Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Інженерно-хімічний факультет

Кафедра «Технічних та програмних засобів автоматизації»

Звіт з лабораторних робіт

Програмування-2. Об'єктно-орієнтоване програмування

Виконав: студент гр. ЛА-03, Сенюк М.П.

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

Київ 2021

**ЛАБОРАТОРНА РОБОТА №1**

**Налаштування середовища JAVA**

Мета роботи – навчитися встановлювати Java-платформу (JDK), написати

елементарну програму, запустити її через IDE та за допомогою командної строки.

**Хід роботи**

1. Завантажити і встановити Java (JDK) за посиланням

<https://www.oracle.com/java/technologies/javase-jdk13-downloads.html>

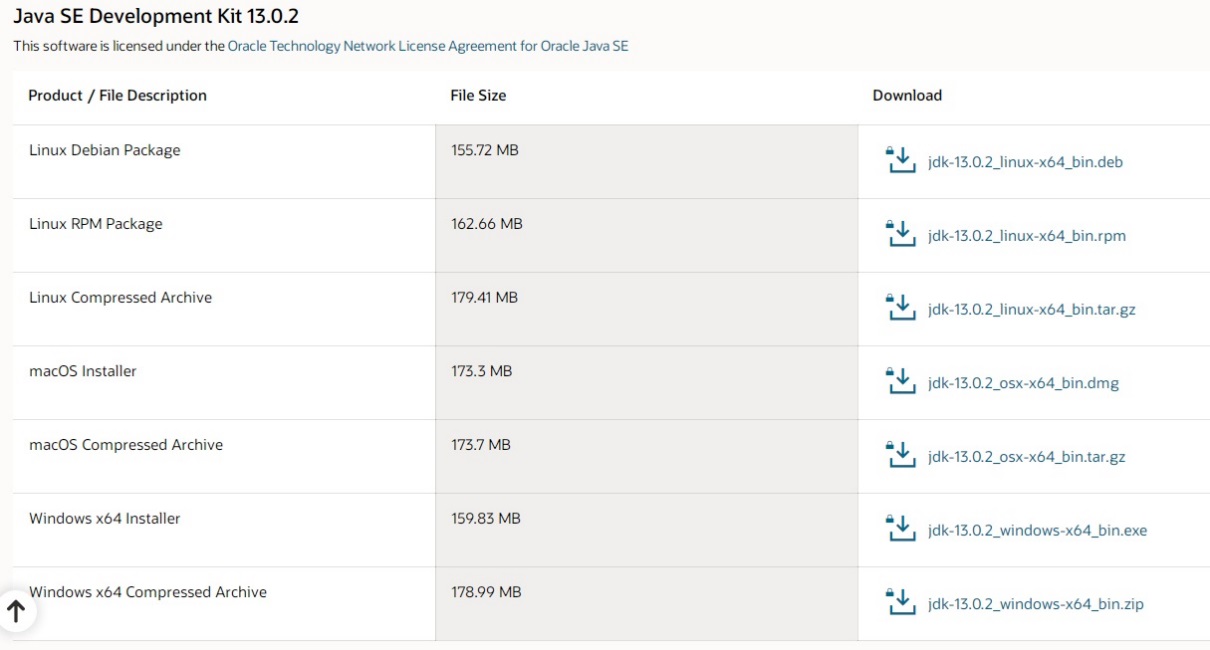


Рис. 1.1. Вибір інсталяційного пакету JDK

2. Завантажити IDE NetBeans за посиланням та встановити

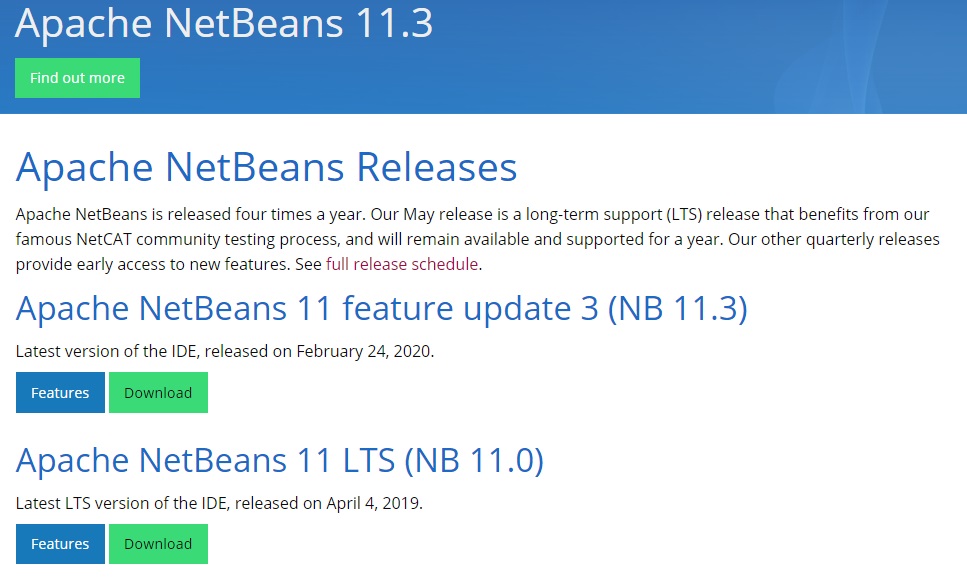
<http://netbeans.apache.org/download/index.html>

