Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Інженерно-хімічний факультетКафедра «Автоматизації хімічних виробництв»

Звіт з лабораторних робіт

Програмування-2. Об'єктно-орієнтоване програмування

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

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

Київ 2020

**ЗМІСТ**

1. Лабораторна робота №2 «Типи даних, оператори, конструкції»……….3
2. Лабораторна робота №3 «Основи роботи з об’єктами»…………………6
3. Лабораторна робота №4 «Реалізація принципів ООП»………………...10
4. Лабораторна робота №5 «Робота з масивами об’єктів»………………..17
5. Лабораторна робота №6 «Вкладені класи»……………………………..21
6. Лабораторна робота №7 «Поліморфізм»………………………………..27
7. Лабораторна робота №8 «Exceptions»…………………………………...35
8. Лабораторна робота №9 «Інтерфейси»………………………………….40
9. Лабораторна робота №10 «Родові типи та колекції»…………………...47

**Лабораторна робота №2**

**«Типи даних, оператори, конструктори»**

**Мета роботи –** ознайомитися з основними типами даних, операціями і конструкціями мови, написати програму для виконання арифметичних операцій.

**Завдання**

1. Написати програму «Калькулятор», яка виконує найпростіші арифметичні операції – додавання, віднімання, множення, порівняння.

2. Кожну арифметичну операцію оформити у вигляді окремого методу

3. Задати константи для кожної з арифметичних операцій.

4. Викликати відповідну функцію, використовуючи інструкцію switch, яка приймає на вхід змінну зі значенням арифметичної операції.

Class JavaLR2 – Калькулятор

|  |  |
| --- | --- |
| int doOperation=2; | Номер операції, яка буде виконуватися |
| static int getSuma(int a, int b) | Отримати суму чисел |
| static int getDobutok(int a, int b) | Отримати добуток чисел |
| static int getRiznitsia(int a, int b) | Отримати різницю чисел |
| static int getChastka(int a, int b) | Отримати частку чисел |
| static boolean getPorivniannia(int a, int b) | Порівняти числа |



**Висновки:**

1. Реалізовано виконання найпростіших арифметичних функцій;
2. Кожна арифметична операція оформлена у вигляді окремого методу;
3. Задано константи для кожної з арифметичних операцій;
4. Викликано відповідну функцію, використовуючи інструкцію switch, яка приймає на вхід змінну зі значенням арифметичної операції.

**Лістинг програми**

package javalr2;

public class JavaLR2 {

public static void main(String[] args) {

int a=10;

int b=2;

int doOperation=2;

switch(doOperation){

case 1:

System.out.println("Suma="+getSuma(a, b));

break;

case 2:

System.out.println("Dobutok="+getDobutok(a, b));

break;

case 3:

System.out.println("Riznitsia="+getRiznitsia(a, b));

break;

case 4:

System.out.println("Chastka="+getChastka(a, b));

break;

case 5:

System.out.println("Porivniannia:"+getPorivniannia(a, b));

break;

}

}

static int getSuma(int a, int b) {

return a+b;

}

static int getDobutok(int a, int b) {

int k=0;

for (int i=1; i<=b; i++)

k=k+a;

return k;

}

static int getRiznitsia(int a, int b) {

return a-b;

}

static int getChastka(int a, int b) {

return a/b;

}

static boolean getPorivniannia(int a, int b) {

if (a==b){

System.out.println("a i b rivni.");

return a==b;

}

else{

System.out.println("a i b ne rivni");

return a!=b;

}

}

}

**Лабораторна робота №3**

**«Основи роботи з об’єктами»**

**Мета роботи –** навчитися виконувати опис класів, створювати об’єкти, викликати методи об’єктів.

**Завдання**

1. Написати клас, який містить інформацію про країну з такими атрибутами: назва, столиця, населення, грошова одиниця.

2. Створити два об’єкти даного класу, ініціалізувати їх поля.

3. Вивести на екран країну, що має більше населення.

4. Виконати присвоєння об’єктів (першій країні присвоїти другу)

5. Змінити значення довільного поля в другому об’єкті

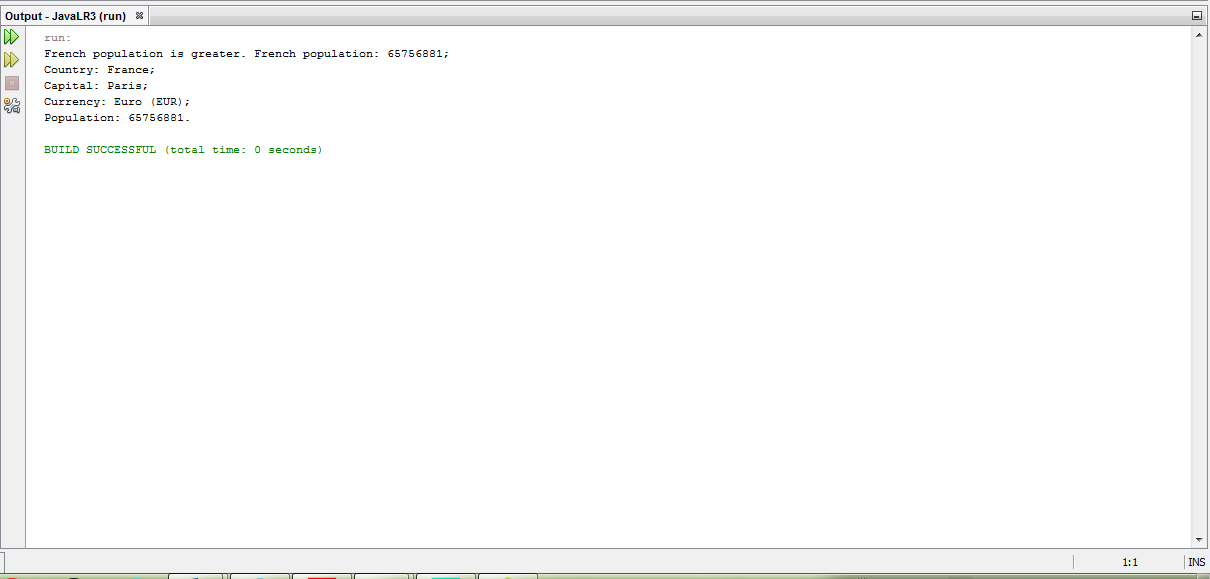
6. Вивести значення полів першого об’єкту

ClassJavaLR3 – менеджер роботи з країнами.

|  |  |
| --- | --- |
| Country Ukraine=new Country(); | Україна |
| Country France=new Country(); | Франція |

Class Country – набір даних країн.

|  |  |
| --- | --- |
| String title; | Назва |
| String capital; | Столиця |
| String currency; | Валюта |
| int population; | К-ть населення |



