Звіт

Автор: Пумня О., КІТ118Б

Дата: 25.02.2020

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.

ОПИС ПРОГРАМИ

**Ієрархія та структура класів**

ТЕКСТ ПРОГРАМИ

**Текст файлу Pumnya15.java**

package labs.pumnya15;

import labs.pumnya07.SchedulerEvent;

import labs.pumnya10.SortByPartAmount;

import java.io.\*;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.LinkedList;

import java.util.List;

public class Pumnya15 {

private Pumnya15() {

// Пустое тело

}

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

boolean isExit = false;

List<String> paramList = Arrays.asList(args);

boolean isAuto = paramList.contains("-auto");

if (isAuto) {

LinkedList<SchedulerEvent> list = new LinkedList<>();

list.add(SchedulerEvent.generate(false));

SchedulerEvent event = new SchedulerEvent();

event.setDate("20.05.2019");

event.setDescription("Конференция.");

event.setDuration(26);

event.setTime("20:20");

event.setVenue("Харьков.");

event.setParticipants(new ArrayList<>(Arrays.asList("Алекс", "Другие")));

list.add(event);

System.out.println("Данные:");

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

System.out.println("Сортировка по количеству участников:");

list.sort(new SortByPartAmount(null));

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

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

ObjectOutputStream oos = new ObjectOutputStream(

new FileOutputStream("DataFile.dat"));

oos.writeObject(list);

oos.close();

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

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

ObjectInputStream ois = new ObjectInputStream(

new FileInputStream("DataFile.dat"));

LinkedList list\_copy =

(LinkedList) ois.readObject();

ois.close();

System.out.println("Прочитанные данные: ");

System.out.println(list\_copy.toString());

} else {

while (!isExit) {

isExit = Dialog.run();

}

}

}

}

**Текст файлу Dialog.java**

package labs.pumnya15;

import labs.pumnya07.SchedulerEvent;

import labs.pumnya10.SortByDate;

import labs.pumnya10.SortByDuration;

import labs.pumnya10.SortByPartAmount;

import java.beans.XMLDecoder;

import java.beans.XMLEncoder;

import java.io.\*;

import java.util.LinkedList;

import java.util.Scanner;

public class Dialog {

private Dialog() {

// Пустое тело

}

/\*\* Двусвязый список. \*/

private static LinkedList<SchedulerEvent> list = new LinkedList<>();

/\*\* Для выбора пунктов в меню. \*/

private static String choice;

/\*\* Для ввода. \*/

private static BufferedReader buffer = new BufferedReader(

new InputStreamReader(System.in));

/\*\*

\* Главный метод, сердце диалога.

\* @return true - если выбран выход

\* @throws IOException при ошибках со вводом

\*/

public static boolean run() throws IOException, ClassNotFoundException {

mainMenu();

boolean flag = mainProcessing();

System.out.println();

return flag;

}

/\*\*

\* Главное меню диалога.

\* @throws IOException при ошибках со вводом

\*/

private static void mainMenu() throws IOException {

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

System.out.println("2. Удалить мероприятия.");

System.out.println("3. Очистить список.");

System.out.println("4. Вывод информации.");

System.out.println("5. Сортировка.");

System.out.println("6. Поиск мероприятий по дате.");

System.out.println("7. Сериализация.");

System.out.println("8. Десериализация.");

System.out.println("9. Сохранить в файл.");

System.out.println("10. Загрузить из файла.");

System.out.println("0. Выход.");

System.out.print("Введите ваш ответ сюда: ");

choice = buffer.readLine();

System.out.println();

}

/\*\*

\* Обработка выбора главного меню.

\* @return true - если выход

\* @throws IOException при ошибках со вводом

\* @throws ClassNotFoundException при ошибке с классами

\*/

private static boolean mainProcessing() throws IOException, ClassNotFoundException {

switch (choice) {

case "1":

list.add(SchedulerEvent.generate());

return false;

case "2":

onDelete();

deleteProcessing();

return false;

case "3":

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

System.out.println("Список пуст!");

return false;

}

System.out.println("Очистка...");

list.clear();

return false;

case "4":

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

System.out.println("Список пуст!");

return false;

}

System.out.println("Данные: ");

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

return false;

case "5":

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

System.out.println("Список пуст!");

return false;

}

onSort();

sortProcessing();

return false;

case "6":

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

System.out.println("Список пуст!");

return false;

}

System.out.println("Введите нужную дату: ");

String date = buffer.readLine();

SchedulerEvent[] events = new SchedulerEvent[list.size()];

for (int i = 0, j = 0; i < list.size(); i++) {

if(list.get(i).getDate().equals(date)) {

events[j++] = list.get(i);

}

}

for (SchedulerEvent event : events) {

if (event != null) {

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

}

}

return false;

case "7":

serialize();

return false;

case "8":

deserialize();

return false;

case "9":

saveToFile();

return false;

case "10":

loadFromFile();

return false;

case "0":

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

return true;

default:

return false;

}

}

/\*\*

\* Меню удаления мероприятия.