Рис. 1.2. Встановлення IDE NetBeans

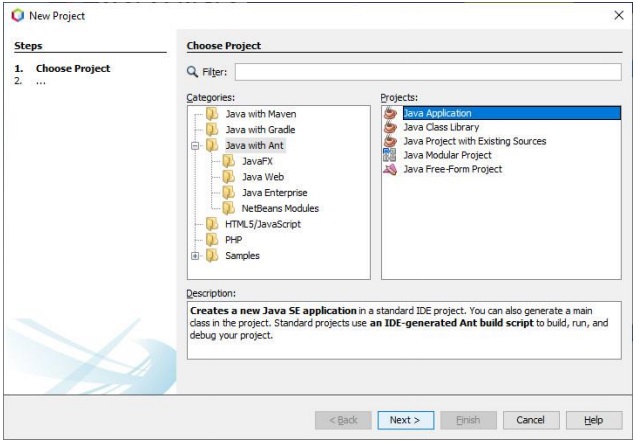
3. Запустити середовище NetBeans, створити java-проект

Рис. 1.4. Створення проекту в IDE NetBeans

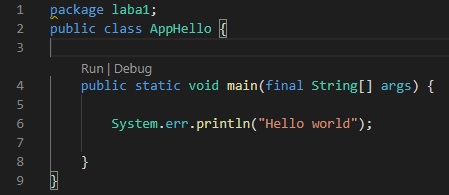
4. Написати код програми

Рис. 1.5. Написання коду програми

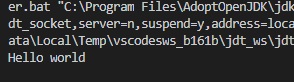
6. Відкомпілювати проект, виконати відлагодження

Рис. 1.6. Запуск програми в IDE середовищі

7. Ознайомитися з структурою проекту

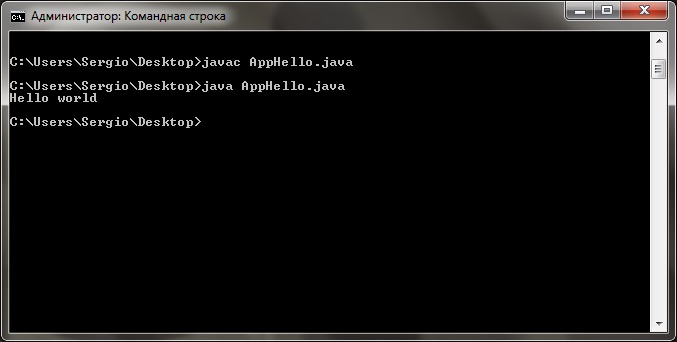
8. Відкомпілювати і запустити програму з командної строки

Рис. 1.7. Запуск програми з командної строки

**ЛАБОРАТОРНА РОБОТА №2**

**Типи даних, оператори, конструкції**

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

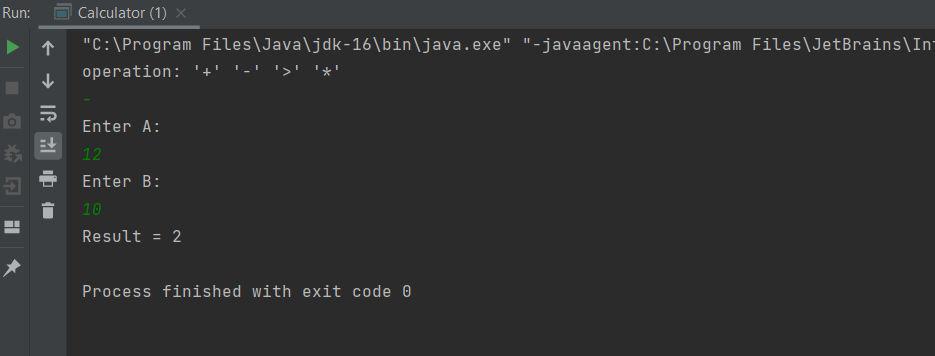
3. Задати константи для кожної з арифметичних операцій.