**Висновки:**

1. Реалізовано клас, який містить інформацію про країну з такими атрибутами: назва, столиця, населення, грошова одиниця.
2. Створено два об’єкти даного класу, ініціалізовано їх поля.
3. Виведено на екран країну, що має більше населення.
4. Виконано присвоєння об’єктів (першій країні присвоєно другу)
5. Змінено значення довільного поля в другому об’єкті
6. Вивести значення полів першого об’єкту

**Лістинг програми**

package javalr3;

public class JavaLR3 {

public static void main(String[] args){

Country Ukraine=new Country();

Country France=new Country();

Ukraine.title="Ukraine";

Ukraine.capital="Kyiv";

Ukraine.currency="Hryvnia (UAH)";

Ukraine.population=37000000;

France.title="France";

France.capital="Paris";

France.currency="Euro (EUR)";

France.population=65756881;

if (Ukraine.population>France.population)

System.out.println("Ukrainian population is greater. "

+ "Ukrainian population: "+Ukraine.population+";");

else

System.out.println("French population is greater. "

+ "French population: "+France.population+";");

Ukraine=France;

Ukraine.printInfo();

}

}

package javalr3;

public class Country {

String title;

String capital;

String currency;

int population;

public void printInfo() {

System.out.println("Country: "+title+";"+"\n"+

"Capital: "+capital+";"+"\n"+

"Currency: "+currency+";"+"\n"+

"Population: "+population+"."+"\n");

}

}

**Лабораторна робота №4**

«**Реалізація принципів ООП»**

**Мета роботи –** навчитися виконувати інкапсуляцію та наслідування класів.

**Завдання**

1. Написати базовий клас «Людина», який містить інформацію про ім’я та прізвище. Клас повинен містити конструктор.

2. Створити клас «Студент», який наслідує базовий клас і містить інформацію про ЗНО, ФДП, АТЕСТАТ та РЕЙТИНГ студента.

3. В класі студент реалізувати два перевантажених конструктора – з ФДП та без ФДП

4. Створити декілька об’єктів класу студент.

5. Розрахувати і вивести рейтинг кожного студента

ClassJavaLR4 – менеджер роботи з системою студентів

|  |  |
| --- | --- |
| StudentI Ivanov = new StudentI (200, 1, 12, "Ivan", "Ivanov"); | Студент Іванов |
| StudentP Petrenko = new StudentP (20, 12, "Petro", "Petrenko"); | Студент Петренко |

Class People – набір даних людини

|  |  |
| --- | --- |
| private String name; | Ім’я людини |
| private String secondName; | Прізвище людини |

ClassStudentI – набір даних студента Іванова

|  |  |
| --- | --- |
| private int rating; | Рейтинг Іванова |
| private int ZNO; | Бал ЗНО Іванова |
| private int FDP; | ФДП Іванова |
| private int atestat; | Середній бал атестату Іванова |
| public boolean checkRating(int bal) | Перевірити, чи рейтинг Іванова більший за прохідний |
| public boolean checkRatingStypendia(int balStyp) | Перевірити, чи рейтинг Іванова підпадає під стипендію |

ClassStudentP – набір даних студента Петренка

|  |  |
| --- | --- |
| private int rating; | Рейтинг Петренка |
| private int ZNO; | Бал ЗНО Петренка |
| private int FDP; | ФДП Петренка |
| private int atestat; | Середній бал атестату Петренка |
| public boolean checkRating(int bal) | Перевірити, чи рейтинг Петренка більший за прохідний |
| public boolean checkRatingStypendia(int balStyp) | Перевірити, чи рейтинг Петренка підпадає під стипендію |



**Висновки:**

1. Реалізовано базовий клас «Людина», який містить інформацію про ім’я та прізвище. Клас повинен містити конструктор;
2. Реалізовано клас «Студент», який наслідує базовий клас і містить інформацію про ЗНО, ФДП, АТЕСТАТ та РЕЙТИНГ студента;
3. В класі студент реалізовано два перевантажених конструктора – з ФДП та без ФДП;
4. Створити декілька об’єктів класу студент;
5. Розраховано і виведено рейтинг кожного студента.

**Лістинг програми**

package javalr4;

public class JavaLR4 {

public static void main(String[] args) {

StudentI Ivanov = new StudentI (200, 1, 12, "Ivan", "Ivanov");

StudentP Petrenko = new StudentP (20, 12, "Petro", "Petrenko");

System.out.println(Ivanov);

System.out.println(Petrenko);

Ivanov.checkRating(60);

Petrenko.checkRating(60);

Ivanov.checkRatingStypendia(100);

Petrenko.checkRatingStypendia(100);

}

}

package javalr4;

public class People {

private String name;

private String secondName;

public String getName() {

return name;

}

public String getSecondName() {

return secondName;

}

public void setName(String name) {

this.name = name;

}

public void setSecondName(String secondName) {

this.secondName = secondName;

}

public People(String name, String fio) {

this.name = name;

this.secondName = fio;

}

}

package javalr4;

public class StudentI extends People{

private int rating;

private int ZNO;

private int FDP;

private int atestat;

public StudentI(int ZNO, int FDP, int atestat, String name, String fio) {

super(name, fio);

this.ZNO = ZNO;

this.FDP = FDP;

this.atestat = atestat;

rating = ZNO + FDP + atestat;

}

public boolean checkRating(int bal) {

if (rating>bal) {

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

return true;

}

else {

System.out.println("Ivanov ne sdal.");

return false;

}

}

public boolean checkRatingStypendia(int balStyp) {

if (rating>balStyp) {

System.out.println("Ivanov na stypendii.");

return true;

}

else {

System.out.println("Ivanov ne na stypendii.");

return false;

}

}

@Override

public String toString() {

return ("Rating Ivanova: "+rating);

}

public int getRating() {

return rating;

}

public void setRating(int rating) {

this.rating = rating;

}

public int getZNO() {

return ZNO;

}

public void setZNO(int ZNO) {

this.ZNO = ZNO;

}

public int getFDP() {

return FDP;

}

public void setFDP(int FDP) {

this.FDP = FDP;

}

public int getAtestat() {

return atestat;

}

public void setAtestat(int atestat) {

this.atestat = atestat;

}

}

package javalr4;

public class StudentP extends People{

private int rating;

private int ZNO;

private int atestat;

public StudentP(int ZNO, int atestat, String name, String secondName) {

super(name, secondName);

this.ZNO = ZNO;

this.atestat = atestat;

rating = ZNO + atestat;

}

public boolean checkRating(int bal) {

if (rating>bal){

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

return true;

}

else {

System.out.println("Petrenko ne sdal.");

return false;

