Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Інженерно-хімічний факультетКафедра «Автоматизації хімічних виробництв»

Розрахункова робота

Програмування-2. Об'єктно-орієнтоване програмування

Варіант 21

Інформаційна система «Атракціон»

Виконав: студент гр. ЛА-92, Симоненко Д.А

Перевірив: к.т.н., доцент Д.О. Ковалюк

Київ 2020

ЗМІСТ

1. Технічне завдання.
2. Розділ 1 – Опис предметної області.
3. Розділ 2 – Структура програмного забезпечення.
4. Висновки.
5. Додатки.

**Технічне завдання**

Інформаційна система «Атракціон». Функції:

* сортування квитків за прізвищем
* сортування квитків за типом атракціону
* вивести всі квитки для певного типу атракціону
* додавання, редагування, видалення квитків
* завантаження та вивантаження даних у Excel та CSV форматах

Клас «квиток» має містити наступні поля:

* прізвище – строка
* тип атракціону – фіксований набір значень (тир, карусель, оглядове колесо)
* вартість квитка - ціле число
* дата придбання - дата

**Розділ 1**

Сутність «Атракціон» - атрибути:

* Прізвище покупця;
* Тип атракціону – (тир, карусель, оглядове колесо);
* Дата придбання;
* Вартість квитка;

Сутність «Атракціон» - методи:

* Отримання списку клієнтів;
* Отримання списку білетів;
* Додавання білетів з вводом дати, прізвища та типу атракціону;
* Сортування за категоріями та ім’ям;

**Розділ 2**

ClassTicket– містить інформацію про страховий поліс.

|  |  |
| --- | --- |
| private String secondName; | Прізвище білета; |
| private int cost; | Ціна білету; |
| AttractionType; | Тип атракціону – (тир, карусель, оглядове колесо); |
| GregorianCalendar ticketDate = new GregorianCalendar(); | Дата видачі; |
| public static Comparator<Ticket > sortingByName = new Comparator<Ticket >() | Сортування списку клієнтів за прізвищем; |
| public static Comparator< > sortingByCategory = new Comparator<Ticket >() | Сортування списку за категорією (тир, карусель, оглядове колесо); |

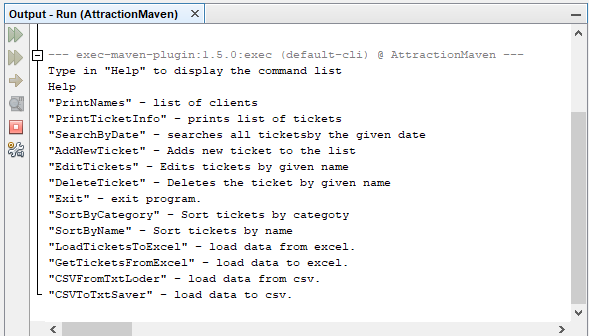
Ticket– менеджер роботи з клієнтами та полісами.

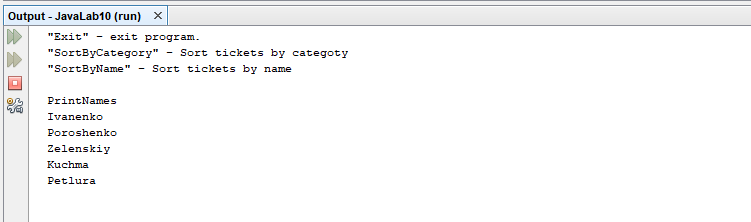
|  |  |
| --- | --- |
| public static void printListName(ArrayList<Ticket > list) | Виведення відсортованого за алфавітом списку клієнтів за прізвищем; |
| public static void printListCategory(ArrayList<Ticket> list) | Виведення відсортованого за категорією (а списку; |
| public static void editTicket (Ticket p) | Редагування страхового полісу; |
| ArrayList<Ticket > client = new ArrayList<Ticket>(); | Список квитків. |

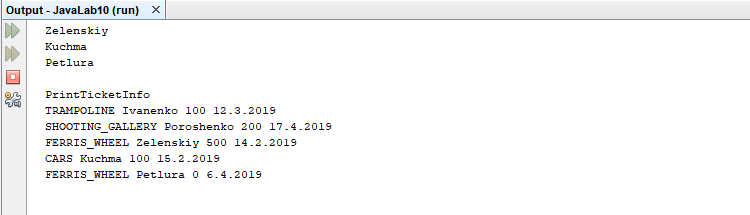
**Висновок**

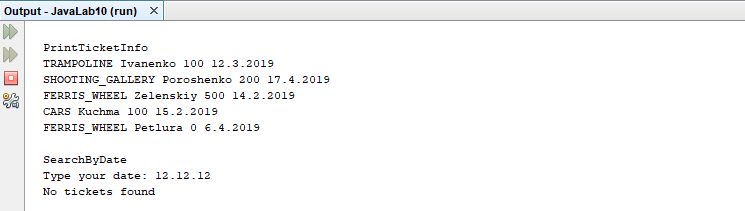
В даній роботі:

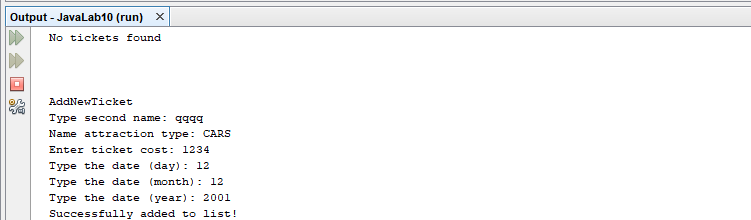
1. Реалізовано роботу в інтерактивному режимі з використанням меню;
2. Реалізовано роботу з колекціями, зокрема операції додавання, редагування, видалення, пошук, сортування, завантаження та вивантаження даних у Excel та CSV форматах.

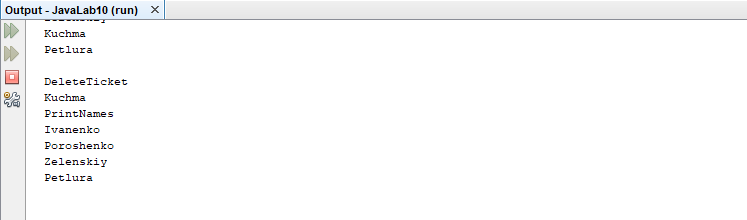


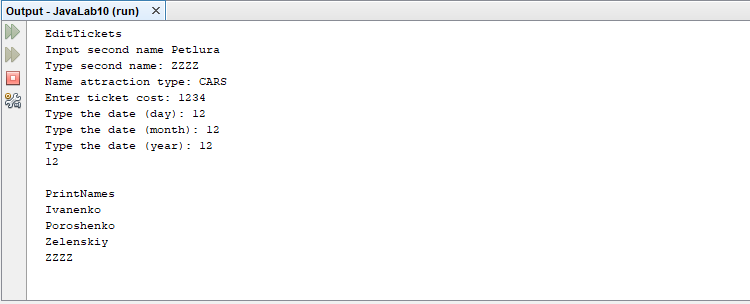




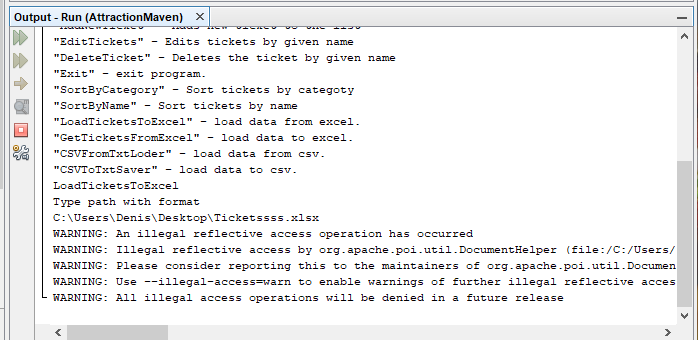


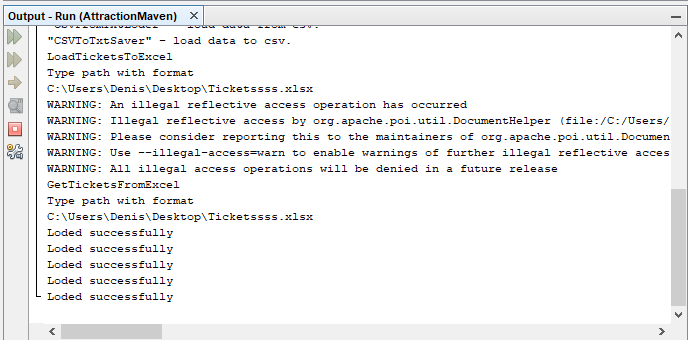




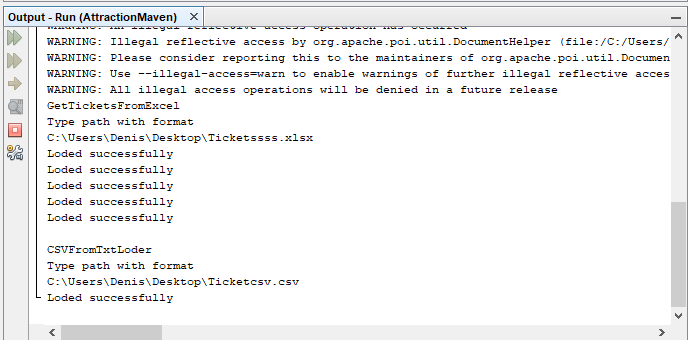












**Додаток – Лістинг програми**

package com.AttractionsMaven;