4. Викликати відповідну функцію, використовуючи конструкцію switch, яка

приймає на вхід змінну зі значенням арифметичної операції.

**Код**

package calculator;  
  
import java.util.Scanner ;  
  
public class Calculator {  
  
 final static int *NOT\_FOUND* = 0;  
 final static int *ADDITION* = 1;  
 final static int *SUBTRACTION* = 2;  
 final static int *MULTIPLICATION* = 3;  
 final static int *COMPARE* = 4;  
  
 int A;  
 int B;  
 int operation;  
 int Result;  
  
  
 public int sum (int a, int b) {  
 return a + b;  
 }  
  
 public int sub (int a, int b) {  
 return a - b;  
 }  
  
 public int multiply (int a, int b) {  
 return a \* b;  
 }  
  
 // public int compare (int a, int b) {  
// return (a > b) ? a : b;  
// }  
 public int compare (int a, int b) {  
 if(a>b){  
 return a;  
 }else {  
 return b;  
 }  
  
 }  
  
  
 public void getOration ( String operationName) {  
 if (operationName.equalsIgnoreCase("+")) {  
 operation = *ADDITION*;  
 }else if (operationName.equalsIgnoreCase("-")){  
 operation = *SUBTRACTION*;  
 }else if (operationName.equalsIgnoreCase(">")){  
 operation = *COMPARE*;  
 }else if (operationName.equalsIgnoreCase("\*")){  
 operation = *MULTIPLICATION*;  
 } else {  
 operation = *NOT\_FOUND*;  
 }  
 }  
  
 public void doOperation () {  
 switch (operation) {  
 case *ADDITION*:  
 Result = sum(A, B);  
 break;  
  
 case *SUBTRACTION*:  
 Result = sub(A, B);  
 break;  
  
 case *MULTIPLICATION*:  
 Result = multiply(A, B);  
 break;  
  
 case *COMPARE*:  
 Result = compare(A, B);  
 break;  
 }  
 }  
  
 public void inputArguments () {  
 Scanner in = new Scanner(System.*in*);  
  
 System.*out*.println("Enter A: ");  
 A = Integer.*valueOf*(in.nextLine());  
  
 System.*out*.println("Enter B: ");  
 B = Integer.*valueOf*(in.nextLine());  
 }  
  
 public void inputOperation () {  
 System.*out*.println("operation: '+' '-' '>' '\*' ");  
 Scanner in = new Scanner(System.*in*);  
 String userOperation = in.nextLine();  
 getOration(userOperation);//присвоюю значення новій операції  
 }  
  
  
  
 public void printResult(){  
 System.*out*.println("Result = " + Result);  
 }  
  
 public static void main(String[] args) {  
 Calculator calc = new Calculator();  
 calc.inputOperation();  
  
 if(calc.operation == *NOT\_FOUND*) {  
 System.*out*.println("Error operation, sorry.");  
 } else {  
 calc.inputArguments();  
 calc.doOperation();  
 calc.printResult();  
 }  
 }  
}



**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. Створити третій об’єкт даного класу.

5. Виконати присвоєння об’єктів (першій країні присвоїти третю).

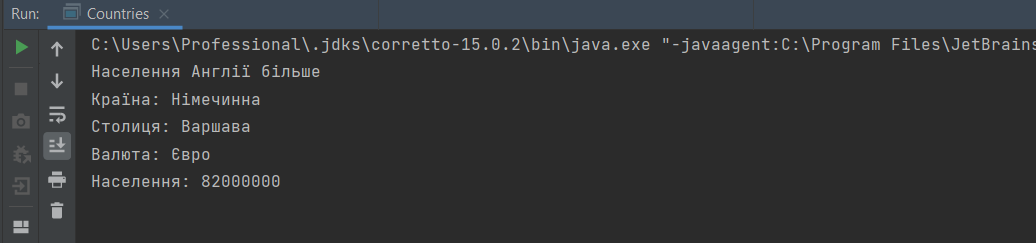
6. Змінити значення довільного поля в третьому об’єкті.

7. Вивести значення полів першого об’єкту.