}

}

public boolean checkRatingStypendia(int balStyp) {

if (rating>balStyp) {

System.out.println("Petrenko na stypendii.");

return true;

}

else {

System.out.println("Petrenko ne na stypendii.");

return false;

}

}

@Override

public String toString() {

return ("Rating Petrenko: "+rating);

}

public int getRating() {

return rating;

}

public void setRating(int rating) {

this.rating = rating;

}

public int getZNO() {

return ZNO;

}

public void setZNO(int ZNO) {

this.ZNO = ZNO;

}

public int getAtestat() {

return atestat;

}

public void setAtestat(int atestat) {

this.atestat = atestat;

}

}

**Лабораторна робота №5**

**«Робота з масивами об’єктів»**

**Мета роботи –** навчитисяпрацювати з масивом об’єктів та з статичними членами класів.

**Завдання**

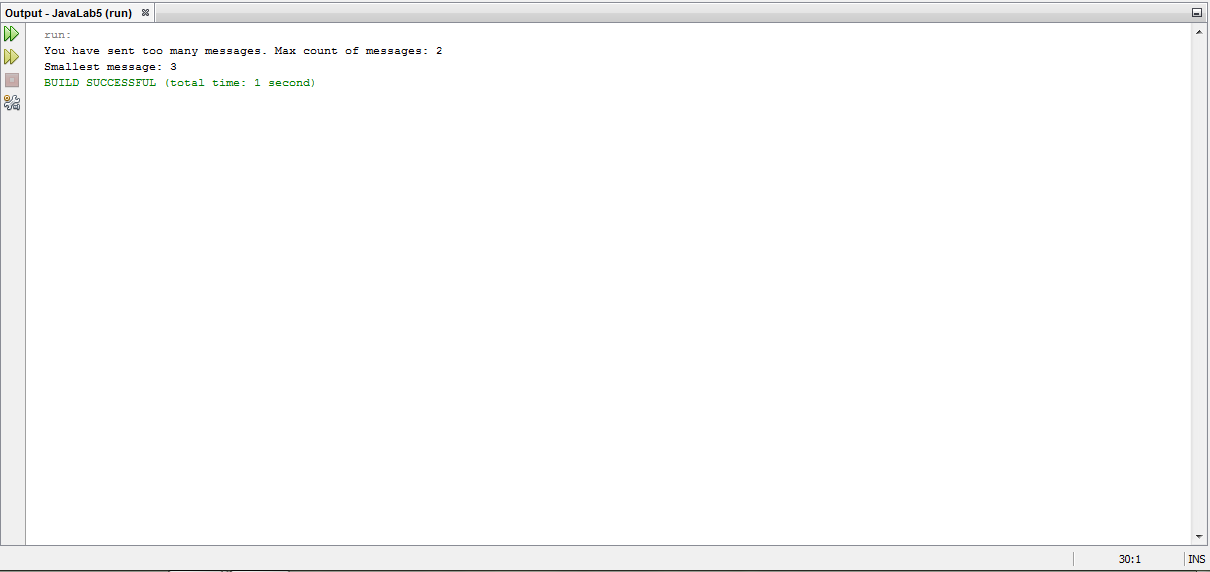
1. Створити клас «Туристична фірма», який містить наступні атрибути: кількість символів в темі електронного повідомлення із запрошенням на курси, кількість символів в описі електронного повідомлення із запрошенням на курси, кількість картинок в електронному повідомленні, ідентифікатор повідомлення. Клас також містить статичне поле, для підрахунку кількості повідомлень.
2. Створити масив об’єктів типу «Повідомлення». Можна даний масив оформити у вигляді окремого класу.
3. Створити декілька об’єктів класу «Туристична фірма», і записати їх в масив.
4. Знайти найменше повідомлення.
5. Перевірити кількість існуючих повідомлень. Якщо це значення перевищує якесь граничне значення – вивести повідомлення.

TourCompany– менеджер роботи з повідомленнями.

|  |  |
| --- | --- |
| Message[] AllMessages = {ticket1, ticket2, ticket3, ticket4}; | Масив зі всіма білетами. |
| public static void smallestMessage(Message[] i) | Визначити найменше повідомлення. |

ClassMessage – містить набір даних електронного повідомлення

|  |  |
| --- | --- |
| private int AbroadTicketsRequest; | Кількість білетів. |
| private int issuedById; | Ким видан |
| private int ticketId; | ИД білета |
| private int planeId; | Ідентифікатор повідомлення. |
| private int ticketRequestCounter;  countMessages(Message[] email) | Визначити максимальну можливу кількість білетів. |



**Висновки:**

1. Реалізовано клас «Туристична фірма», який містить наступні атрибути: кількість символів в темі електронного повідомлення із запрошенням на курси, кількість символів в описі електронного повідомлення із запрошенням на курси, кількість картинок в електронному повідомленні, ідентифікатор повідомлення. Клас також містить статичне поле, для підрахунку кількості повідомлень.
2. Створено масив об’єктів типу «Туристична фірма».
3. Створено декілька об’єктів класу «Туристична фірма», і записано в масив.
4. Реалізовано знаходження найменшого повідомлення.
5. Реалізовано перевірку кількості існуючих повідомлень. Якщо це значення перевищує якесь граничне значення – виводиться повідомлення.

**Лістинг програми**

public class AbroadTicketsRequest {

private int ticketRequestCounter;

private int issuedById;

private int planeId;

private int ticketId;

public AbroadTicketsRequest(int ticketRequestCounter, int issuedById,

int planeId, int Id) {

this.ticketRequestCounter = ticketRequestCounter;

this.issuedById = issuedById;

this.planeId = planeId;

this.ticketId = Id;

}

public static void countMessages(AbroadTicketsRequest[] email) {

int maxMessages = 4;

if (email.length>maxMessages) {

System.out.println("Too many messages have been sent."

+ " Maximum messages queue: "+maxMessages);

}

}

public int getTicketRequestCounter() {

return ticketRequestCounter;

}

public int getIssuedById() {

return issuedById;

}

public int getPlaneId() {

return planeId;

}

public int getID() {

return ticketId;

}

}

**Лабораторна робота №6**

**«Вкладені класи»**

**Мета роботи –**навчитисяоголошувати і використовувативкладені класи.

**Завдання**

1. Написати клас «Студент», який містить наступні поля: прізвище, середній бал, кількість пропусків, вік, стать, ознаку отримання стипендії та ознаку необхідності військового обліку.
2. В класі «Студент» написати вкладений класдля обчислення стипендії. Стипендія дається студентам в яких середній бал більше 4,5 та кількість пропусків менша 10.
3. В класі «Студент» написати локальний класдля визначення військовозобов’язаних студентів. Використати його в конструкторі. Військовозобов’язаними вважати чоловіків старших 18 років.
4. Створити декілька об’єктів класу студент.

