Лабораторна робота №8

[**ОСНОВИ ВВЕДЕННЯ/ВИВЕДЕННЯ JAVA SE**](https://oop-khpi.gitlab.io/#tasks)

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

**ВИМОГИ**

**Розробник:**

- Чугунов Вадим Юрійович;

- КІТ-119а;

- Варіант №24.

**Загальне завдання:**

1. Забезпечити можливість збереження і відновлення масива об'єктів рішення завдання [лабораторної роботи №7](https://oop-khpi.gitlab.io/%D0%B7%D0%B0%D0%B2%D0%B4%D0%B0%D0%BD%D0%BD%D1%8F/task07/).
2. Забороняється використання [стандартного протокола серіалізації](https://docs.oracle.com/javase/8/docs/platform/serialization/spec/serialTOC.html).
3. Продемонструвати використання моделі [Long Term Persistence](https://docs.oracle.com/javase/tutorial/javabeans/advanced/longpersistence.html).
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.

**Прикладна галузь:**

Автостанція.

Запис в розкладі: номер рейсу; час відправлення; дні тижня; кількість вільних місць; маршрут - необмежений набір значень у вигляді "назва станції, час прибуття".

**ОПИС ПРОГРАМИ**

BusStation[] Entries; // масив об'єктів, що визначає сутність записів у розкладі

String key; // змінна, яка відповідає за вибір у меню користувача

**Ієрархія та структура класів:**

**class** Main – точка входу в програму.

**class** BusStation **-**  клас, який успадковується та реалізує прикладну галузь.

**ТЕКСТ ПРОГРАМИ**

Текст файлу **Main**:

package ua.oop.khpi.chugunov08;

import ua.oop.khpi.chugunov07.BusStation;

import java.beans.XMLDecoder;

import java.beans.XMLEncoder;

import java.io.BufferedInputStream;

import java.io.BufferedOutputStream;

import java.io.BufferedReader;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.InputStreamReader;

import java.util.Scanner;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

/\*\*

\* Entry class.

\* Contains entry point of a program.

\*

\* @author chugunov-vadim

\*/

public class Main {

/\*\*

\* Main method - entry point of a program.

\* Contains user menu for library control.

\*

\* @param args - command line parameters

\* @throws IOException - if there is any unresolved input/output

\*/

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

BufferedReader reader = new BufferedReader(

new InputStreamReader(System.in));

/\* Holds objects that define the entity of books. \*/

BusStation[] Entries = null;

/\* Holds key for menu point. \*/

String key = "";

/\* Processes user's choices. \*/

while (!key.equals("5")) {

/\* User menu output. \*/

System.out.println("\n---МЕНЮ---");

System.out.println("1. Определить кол-во элементов массива");

System.out.println("2. Назначить элементы массива");

System.out.println("3. Сохранить массив на длительный срок");

System.out.println("4. Извлечь массив из файла");

System.out.println("5. Выход");

System.out.print("Введите свой выбор: ");

key = reader.readLine();

switch (key) {

case "1" :

System.out.print("\nВведите значение (кол-во записей в расписании): ");

Entries = new BusStation[Integer.parseInt(reader.readLine())];

break;

case "2" :

String init;

Scanner in = new Scanner(System.in);

int number = 0;

for (int i = 0; i < Entries.length; i++) {

System.out.print("\nЗапись в расписании #"+(i+1)+"\n");

Entries[i] = new BusStation();

System.out.print("Номер рейса: ");

number = in.nextInt();

Entries[i].setFlightNumber(number);

System.out.print("Время отправления: ");

init = reader.readLine();

Entries[i].setDepartureTime(init);

System.out.print("День недели: ");

init = reader.readLine();

Entries[i].setDayOfTheWeek(init);

System.out.print("Кол-во свободных мест: ");

number = in.nextInt();

Entries[i].setNumberOfFreeSeats(number);

System.out.print("Маршрут: \n");

System.out.print("Введите кол-во станций входящих в маршрут: ");

number = in.nextInt();

Entries[i].enterRoute(number);

}

System.out.println();

break;

case "3" :

Pattern pattern;

Matcher matcher;

StringBuilder direct = new StringBuilder("");

final String SEMICOLON = "\\\*";

pattern = Pattern.compile(SEMICOLON);

matcher = pattern.matcher("");