**Код**

public class Country {  
 String title;  
 String capital;  
 String currency;  
 int population;  
}

public class Countries {  
  
 public static void main(String[] args) {  
  
 Country Ukraine = new Country();  
 Ukraine.title = "Україна";  
 Ukraine.capital = "Київ";  
 Ukraine.currency = "UAH";  
 Ukraine.population = 37000000;  
   
 Country England = new Country();  
 England.title = "Англія";  
 England.capital = "Лондон";  
 England.currency = "Pound sterling";  
 England.population = 56000000;  
  
  
   
 if(England.population > Ukraine.population){  
 System.*out*.println("Населення Англії більше");  
 } else {  
 System.*out*.println("Населення України більше");  
 }  
  
 Country Germany = new Country();  
 Germany.title = "Німечинна";  
 Germany.capital = "Берлін";  
 Germany.currency = "Євро";  
 Germany.population = 82000000;  
  
 Ukraine = Germany;  
   
 Ukraine.capital = "Варшава";  
  
 System.*out*.println(  
 "Країна: " + Ukraine.title + '\n' +   
 "Столиця: " + Ukraine.capital + '\n' +   
 "Валюта: " + Ukraine.currency + '\n' +   
 "Населення: " + Ukraine.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.

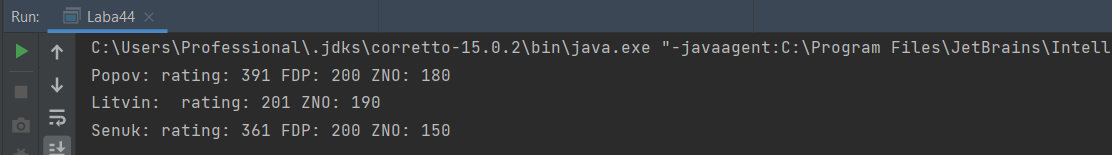
5. Розрахувати і вивести рейтинг кожного студента.

**Код**

package com.mycompany.laba4;  
public class Student extends People{  
 private int rating;  
 private int ZNO;  
 private int PDF;  
 private int atestat;  
 private static int *numberOfBoxes* = 0;  
  
  
  
 public Student(int ZNO, int PDF, int atestat, String name, String fio) {  
 super(name, fio);  
 this.ZNO = ZNO;  
 this.PDF = PDF;  
 this.atestat = atestat;  
 rating = ZNO + PDF + atestat;  
 }  
  
  
 public Student(int ZNO, int atestat, String name, String fio) {  
 super(name, fio);  
 this.ZNO = ZNO;  
 this.atestat = atestat;  
 rating = ZNO + atestat;   
 }  
  
  
 public boolean checkRating(int bal) {  
 return (rating >= bal);  
 }  
 public void printInfo(){  
 if (PDF > 0 ) {  
 System.*out*.println( getFio() + ": " + "rating: " + rating + " FDP: " +  
 PDF + " ZNO: " + ZNO );  
 } else {  
 System.*out*.println( getFio() + ": " + " rating: " + rating +  
 " ZNO: " + ZNO );  
 }  
 }  
 public int getRating() {  
 return rating;  
 }  
 public int getZNO() {  
 return ZNO;  
 }  
 public int getPDF() {  
 return PDF;  
 }  
 public int getAtestat() {  
 return atestat;  
 }  
 public void setRating(int rating) {  
 this.rating = rating;  
 }  
 public void setZNO(int ZNO) {  
 this.ZNO = ZNO;  
 }  
 public void setPDF(int PDF) {  
 this.PDF = PDF;  
 }  
 public void setAtestat(int atestat) {  
 this.atestat = atestat;  
 }   
}  
package com.mycompany.laba4;  
  
  
public class Boxes {  
 private Student[] students;  
 public void generateStudents(){  
 Student Nikita = new Student(180, 200, 11, "Nikita", "Didenko");  
 Student Robert = new Student(150, 200, 11, "Nastia", "Platina");  
 Student Oleg = new Student(190, 11, "Oleg", "Kizaru");  
 students = new Student[3];  
 students[0] = Nikita;  
 students[1] = Oleg;  
 students[2] = Robert;  
 }  
 public Student getSmallestBox(){  
 Student min = students[0];  
   
 for (Student item : students){  
 if(item.getRating()< min.getRating()){  
 min = item;   
 }  
 }  
 return min;  
 }  
 public Student getBiggestBox(){  
 Student max = students[0];  
 for (Student item : students){  
 if(item.getRating()> max.getRating()){  
 max = item;   
 }  
 }  
 return max;  
 }  
 public Student getSerBox(){  
 Student min = getSmallestBox();  
 Student max = getBiggestBox();  
 Student ser = students[0];  
 for (Student item : students){  
 if(item.getRating() != max.getRating()){  
 if(item.getRating() != min.getRating()){  
 ser = item;  
 }   
 }  
 }  
 return ser;  
 }  
}