Розрахувати і вивести характеристики кожного студента

ClassJavaLR6 – менеджер роботи з даними студентів

|  |  |
| --- | --- |
| Student Ftemov = new Student("Ftemov", 18, Sex.MAN, 2, 8.2); | Студент Фтемов |
| Student Sokhatska = new Student("Sokhatska", 18, Sex.WOMAN, 0, 10); | Студентка Сохацька |
| Student Kryvenko = new Student("Kryvenko", 17, Sex.MAN, 20, 2); | Студент Кривенко |

Class Student – набір даних студента

|  |  |
| --- | --- |
| private Sex sex; | Стать; |
| private String secondName; | Прізвище; |
| private int age; | Вік; |
| private int missedLessons; | Пропуски; |
| private double avarageMark; | Середній бал; |
| boolean scolarship; | Можливість отримувати стипендію; |
| boolean army; | Можливість служити в армії. |

ClassScholarship–містить дані про можливість отримання стипендії.

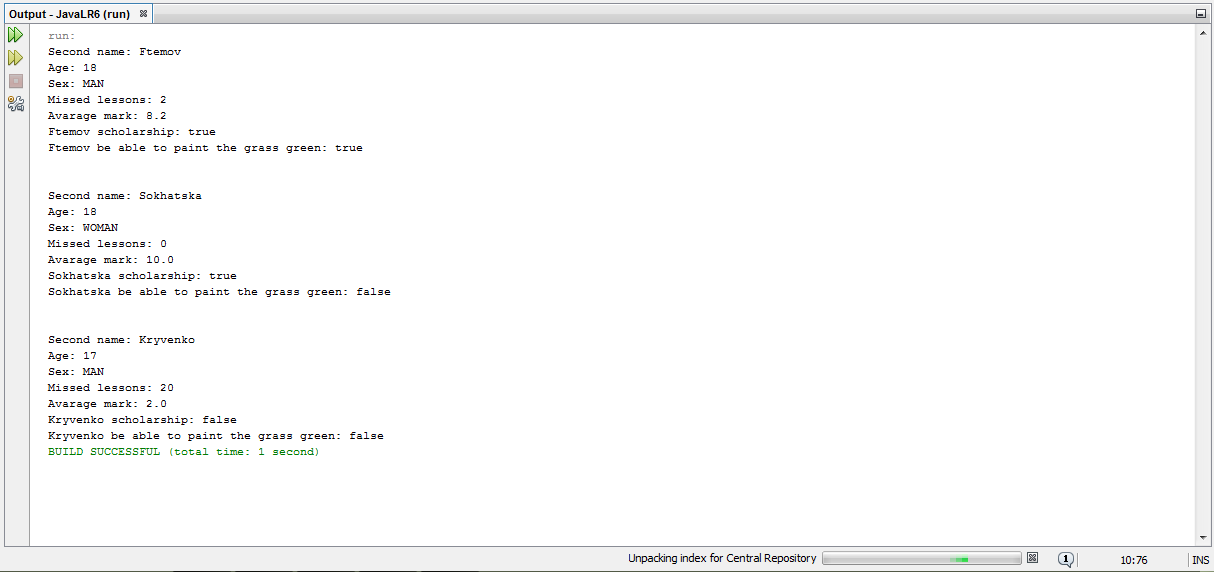
|  |  |
| --- | --- |
| private final int MAX\_MISSED\_LESSONS | Кількість максимальних пропусків для можливості отримання стипендії; |
| private final double MIN\_AVARAGE\_MARK | Мінімальний середній бал для можливості отримання стипендії; |
| public void calculateScolarship() | Розрахунок можливості отримання стипендії. |

ClassArmy – містить дані про можливість служити в армії.

|  |  |
| --- | --- |
| public void checkGrassForPaint() | Розрахувати можливість служити в армії. |

ClassSex – містить інформацію про стать студента.

|  |  |
| --- | --- |
| public enum Sex | Стать студента. |



**Висновки:**

1. Релізовано клас «Студент», який містить наступні поля: прізвище, середній бал, кількість пропусків, вік, стать, ознаку отримання стипендії та ознаку необхідності військового обліку.
2. Написано вкладений класв класі «Студент» для обчислення стипендії. Стипендія дається студентам в яких середній бал більше 4,5 та кількість пропусків менша 10.
3. В класі «Студент» написано локальний класдля визначення військовозобов’язаних студентів. Використано його в конструкторі. Створено декілька об’єктів класу студент.

Реалізовано виведення характеристики кожного студента

**Лістинг програми**

package javalr6;

public class JavaLR6 {

public static void main(String[] args) {

Student Ftemov = new Student("Ftemov", 18, Sex.MAN, 2, 8.2);

Ftemov.Data();

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

Student Sokhatska = new Student("Sokhatska", 18, Sex.WOMAN, 0, 10);

Sokhatska.Data();

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

Student Kryvenko = new Student("Kryvenko", 17, Sex.MAN, 20, 2);

Kryvenko.Data();

}

}

package javalr6;

public enum Sex {

MAN, WOMAN

}

package javalr6;

public class Student {

private Sex sex;

private String secondName;

private int age;

private int missedLessons;

private double avarageMark;

boolean scolarship;

boolean army;

class Scolarship {

private final int MAX\_MISSED\_LESSONS = 15;

private final double MIN\_AVARAGE\_MARK = 5D;

public void calculateScolarship() {

scolarship = (missedLessons < MAX\_MISSED\_LESSONS) && (avarageMark >= MIN\_AVARAGE\_MARK);

}

}

public Student(String secondName, int age, Sex sex, int missedLessons, double raiting) {

this.secondName = secondName;

this.age = age;

this.sex = sex;

this.missedLessons = missedLessons;

this.avarageMark = raiting;

class Army {

public void checkGrassForPaint() {

army = age >= 18 && sex == Sex.MAN;

}

}

new Army().checkGrassForPaint();

new Scolarship().calculateScolarship();

}

public void Data() {

System.out.println("Second name: " + secondName + '\n' +

"Age: " + age + '\n' +

"Sex: " + sex + '\n' +

"Missed lessons: " + missedLessons + '\n' +

"Avarage mark: " + avarageMark + '\n' +

secondName + " scholarship: " + scolarship + '\n' +

secondName + " be able to paint the grass green: " + army);