import com.AttractionsMaven.Ticket;

import java.io.File;

import java.util.\*;

public class Laba10 {

public static void printListName(ArrayList<Ticket> list) {

for(Ticket p : list) {

System.out.println(p.getSecondName());

}

}

public static void printListCategory(ArrayList<Ticket> list) {

for(Ticket p : list) {

System.out.println(p.getCategory() + " " + p.getSecondName()

+ " " + p.getCost() + " " + p.getDate());

}

}

public static void editTicket(Ticket p) {

try {

Scanner commandString = new Scanner(System.in);

System.out.print("Type second name: ");

String secondName = commandString.nextLine();

System.out.print("Name attraction type: ");

Ticket.AttractionType category =

Ticket.AttractionType.valueOf(commandString.nextLine());

System.out.print("Enter ticket cost: ");

int cost = commandString.nextInt();

int day, month, year;

System.out.print("Type the date (day): ");

day = commandString.nextInt();

System.out.print("Type the date (month): ");

month = commandString.nextInt();

System.out.print("Type the date (year): ");

year = commandString.nextInt();

p.setSecondName(secondName);

p.setCategory(category);

p.setCost(cost);

p.setDate(day, month, year);

}

catch (Exception e) {

System.out.println("Please \"AddNewTicket\""

+ e.getMessage());

}

}

public static void main(String[] args) {

ArrayList<Ticket> client = new ArrayList<Ticket>();

client.add(new Ticket("Ivanenko", Ticket.AttractionType.TRAMPOLINE,

100, new GregorianCalendar(2019, 3, 12)));

client.add(new Ticket("Poroshenko", Ticket.AttractionType.SHOOTING\_GALLERY,

200, new GregorianCalendar(2019, 4, 17)));

client.add(new Ticket("Zelenskiy",

Ticket.AttractionType.FERRIS\_WHEEL,

500, new GregorianCalendar(2019, 2, 14)));

client.add(new Ticket("Kuchma",

Ticket.AttractionType.CARS,

100, new GregorianCalendar(2019, 2, 15)));

client.add(new Ticket("Petlura",

Ticket.AttractionType.FERRIS\_WHEEL,

0, new GregorianCalendar(2019, 4, 6)));

System.out.println("Type in \"Help\" to display the command list");

String command = "";

Scanner commandString = new Scanner(System.in);

do {

command = commandString.nextLine();

switch(command) {

case "PrintNames":

printListName(client);

System.out.println();

break;

case "SortByName":

client.sort(Ticket.sortingByName);

break;

case "PrintTicketInfo":

printListCategory(client);

System.out.println();

break;

case "SortByCategory":

client.sort(Ticket.sortingByCategory);

System.out.println();

break;

case "SearchByDate":

int clientsFound = 0;

System.out.print("Type your date: ");

command = commandString.nextLine();

for(Ticket p : client) {

if (p.getDate().equals(command)) {

System.out.println(p.getCategory() + " "

+ p.getSecondName() + " " + p.getCost()

+ " " + p.getDate());

clientsFound++;

}

}

if (clientsFound == 0) {

System.out.println("No tickets found");

}

break;

case "AddNewTicket":

Ticket addPolis = new Ticket();

editTicket(addPolis);

client.add(addPolis);

System.out.print ("Successfully added to list!");

break;

case "EditTickets":

System.out.print("Input second name ");

String secondName = commandString.nextLine();

for(Ticket p : client) {

if (p.getSecondName().equals(secondName)) {

editTicket(p);

break;

}

}

break;

case "DeleteTicket":

System.out.print("");

String secondNameForDeleting = commandString.nextLine();

for(Ticket p : client) {

if (p.getSecondName().equals(secondNameForDeleting)) {

client.remove(p);

break;

}

}

break;

case "CSVFromTxtLoder":

CSVLoader loader = new CSVLoader();

System.out.println("Type path with format");