package com.mycompany.laba4;  
  
  
public class People {  
 private String name;  
 private String fio;  
  
 public People(String name, String fio) {  
 this.name = name;  
 this.fio = fio;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public String getFio() {  
 return fio;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public void setFio(String fio) {  
 this.fio = fio;  
 }   
}

package com.mycompany.laba4;  
  
  
public class Laba44 {  
 public static void main(String[] args) {  
 Student Nikita = new Student(180, 200, 11, "Nikita", "Popov");  
 Student Maks = new Student(150, 200, 11, "Maks", "Senuk");  
 Student Oleg = new Student(190, 11, "Oleg", "Litvin");  
 Nikita.printInfo();  
 Oleg.printInfo();  
 Maks.printInfo();  
  
 }  
}



**ЛАБОРАТОРНА РОБОТА №5**

**Робота з масивами об’єктів**

Мета роботи – навчитися працювати з масивом об’єктів та з статичними

членами класів.

**Завдання**

1. Створити клас «Квиток», який містить наступні атрибути: прізвище, тип атракціону (фіксований набір значень (тир, карусель, оглядове колесо)), вартість квитка, ідентифікатор. Клас також містить статичне поле, для підрахунку кількості посилок.

2. Створити масив об’єктів типу «Квиток». Можна даний масив оформити

у вигляді окремого класу.

3. Створити декілька об’єктів класу «Квиток», і записати їх в масив.

4. Знайти найдешевший та найдорожчий квиток.

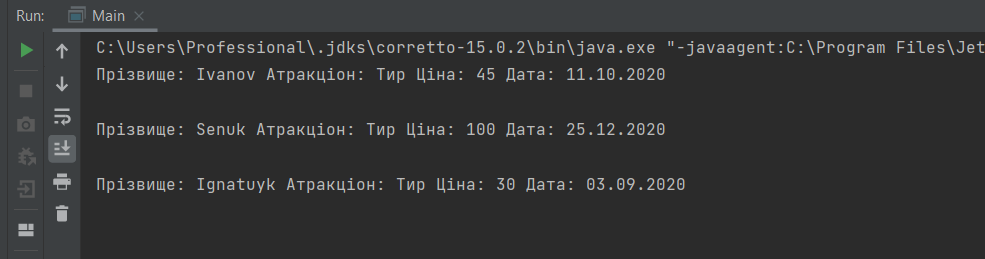
5. Перевірити кількість вільних квитків. Якщо це значення перевищує(або дорівнює) якесь граничне значення – вивести повідомлення.

**Код**

public class Ticket {  
  
 private String lastName;  
 private String attractionType;  
 private int price;  
 private String date;  
  
 public Ticket(String lastName, String attractionType, int price, String date) {  
 this.lastName = lastName;  
 this.attractionType = attractionType;  
 this.price = price;  
 this.date = date;  
 }  
  
 public String getAttractionType() {  
 return attractionType;  
 }  
  
  
 public void printTicket(){  
 System.*out*.println("Прізвище: " + lastName + " " +  
 "Атракціон: " + attractionType + " " +  
 "Ціна: " + price + " " +  
 "Дата: " + date + "\n"  
 );  
 }  
  
  
}

public class Tickets {  
  
 private Ticket[] tickets;  
  
 public void generateTickets() {  
 Ticket ticket1 = new Ticket("Ivanov", "Тир", 45, "11.10.2020");  
 Ticket ticket2 = new Ticket("Senuk", "Оглядове колесо", 30, "04.09.2020");  
 Ticket ticket3 = new Ticket("Senuk", "Тир", 100, "25.12.2020");  
 Ticket ticket4 = new Ticket("Ignatuyk", "Карусель", 30, "05.09.2020");  
 Ticket ticket5 = new Ticket("Ignatuyk", "Тир", 30, "03.09.2020");  
 Ticket ticket6 = new Ticket("Petrenko", "Оглядове колесо", 30, "09.09.2020");  
 Ticket ticket7 = new Ticket("Fesenko", "Карусель",30, "30.02.2021");  
  
  
 tickets = new Ticket[7];  
 tickets[0] = ticket1;  
 tickets[1] = ticket2;  
 tickets[2] = ticket3;  
 tickets[3] = ticket4;  
 tickets[4] = ticket5;  
 tickets[5] = ticket6;  
 tickets[6] = ticket7;  
  