}

}

**Лабораторна робота №7**

**«Поліморфізм»**

**Мета роботи –** навчитисявикористовувати інтерфейси та абстрактні класи для реалізації поліморфізму.

**Завдання**

1. Написати абстрактний клас «Документ», який містить поля «назва», «автор», конструктор і абстрактний метод для друку цих полів.
2. Створити дочірні класи «Книга» та «Стаття», з власною реалізацією методу для друку.
3. Створити масив об’єктів класу «Документ», який містить декілька статей та книг.
4. В циклі по масиву вивести інформацію про видання.
5. Створити ієрархію класів робітників компанії з погодинною оплатою праці та фіксованою ставкою з премією. В кожному з класів реалізувати по своєму метод нарахування зарплати. Створити декілька об’єктів цих класів, в циклі розрахувати заробітну плату для всіх працівників компанії

publicclassJavaLR7 – менеджер роботи з акторами.

|  |  |
| --- | --- |
| Workman[] workmen = new Workman[3]; | Список акторів; |
| double salary | Заробітна плата. |

public abstract class Workman – містить дані актора.

|  |  |
| --- | --- |
| private int age; | Вік; |
| private String name; | Ім’я; |
| protected double salary; | Заробітна плата. |

publicclassWorkmanForHourextendsWorkman – містить дані про актора, який працює погодинно.

|  |  |
| --- | --- |
| private double fame; | Слава актора; |
| private int hour; | Кількість годин; |
| public void Report() | Розрахувати заробітну плату актора за годину. |

class WorkmanWithSalary extends Workman – містить дані про актора, який працює щомісячно.

|  |  |
| --- | --- |
| private double salaryForWork; | Заробітна плата актора за місяць. |
| private int views; | Кількість переглядів фільму; |
| private double copyright; | Гроші за авторські права актора; |
| public void Report() | Розрахувати заробітну плату актора за місяць. |



**Висновки:**

1. Написано абстрактний клас «Документ», який містить поля «назва», «автор», конструктор і абстрактний метод для друку цих полів.
2. Створено дочірні класи «Книга» та «Стаття», з власною реалізацією методу для друку.
3. Створено масив об’єктів класу «Документ», який містить декілька статей та книг.
4. В циклі по масиву реалізовано виведення інформації про видання.
5. Створено ієрархію класів робітників компанії з погодинною оплатою праці та фіксованою ставкою з премією. В кожному з класів реалізовано по своєму метод нарахування зарплати. Створено декілька об’єктів цих класів, в циклі розраховано заробітну плату для всіх працівників компанії

**Лістинг програми**

package javalr7;

public class JavaLR7 {

public static void main(String[] args) {

Workman[] workmen = new Workman[3];

workmen[0] = new WorkmanWithSalary(500000, 1500000, 15000, 52, "Jason Statham");

workmen[1] = new WorkmanWithSalary(600000, 1500000, 16000, 55, "Keanu Reeves");

workmen[2] = new WorkmanForHour(1200000, 20, 56, "Jonny Depp");

double salary = 0;

for (Workman workman : workmen) {

workman.Report();

System.out.println("Name: " + workman.getName() + "," + " Salary: "

+ workman.getSalary());

salary += workman.getSalary();

}

System.out.println("Total salary: " + salary);

}

}

package javalr7;

public abstract class Workman {

private int age;

private String name;

protected double salary;

public Workman(int age, String name) {

this.age = age;

this.name = name;

}

public int getAge() {

return age;

}

public void setAge(int age) {

this.age = age;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public double getSalary() {

return salary;

}

public void setSalary(double salary) {

this.salary = salary;

}

public abstract void Report();

}

package javalr7;

public class WorkmanForHour extends Workman {

private double fame;

private int hour;

public WorkmanForHour(double fame, int hour, int age, String name) {

super(age, name);

this.fame = fame;

this.hour = hour;

}

@Override

public void Report() {

salary = fame \* hour;

}

public double getFame() {

return fame;

}

public void setFame(double fame) {

this.fame = fame;

}

public double getHour() {

return hour;

}

public void setHour(int hour) {

this.hour = hour;

}

}

package javalr7;

class WorkmanWithSalary extends Workman {

private double salaryForWork;

private int views;

private double copyright;

public WorkmanWithSalary(double salaryForWork, int views, double copyright,

int age, String name) {

super(age, name);

this.salaryForWork = salaryForWork;

this.views = views;

this.copyright = copyright;

}

public double getSalaryForWork() {

return salaryForWork;

}

public void setSalaryForWork(double salaryForWork) {

this.salaryForWork = salaryForWork;

}

public double getViews() {

return views;

}

public void setViews(int views) {

this.views = views;

}

public double getCopyright() {

return copyright;

}

public void setCopyright(double copyright) {

this.copyright = copyright;

}

@Override

public void Report() {

salary = salaryForWork \* views \* copyright;

}

}

**Лабораторна робота №8**

**«Exceptions»**

**Мета роботи –**навчитися оброблювати і генерувати виняткові ситуації.

**Завдання**

Написати клас з методами, які генерують наступні виняткові ситуації

1. ділення на нуль (без обробки)
2. ділення на нуль (з блоком *try … catch*).
3. з декількома блоками *catch*.
4. з виведенням опису помилки та стеку викликів
5. методи, які повертають помилку *throwable*
6. в методі, описаному як *throwable* згенерувати помилку і обробити її в головній програмі

Написати користувацький клас виключення, згенерувати і перехопити помилку даного класу.

public class DivideByZeroTryCatch – ділення на нуль і виклик 6-ї клітинки масиву з 5 через trycatch

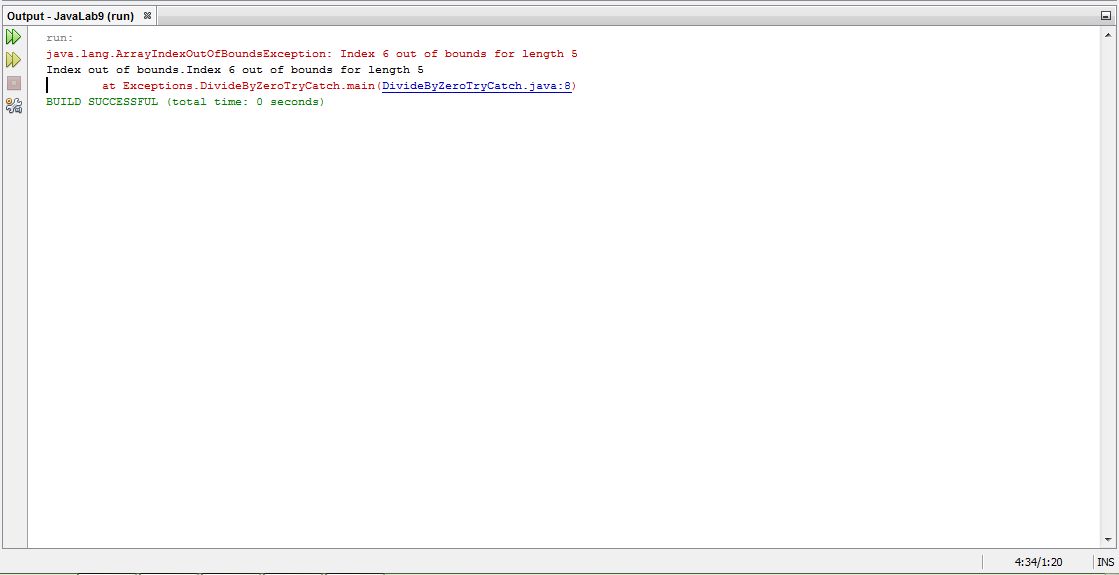
|  |  |
| --- | --- |
| int[] mas = new int[5]; | Масив на 5 клітинок; |
| int b=1/0; | Змінна, яка ділиться на нуль. |