\* @throws IOException при ошибках со вводом

\*/

private static void onDelete () throws IOException {

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

System.out.println("Список пуст!");

} else {

System.out.println("1. Удалить первый.");

System.out.println("2. Удалить последний.");

System.out.println("3. Удалить по индексу.");

System.out.println("Любая клавиша. Назад.");

System.out.print("Введите ваш ответ сюда: ");

choice = buffer.readLine();

System.out.println();

}

}

/\*\* Обработка выбора меню удаления. \*/

private static void deleteProcessing() {

Scanner scan = new Scanner(System.in);

switch (choice) {

case "1":

list.removeFirst();

break;

case "2":

list.removeLast();

break;

case "3":

System.out.print("Введите индекс:");

list.remove(scan.nextInt());

break;

}

}

/\*\*

\* Меню сортировки мероприятий.

\* @throws IOException при ошибках со вводом

\*/

private static void onSort() throws IOException {

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

System.out.println("Список пуст!");

} else {

System.out.println("1. Сортировка по дате.");

System.out.println("2. Сортировка по длительности.");

System.out.println("3. Сортировка по количеству участников.");

System.out.println("Любая клавиша. Назад.");

System.out.print("Введите ваш ответ сюда: ");

choice = buffer.readLine();

System.out.println();

}

}

/\*\* Обработка выбора меню сортировки. \*/

private static void sortProcessing() {

switch (choice) {

case "1":

list.sort(new SortByDate(null));

break;

case "2":

list.sort(new SortByDuration(null));

break;

case "3":

list.sort(new SortByPartAmount(null));

break;

}

}

/\*\* Сохранение данных стандартной сериализацией. \*/

private static void serialize() {

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

try {

ObjectOutputStream oos = new ObjectOutputStream(

new FileOutputStream("DataFile.dat"));

oos.writeObject(list);

} catch(IOException e) {

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

}

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

}

/\*\* Чтение данных стандартной сериализацией. \*/

private static void deserialize() throws ClassNotFoundException {

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

try {

ObjectInputStream ois = new ObjectInputStream(

new FileInputStream("DataFile.dat"));

LinkedList list\_copy =

(LinkedList) ois.readObject();

System.out.println("Прочитанные данные: ");

System.out.println(list\_copy.toString());

} catch (IOException e) {

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

}

}

/\*\* Сохранение данных с помощью XML. \*/

private static void saveToFile(){

System.out.println("Сохраненеие в XML...");

try {

FileOutputStream fos = new FileOutputStream("Encoded.xml");

XMLEncoder xmlEncoder = new XMLEncoder(new BufferedOutputStream(fos));

xmlEncoder.writeObject(list);

xmlEncoder.close();

} catch(FileNotFoundException e) {

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

}

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

}

/\*\* Чтение данных с помощью XML. \*/

private static void loadFromFile() {

System.out.println("Чтение из XML...");

try {

FileInputStream fis = new FileInputStream("Encoded.xml");

XMLDecoder xmlDecoder = new XMLDecoder(

new BufferedInputStream(fis));

LinkedList list2 =

(LinkedList) xmlDecoder.readObject();

System.out.print(list2.toString());

xmlDecoder.close();

} catch (IOException e) {

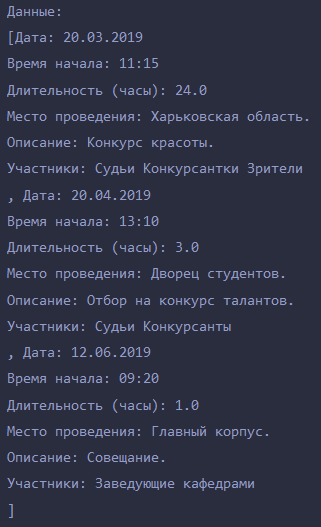
System.out.println(e.toString());

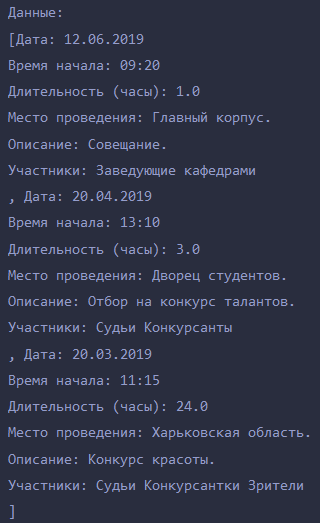
}

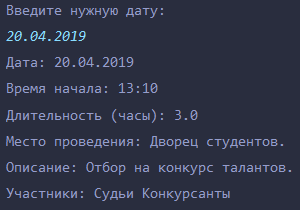
}

}

ВАРІАНТИ ВИКОРИСТАННЯ

  
Рисунок 1 – Зчитані дані з файлу

  
Рисунок 2 – Сортування за тривалістю

  
Рисунок 3 – Пошук за датою проведення

ВИСНОВКИ

При виконанні лабораторної роботи набуто практичних навичок використання контейнерів Java.