 }  
  
 public void getAllTicketsOfType(String typeTicket){  
 if (typeTicket == "Карусель" || typeTicket == "Тир" || typeTicket == "Оглядове колесо" ){  
 for (Ticket currentTicket : tickets){  
 if (currentTicket.getAttractionType() == typeTicket){  
 currentTicket.printTicket();  
 }  
 }  
 } else {  
 System.*out*.println("Такого атракціону не існує");  
 }  
 }  
}

public class Main {  
 public static void main(String[] args) {  
  
 Tickets tickets = new Tickets();  
  
 tickets.generateTickets();  
 tickets.getAllTicketsOfType("Тир");  
 }  
}



**ЛАБОРАТОРНА РОБОТА №6**

**Вкладені класи**

Мета роботи – навчитися оголошувати і використовувати вкладені 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 клас для обчислення стипендії.

Стипендія дається студентам в яких середній бал більше 4,5 та кількість

пропусків менша 10.

3. В класі «Студент» написати локальний клас для визначення

військовозобов’язаних студентів. Використати його в конструкторі.

Військовозобов’язаними вважати чоловіків старших 18 років.

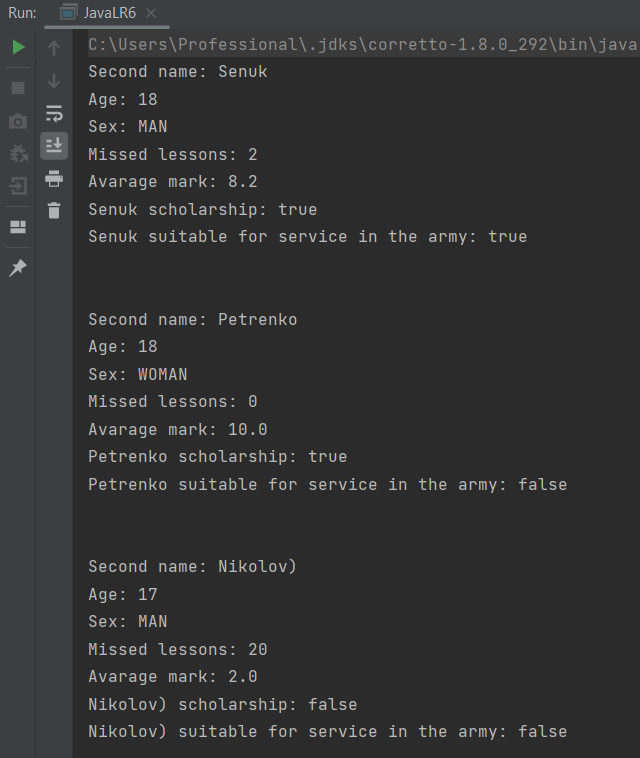
4. Створити декілька об’єктів класу студент.

5. Розрахувати і вивести характеристики кожного студента.

**Код**

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 = 10;  
 private final double MIN\_AVARAGE\_MARK = 4.5;  
  
 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);  
 }  
}package javalr6;  
  
public enum Sex {  
 *MAN*, *WOMAN*}

package javalr6;  
  
public class JavaLR6 {  
 public static void main(String[] args) {   
 Student Ftemov = new Student("Senuk", 18, Sex.MAN, 2, 8.2);  
 Ftemov.Data();  
  
 System.*out*.println("\n");  
  
 Student Sokhatska = new Student("Petrenko", 18, Sex.WOMAN, 0, 10);  
 Sokhatska.Data();  
  
 System.*out*.println("\n");  
  
 Student Kryvenko = new Student("Nikolov)", 17, Sex.MAN, 20, 2);  
 Kryvenko.Data();  
 }  
}



**ЛАБОРАТОРНА РОБОТА №7**

**Поліморфізм**

Мета роботи – навчитися використовувати 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.

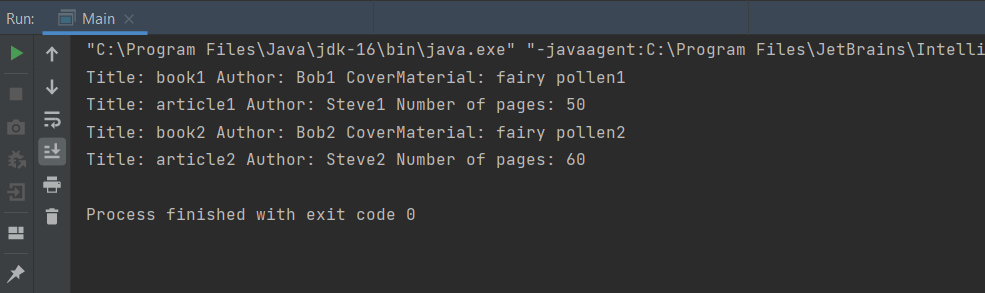
5. Створити ієрархію класів робітників компанії з погодинною оплатою

праці та фіксованою ставкою з премією. В кожному з класів реалізувати метод

нарахування зарплати. Створити декілька об’єктів цих класів, в циклі

розрахувати заробітну плату для всіх працівників компанії.