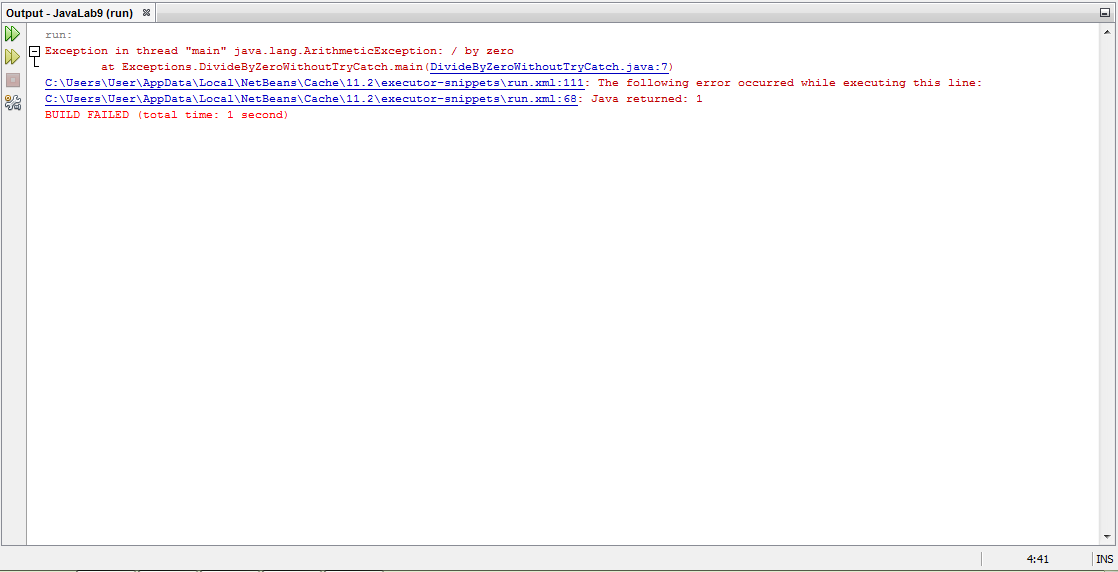
public class DivideByZeroWithoutTryCatch – ділення на нуль без обробки

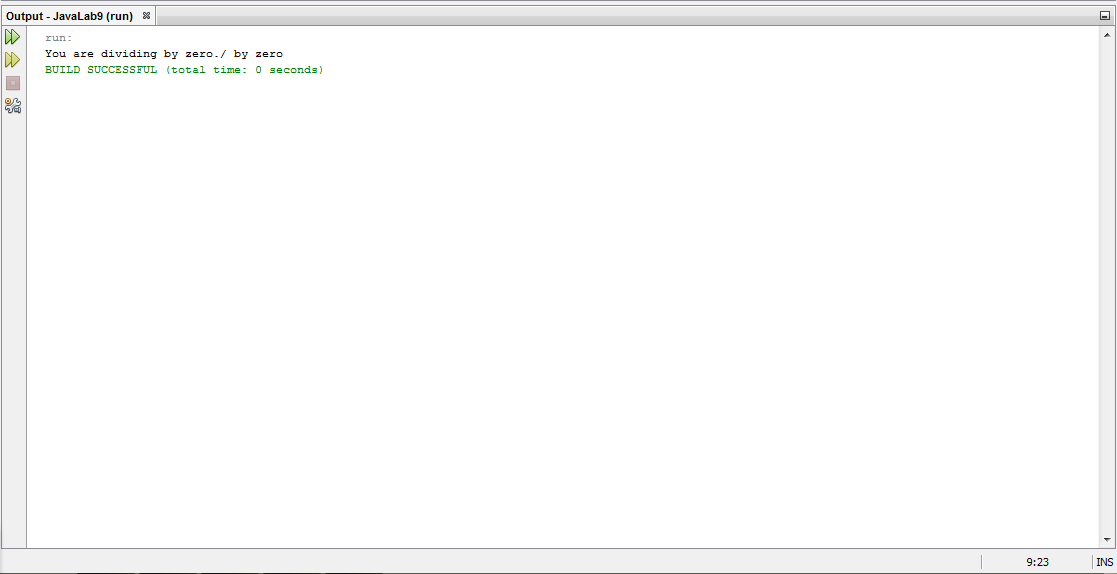
|  |  |
| --- | --- |
| int b=1/0; | Змінна, яка ділиться на нуль. |

public class Throwable – ділення на нуль з throwable

|  |  |
| --- | --- |
| public static int divide(int a, int b) throws ArithmeticException | Поділити на нуль з throwable |







**Висновки:**

1. Написано клас з методами, які генерують наступні виняткові ситуації
2. ділення на нуль (без обробки)
3. ділення на нуль (з блоком *try … catch*).
4. з декількома блоками *catch*.
5. з виведенням опису помилки та стеку викликів
6. методи, які повертають помилку *throwable*
7. в методі, описаному як *throwable* згенеровано помилку і оброблено її в головній програмі
8. Написано користувацький клас виключення, згенеровано і перехоплено помилку даного класу.