String c = commandString.nextLine();

File f = new File(c);

if (f.exists()) {

client = loader.loadTickets(c);

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

}

break;

case "CSVToTxtSaver":

CSVLoader load = new CSVLoader();

System.out.println("Type path with format");

String s = commandString.nextLine();

load.saveTickets(s, client);

break;

case "LoadTicketsToExcel":

ExcelLoader save = new ExcelLoader();

System.out.println("Type path with format");

String k = commandString.nextLine();

save.saveTickets(k, client);

break;

case "GetTicketsFromExcel":

ExcelLoader l = new ExcelLoader();

System.out.println("Type path with format");

String h = commandString.nextLine();

File g = new File(h);

if (g.exists()) {

client = l.loadTickets(h);

}

break;

case "Help":

System.out.println("\"PrintNames\" - list of clients ");

System.out.println("\"PrintTicketInfo\" - prints list of tickets ");

System.out.println("\"SearchByDate\" - searches all tickets"

+ "by the given date");

System.out.println("\"AddNewTicket\" - Adds new ticket to "

+ "the list");

System.out.println("\"EditTickets\" - Edits tickets by "

+ "given name");

System.out.println("\"DeleteTicket\" - Deletes the ticket "

+ "by given name");

System.out.println("\"Exit\" - exit program.");

System.out.println("\"SortByCategory\" - Sort tickets by "

+ "categoty");

System.out.println("\"SortByName\" - Sort tickets by "

+ "name");

System.out.println("\"LoadTicketsToExcel\" - load data from excel.");

System.out.println("\"GetTicketsFromExcel\" - load data to excel.");

System.out.println("\"CSVFromTxtLoder\" - load data from csv.");

System.out.println("\"CSVToTxtSaver\" - load data to csv.");

break;

}

}

while(!command.equals("Exit"));

}

}

package com.AttractionsMaven;

import java.util.Calendar;

import java.util.GregorianCalendar;

import java.util.Comparator;

public class Ticket {

private String secondName;

private int cost;

public enum AttractionType {

CARS, TRAMPOLINE, FERRIS\_WHEEL, ROOM\_OF\_FEAR, SHOOTING\_GALLERY, CAROUSEL;

}

AttractionType category;

GregorianCalendar ticketDate = new GregorianCalendar();

public static Comparator<Ticket> sortingByName =

new Comparator<Ticket>() {

@Override

public int compare(Ticket ticket,

Ticket ticket2) {

return ticket.secondName.compareTo(ticket2.secondName);

}

};

public static Comparator<Ticket> sortingByCategory =

new Comparator<Ticket>() {

@Override

public int compare(Ticket category1,

Ticket category2) {

return category1.category.toString().compareTo

(category2.category.toString());

}

};

public Ticket(String secondName, AttractionType category, int cost,

GregorianCalendar ticketDate) {

this.secondName = secondName;

this.cost = cost;

this.category = category;

this.ticketDate = ticketDate;

}

public Ticket() {

this.secondName = "";

this.cost = 0;

this.category = AttractionType.CAROUSEL;

this.ticketDate = new GregorianCalendar();

}

public String getSecondName() {

return secondName;

}

public int getCost() {

return cost;

}

public AttractionType getCategory() {

return category;

}

public String getDate() {

return ticketDate.get(Calendar.DAY\_OF\_MONTH) + "."

+ ticketDate.get(Calendar.MONTH) + "."

+ ticketDate.get(Calendar.YEAR);

}

public Calendar getCalendar (){

return ticketDate;

}

public void setSecondName(String secondName) {

this.secondName = secondName;

}

public void setCost(int cost) {

this.cost = cost;

}

public void setCategory(AttractionType category) {

this.category = category;

}

public void setDate(int day, int month, int year) {

ticketDate.set(year, month, day);

}

}

package com.AttractionsMaven;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.IOException;

import java.util.ArrayList;

import java.util.Iterator;

import java.util.List;

import java.util.GregorianCalendar;

import java.util.Date;

import java.io.FileOutputStream;

import org.apache.poi.ss.usermodel.CellStyle;

import org.apache.poi.ss.usermodel.CreationHelper;

import java.text.SimpleDateFormat;

import org.apache.poi.ss.usermodel.Cell;

import org.apache.poi.ss.usermodel.DataFormatter;

import org.apache.poi.ss.usermodel.Row;

import org.apache.poi.ss.usermodel.Sheet;

import org.apache.poi.ss.usermodel.Workbook;

import org.apache.poi.xssf.usermodel.XSSFWorkbook;

public class ExcelLoader {

public ArrayList<Ticket> loadTickets(String sourceName) {

ArrayList<Ticket> clientsList = new ArrayList<>();

try {

FileInputStream inputStream =

new FileInputStream(new File(sourceName));