**Код**

package com.company;  
  
public class Main {  
  
 static abstract class Document {  
 String title;  
 String author;  
  
 public Document(String title\_, String author\_){  
 title = title\_;  
 author = author\_;  
 }  
  
 public abstract void print();  
 }  
  
 static class Book extends Document {  
 String coverMaterial;  
 public Book(String title\_, String author\_, String coverMaterial\_){  
 super(title\_, author\_);  
 coverMaterial = coverMaterial\_;  
 }  
  
 public void print(){  
 System.*out*.println("Title: "+ title+" Author: " + author+ " CoverMaterial: "+ coverMaterial);  
 }  
  
 }  
  
 static class Article extends Document {  
 int pagesNum;  
 public Article(String title\_, String author\_, int pagesNum\_){  
 super(title\_, author\_);  
 pagesNum = pagesNum\_;  
 }  
  
 public void print(){  
 System.*out*.println("Title: "+ title+" Author: " + author+ " Number of pages: "+ pagesNum);  
 }  
 }  
  
  
 public static void main(String[] args) {  
 Document[] doc = new Document[] {  
 new Book("book1", "Bob1", "fairy pollen1"),  
 new Article("article1", "Steve1", 50),  
 new Book("book2", "Bob2", "fairy pollen2"),  
 new Article("article2", "Steve2", 60)  
 };  
  
 for(int i = 0; i < 4; i++){  
 doc[i].print();  
 }  
 }  
}

**ЛАБОРАТОРНА РОБОТА №8**

**Exceptions**

Мета роботи – навчитися оброблювати і генерувати виняткові ситуації.

**Завдання**

Написати клас з методами, які генерують наступні виняткові ситуації:

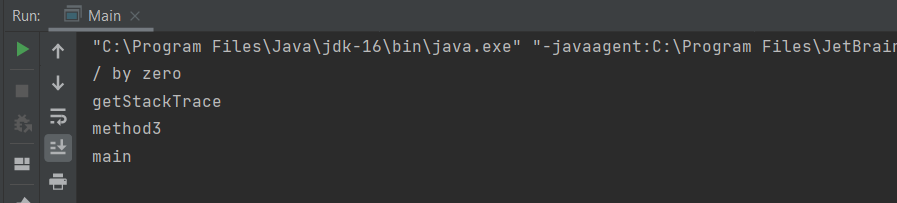
* ділення на нуль (без обробки);
* ділення на нуль (з блоком try ... catch);
* з декількома блоками catch;
* з виведенням опису помилки та стеку викликів;
* методи, які повертають помилку throwable;
* в методі, описаному як throwable згенерувати помилку і обробити її в головній програмі.

Написати користувацький клас виключення, згенерувати і перехопити

помилку даного класу.

**Код**

package com.company;  
  
public class Main {  
  
 static class Math{  
 int a;  
 int b;  
  
 public Math(int a\_, int b\_){  
 a = a\_;  
 b = b\_;  
 }  
 //1  
 public void divOnZero(){  
 a = a / 0;  
 }  
  
 //2,3,4  
 public void divOnZeroTry(int exp){  
 int c;  
 try {  
 if(exp == 1)  
 c = a / 0;  
 else  
 Integer.*parseInt*("aaa");  
 }  
 catch (ArithmeticException e){  
 System.*out*.println(e.getMessage());  
 *method3*();  
 }  
 catch (NumberFormatException e){  
 System.*out*.println(e.getMessage());  
 *method3*();  
 }  
 }  
  
 //5  
 public Throwable throwableRet(){  
 try {  
 int n = 2 / 0;  
 }  
 catch (Throwable e){  
 System.*out*.println(e.getMessage());  
 return e;  
 }  
 return new Throwable();  
 }  
  
 //6  
 public void throwable(){  
 int n = 2 / 0;  
 }  
  
 //3  
 public static void method3()  
 {  
 StackTraceElement[] stackTraceElements = Thread.*currentThread*().getStackTrace();  
 for (StackTraceElement element : stackTraceElements)  
 {  
 System.*out*.println(element.getMethodName());  
 }  
 }  
  
 }  
  
 public static void main(String[] args) {  
 Math a = new Math(2, 2);  
  
 //1  
 //a.divOnZero();  
  
 //2,3,4  
 //a.divOnZeroTry(2);  
  
 //6  
 try {  
 a.throwable();  
 }  
 catch (Throwable e){  
 System.*out*.println(e.getMessage());  
 a.*method3*();  
 }  
 }  
}



**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

2. Написати 2 класи, які реалізують цей інтерфейс: Samsung, RF

3. Написати клас ARM (автоматизоване робоче місце касира), для продажу

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

4. Промоделювати роботу каси, по черзі підключити два касових апарати,

продати 2-3 товари, вивести суму за день.