**Лістинг програми**

package laba9;

import java.lang.ArithmeticException;

public class DoubleTryCatchZero {

public static void main(String[] args) {

try {

int[] arr = new int[5];

System.out.println(arr[6]);

int k=1/0;

}

catch (ArithmeticException m) {

System.out.println("Division by zero");

}

catch (Exception k) {

System.out.println("Out of array"+k.getMessage());

k.printStackTrace();

}

}

}

package laba9;

import java.lang.NumberFormatException;

public class ErrorZero {

public static void main(String[] args) {

int k=213054/0;

}

}

package laba9;

import java.lang.ArithmeticException;

public class Method\_Throwable {

public static void main(String[] args) {

try {

division(1,0);

}

catch (ArithmeticException r) {

System.out.println(r.getMessage());

}

}

public static int division(int c, int d) throws ArithmeticException {

return c/d;

}

}

**Лабораторна робота №9**

**«Інтерфейси»**

**Мета роботи –**навчитися працювати з інтерфейсами, реалізовувати поліморфізм на основі інтерфейсів.

**Завдання**

1. Написати інтерфейс касового апарату, який реалізує наступні операції:

а) початок зміни (введення логіну оператора)

б) друк чека (виводить назву товару, ціну і кількість)

в) звіт (загальна сума продажів) за день

1. Написати 2 класи, які реалізують цей інтерфейс: Samsung, RF
2. Написати клас ARM (автоматизоване робоче місце касира), дляпродажу товарів, яка реалізує наступні функції

а) початок зміни оператора

б) підключення касового апарату

в) продаж товару (збереження товару і друк чеку)

г) звіт за день

1. Промоделювати роботу каси, по черзі підключити два касових апарати, продати 2-3 товари, вивести суму за день.

public class Cash implements Functions – менеджер роботи з касами.

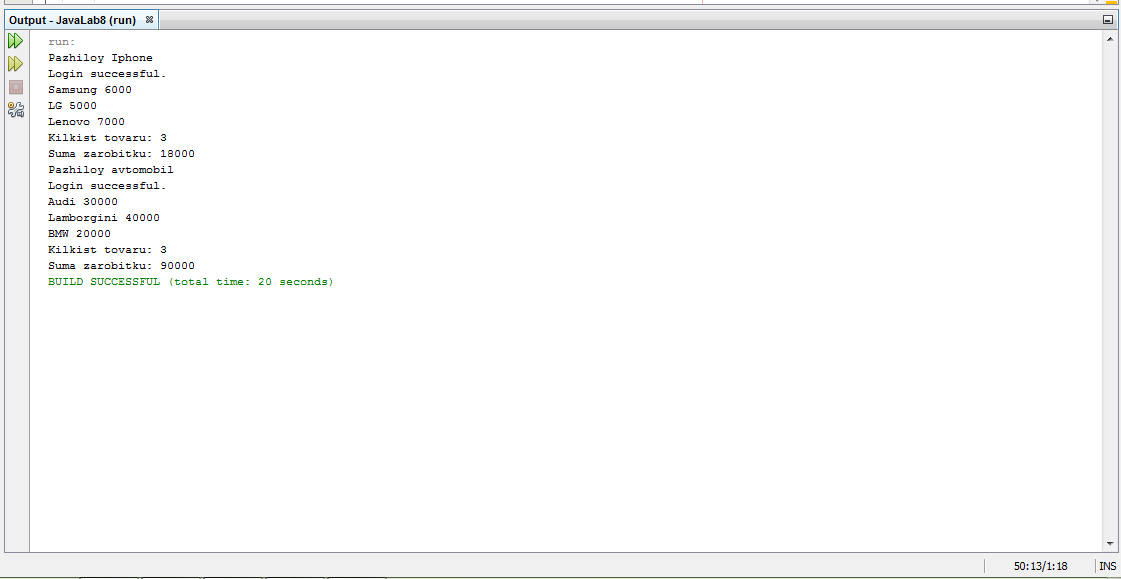
|  |  |
| --- | --- |
| String password; | Пароль для входу в систему каси; |
| Tovar[] Spisok; | Список товару; |
| public void login() | Увійти в систему каси; |
| public void printChek() | Друк чеку; |
| public void printZvit() | Друк звіту продаж; |

public interface Functions – інтерфейс з функціоналом кас

|  |  |
| --- | --- |
| void login(); | Увійти в систему каси; |
| void printChek(); | Роздрукувати чек; |
| void printZvit(); | Роздрукувати звіт продаж. |

public class Tovar – містить набір даних про товар

|  |  |
| --- | --- |
| private String name; | Назва товару; |
| private int cost; | Ціна. |



**Висновки:**

1. Написано інтерфейс касового апарату, який реалізує наступні операції:

а) початок зміни (введення логіну оператора)

б) друк чеку (виводить назву товару, ціну і кількість)

в) звіт (загальна сума продажів) за день

1. Написано 2 класи, які реалізують цей інтерфейс: Samsung, RF
2. Написано клас ARM (автоматизоване робоче місце касира), дляпродажу товарів, яка реалізує наступні функції

а) початок зміни оператора

б) підключення касового апарату

в) продаж товару (збереження товару і друк чеку)

г) звіт за день

1. Змодельованороботу каси, по черзі підключено два касових апарати, продано 2-3 товари, виведено суму за день.

**Лістинг програми**

import java.util.Scanner;

public class Cash implements Functions {

String password;

Tovar[] Spisok;

public void login(){

String l;

Scanner login = new Scanner(System.in);

l = login.nextLine();

if (l.equals(password)) {

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

}

else {

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

System.exit(0);

}

}

public void printChek() {

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

System.out.println(Spisok[i].getName()+" "+Spisok[i].getCost());

}

System.out.println("Kilkist tovaru: "+Spisok.length);

}

public void printZvit() {

int suma = 0;

for (int i = 0; i < 3; i++) {

suma += Spisok[i].getCost();

}

System.out.println ("Suma zarobitku: "+suma);

}

public Cash(Tovar[] Spisok, String password) {

this.Spisok = Spisok;

this.password = password;

}

public static void main (String[] args){

Cash phones = new Cash(new Tovar[] {

new Tovar(6000, "Samsung"), new Tovar(5000, "LG"),

new Tovar(7000, "Lenovo")

}, "Pazhiloy Iphone");

Cash cars = new Cash(new Tovar[] {

new Tovar(30000, "Audi"), new Tovar(40000, "Lamborgini"),

new Tovar(20000, "BMW")

}, "Pazhiloy avtomobil");

phones.login();

phones.printChek();

phones.printZvit();

cars.login();

cars.printChek();

cars.printZvit();

}

}

public interface Functions {

void login();

void printChek();

void printZvit();

}

public class Tovar{

private String name;

private int cost;

public Tovar(int cost, String name) {

this.cost = cost;

this.name = name;

}

public String getName() {

return name;

}

public int getCost() {

return cost;

}

}

**Лабораторна робота №10**

**«Родові типи та колекції»**

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); |

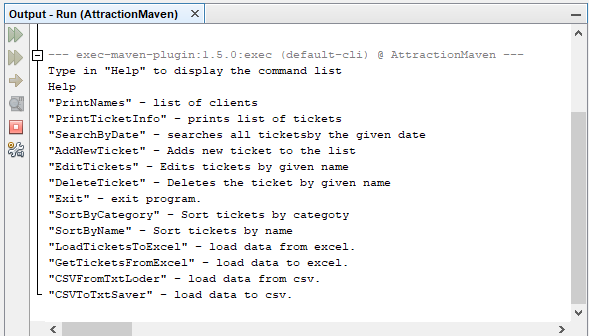
Ticket– менеджер роботи з клієнтами та полісами.

