    continue;
                }
                System.out.println(it.getName());
            }
            System.out.println("---------------");
        }
        String currentDir = direct.toString();
        currentDir = currentDir.replaceAll(semi, "");
        FileOutputStream fos = new FileOutputStream(
                currentDir + "\\Encoded.xml");
        XMLEncoder xmlEncoder = new XMLEncoder(new BufferedOutputStream(fos));
        xmlEncoder.writeObject(events);
        xmlEncoder.close();
    }
    /**
     * Загрузка обьекта из файла.
     * @return массив объектов
     * @throws IOException при неудачном
     * считывании или записи.
     */
    public static SchedulerEvent[] loadFromFile() throws IOException {
        System.out.print("Введите имя директории: ");
        String dirToExtract = buffer.readLine();
        FileInputStream fis = new FileInputStream(dirToExtract);
        XMLDecoder xmlDecoder = new XMLDecoder(new BufferedInputStream(fis));
        SchedulerEvent[] getEvents = (SchedulerEvent[]) xmlDecoder.readObject();
        xmlDecoder.close();
        return getEvents;
    }
}
```

ВАРІАНТИ ВИКОРИСТАННЯ

```
1. Добавление мероприятия.
2. Удаление мероприятия.
3. Вывод информации.
4. Сохранить в файл.
5. Загрузить из файла.
0. Выход.
```

Рисунок 1 – Діалогове меню

```
1. Добавление мероприятия.
2. Удаление мероприятия.
3. Вывод информации.
4. Сохранить в файл.
5. Загрузить из файла.
0. Выход.
Введите ваш ответ сюда: 3

МЕРОПРИЯТИЕ 1:
Дата: 15.11.2019
Время начала: 12:20
Длительность: 150 минут
Место проведения: Французский бульвар
Описание: Поход в кино.
Участники: Александр Сергей Александра Иван
```

Рисунок 2 – Виведення інформації про заходи

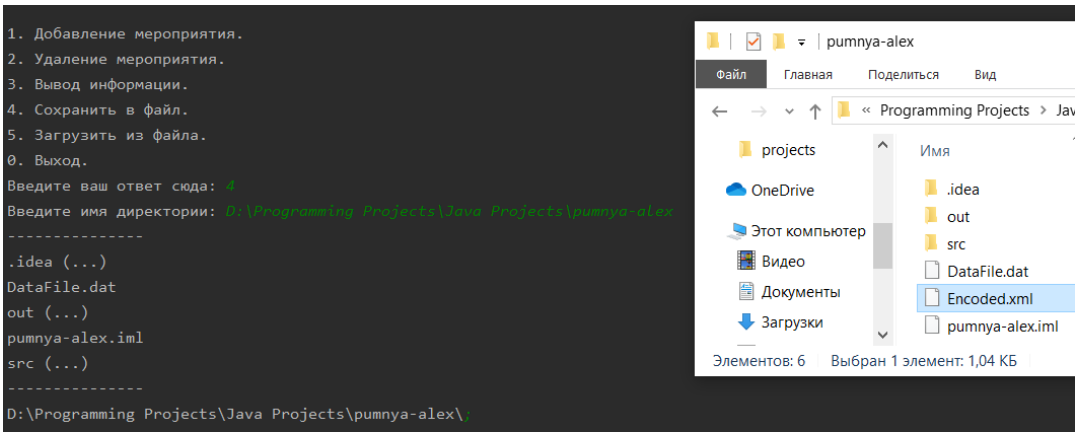

Рисунок 3 – Виконання зберігання об’єкта у файл

```
1. Добавление мероприятия.
2. Удаление мероприятия.
3. Вывод информации.
4. Сохранить в файл.
5. Загрузить из файла.
0. Выход.
Введите ваш ответ сюда: 5
Введите имя директории: Encoded.xml

МЕРОПРИЯТИЕ 1:
Дата: 15.11.2019
Время начала: 12:20
Длительность: 150 минут
Место проведения: Французский бульвар
Описание: Поход в кино.
Участники: Александр Сергей Александра Иван
```

Рисунок 4 – Виконання зчитування з файлу

Програму можна використовувати для планування заходів. Реалізовано додавання та видалення заходів.

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.