System.out.println("Введите каталог для сохранения массива: ");

while (!matcher.find()) {

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

direct.append(reader.readLine());

matcher = pattern.matcher(direct.toString());

File directory = new File(direct.toString());

File[] list = directory.listFiles();

if (list == null) {

System.out.println("Неправильный или результирующий каталог!");

continue;

}

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

for (File it : list) {

if (it.isDirectory()) {

System.out.print(it.getName());

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

continue;

}

System.out.println("\n" + it.getName() + "\n");

}

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

}

String currentDir = direct.toString();

currentDir = currentDir.replaceAll(SEMICOLON, "");

FileOutputStream fos = new FileOutputStream(currentDir + "\\JavaBeans.xml");

BufferedOutputStream bos = new BufferedOutputStream(fos);

XMLEncoder xmlEncoder = new XMLEncoder(bos);

xmlEncoder.writeObject(Entries);

xmlEncoder.close();

break;

case "4" :

System.out.println("Введите каталог для извлечения массива в кодировке XML: ");

String dirToExtract = reader.readLine();

FileInputStream fis = new FileInputStream(dirToExtract);

BufferedInputStream bis = new BufferedInputStream(fis);

XMLDecoder xmlDecoder = new XMLDecoder(bis);

BusStation[] getEntries = (BusStation[]) xmlDecoder.readObject();

for (int i = 0; i < getEntries.length; i++) {

System.out.format("Рейс №%d\n", i + 1);

System.out.println(getEntries[i].toString() + "\n");

}

xmlDecoder.close();

break;

case "5" :

System.out.println("Выход...");

break;

default : break;

}

}

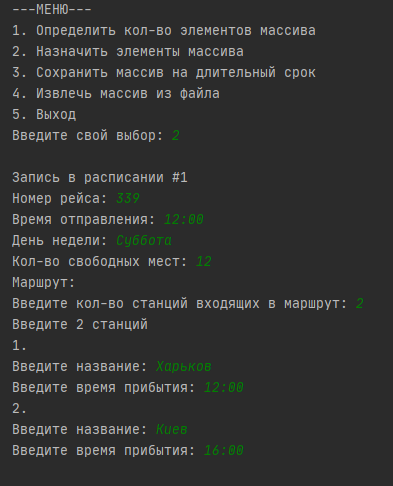
reader.close();

}

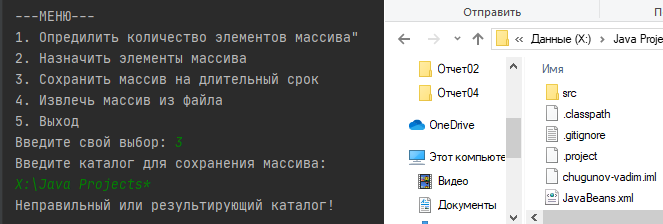
}

**РЕЗУЛЬТАТ РОБОТИ ПРОГРАМИ**

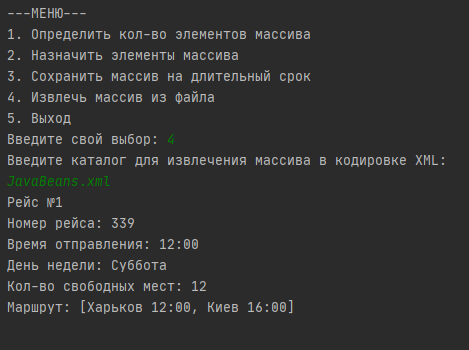
Перше зображення представляє ініціалізацію масиву об'єктів домену (рис. 8.1а). Другий показує використання моделі Long Term Persistence (одиниці довгострокової стійкості) (рис. 8.1б). Відновлення масиву об’єктів за заданою директорією (рис. 8.1в).

****

а)

****

б)



в)

Рисунок 8.1 – Результат виконання завдання

**ВАРІАНТИ ВИКОРИСТАННЯ**

Програму можна використовувати для ведення обліку маршрутів у автобусів, додавання та видалення записів у розкладі.

**ВИСНОВОК**

Під час виконання лабораторної роботи було набуто навичок управління введенням/виведенням даних з використанням класів платформи Java SE. Забезпечено можливість збереження і відновлення масиву об'єктів. Реалізовано, використання Long Term Persistence, який дає новий спосіб довгострокового збереження об’єктів домену та отримання їх у будь-який час. Масив об'єктів домену збережено та повністю отримано з XML-файлу.