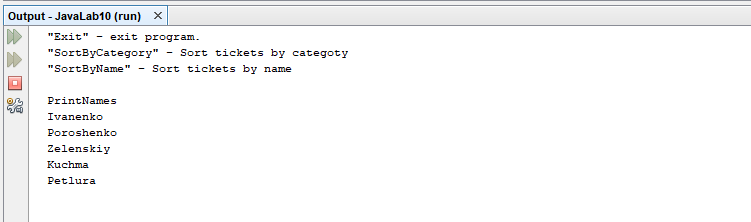
|  |  |
| --- | --- |
| public static void printListName(ArrayList<Ticket > list) | Виведення відсортованого за алфавітом списку клієнтів за прізвищем; |
| public static void printListCategory(ArrayList<Ticket> list) | Виведення відсортованого за категорією (а списку; |
| public static void editTicket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

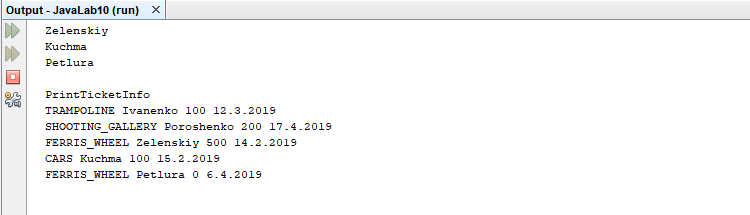
**Висновок**

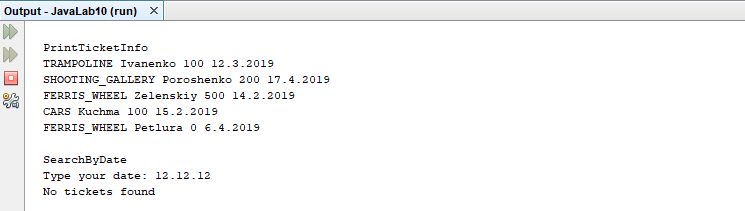
В даній роботі:

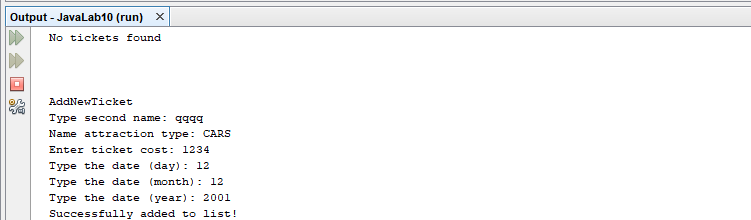
1. Реалізовано роботу в інтерактивному режимі з використанням меню;
2. Реалізовано роботу з колекціями, зокрема операції додавання, редагування, видалення, пошук, сортування, завантаження та вивантаження даних у Excel та CSV форматах.

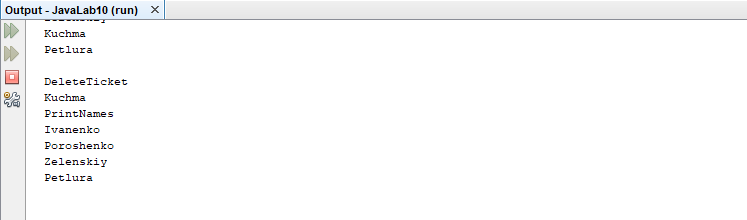


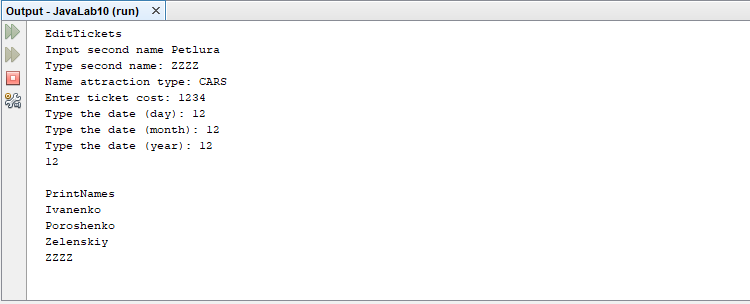




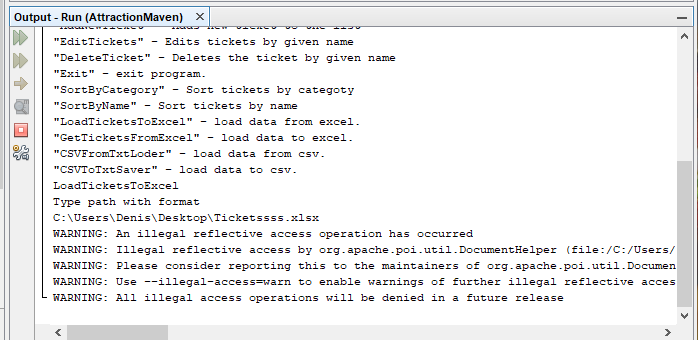


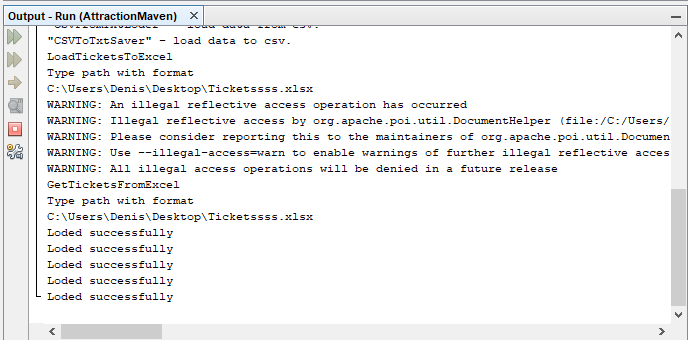




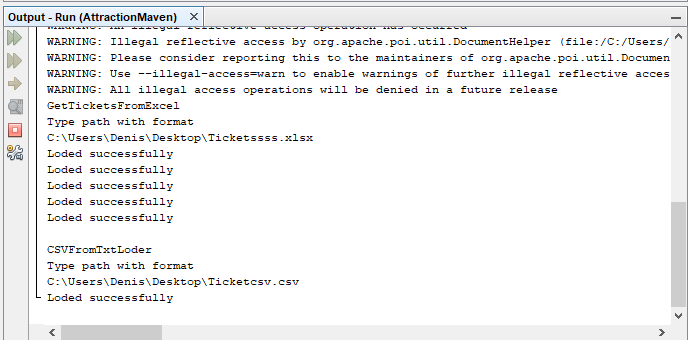












**Додаток – Лістинг програми**

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());

}

}

}