Workbook workbook = new XSSFWorkbook(inputStream);

Sheet firstSheet = workbook.getSheetAt(0);

Iterator<Row> iterator = firstSheet.iterator();

SimpleDateFormat formatter = new SimpleDateFormat("dd.MM.yyyy");

while (iterator.hasNext()) {

Row line = iterator.next();

Iterator<Cell> cellIterator = line.cellIterator();

Cell cell = cellIterator.next();

String secondName = cell.getRichStringCellValue().getString();

cell = cellIterator.next();

Ticket.AttractionType type =

Ticket.AttractionType.valueOf

(cell.getRichStringCellValue().getString());

cell = cellIterator.next();

int cost = (int) cell.getNumericCellValue();

cell = cellIterator.next();

Date date = cell.getDateCellValue();

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

GregorianCalendar calendar = new GregorianCalendar();

calendar.setTime(date);

calendar.add(calendar.MONTH, 1);

clientsList.add(new Ticket(secondName, type, cost,

calendar));

}

workbook.close();

inputStream.close();

} catch (FileNotFoundException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}

return clientsList;

}

public void saveTickets(String sourceName, List<Ticket> client) {

try {

FileOutputStream saver = new FileOutputStream(sourceName);

Workbook workbook = new XSSFWorkbook();

Sheet listok = workbook.createSheet("Tickets");

CreationHelper createHelper = workbook.getCreationHelper();

CellStyle dateCellStyle = workbook.createCellStyle();

dateCellStyle.setDataFormat(createHelper.createDataFormat().

getFormat("dd.MM.yyyy"));

int i = 0;

for (Ticket ticket: client) {

Row stroka = listok.createRow(i);

Cell cell = stroka.createCell(0);

cell.setCellValue(ticket.getSecondName());

cell = stroka.createCell(1);

cell.setCellValue(ticket.getCategory().toString());

cell = stroka.createCell(2);

cell.setCellValue((double)ticket.getCost());

cell = stroka.createCell(3);

cell.setCellValue(ticket.getCalendar());

cell.setCellStyle(dateCellStyle);

i++;

}

workbook.write(saver);

saver.close();

}

catch(Exception e) {

}

}

}

package com.AttractionsMaven;

import com.opencsv.\*;

import java.io.FileReader;

import java.io.FileWriter;

import java.text.DateFormat;

import java.text.SimpleDateFormat;

import java.util.ArrayList;

import java.util.Date;

import java.util.GregorianCalendar;

import java.util.List;

import java.util.Locale;

/\*\*

\*

\* @author Dima

\*/

public class CSVLoader {

public ArrayList<Ticket> loadTickets(String sourceName) {

ArrayList<Ticket> polisesList = new ArrayList<>();

try {

FileReader filereader = new FileReader(sourceName);

CSVReader reader = new CSVReader(filereader);

DateFormat df = new SimpleDateFormat("dd.MM.yyyy");

String[] nextLine;

while ((nextLine = reader.readNext()) != null) {

// nextLine[] is an array of values from the line

String secondName = nextLine[0];

Ticket.AttractionType type = Ticket.AttractionType.valueOf

(nextLine[1]);

int cost = Integer.parseInt(nextLine[2]);

String ticketExtradition = nextLine[3];

GregorianCalendar iaCalendar = new GregorianCalendar();

Date ticketDate = df.parse(ticketExtradition);

iaCalendar.setGregorianChange(ticketDate);

Ticket ticket = new Ticket(secondName, type, cost,

iaCalendar);

polisesList.add(ticket);

}

} catch (Exception e) {

System.err.println("Error occured while loading " + e.getMessage());

return null;

}

return polisesList;

}

public void saveTickets(String sourceName, ArrayList<Ticket> client) {

try {

CSVWriter writer = new CSVWriter(new FileWriter(sourceName));

DateFormat df = new SimpleDateFormat("dd.MM.yyyy");

for (Ticket ticket : client) {

String[] entity = new String[4];

entity[0] = ticket.getSecondName();

entity[1] = ticket.getCategory().toString();

entity[2] = Integer.toString(ticket.getCost());

entity[3] = ticket.getDate();

writer.writeNext(entity);

}

writer.close();

}

catch (Exception m) {

System.err.println("Error occured while saving " + m.getMessage());

}

}

}