**Код**

public class Supermarket {  
  
 public static void main(String[] args) {  
 int samsungSum = 0;  
 int rfSum = 0;  
 int totalMoney = 0;  
 CashierARM kassa = new CashierARM();  
  
  
 Ecr aparatEcr = new Samsung(1, "Ivanova");  
 kassa.connectErc(aparatEcr);  
  
  
 kassa.newBuyer();  
  
  
 kassa.addGoodsToBascket(1, 2);  
 kassa.addGoodsToBascket(2,1);  
 kassa.addGoodsToBascket(3, 8);  
  
 int userCash = 1000;  
 kassa.printReceipt(userCash);  
  
 kassa.newBuyer();  
  
  
 kassa.addGoodsToBascket(1, 5);  
 kassa.addGoodsToBascket(3,2);  
 kassa.addGoodsToBascket(2, 1);  
  
  
 userCash = 1500;  
  
 kassa.printReceipt(userCash);  
  
 samsungSum += kassa.doReport();  
  
 aparatEcr = new RF(2, "Golovatyuk");  
 kassa.connectErc(aparatEcr);  
  
  
 kassa.newBuyer();  
  
 kassa.addGoodsToBascket(2, 4);  
 kassa.addGoodsToBascket(1, 3);  
 kassa.addGoodsToBascket(2, 8);  
  
 userCash = 500;  
  
 kassa.printReceipt(userCash);  
  
 rfSum += kassa.doReport();  
  
 totalMoney = samsungSum + rfSum;  
  
  
 System.*out*.println("Сума зароблена з Samsung: " + samsungSum);  
 System.*out*.println("Сума зароблена з RF: " + rfSum);  
 System.*out*.println("\n-----------------------------------------------------");  
 System.*out*.printf("%-20s %s \n%s %d\n","","[Підсумок]", "Сума зароблена за сьогодні (з двох апаратів): ", totalMoney);  
 System.*out*.println("-----------------------------------------------------");  
  
 }  
}

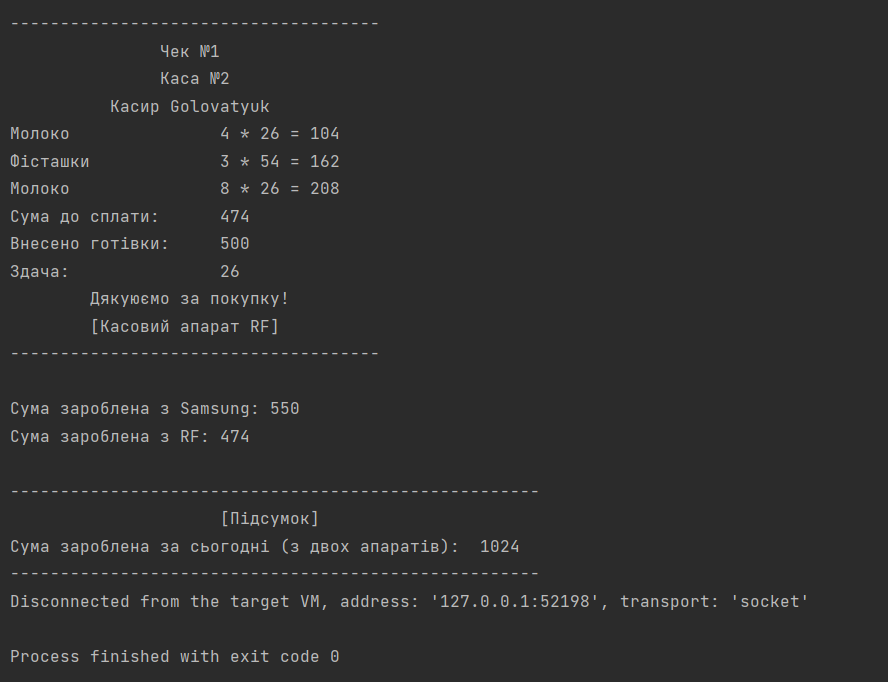
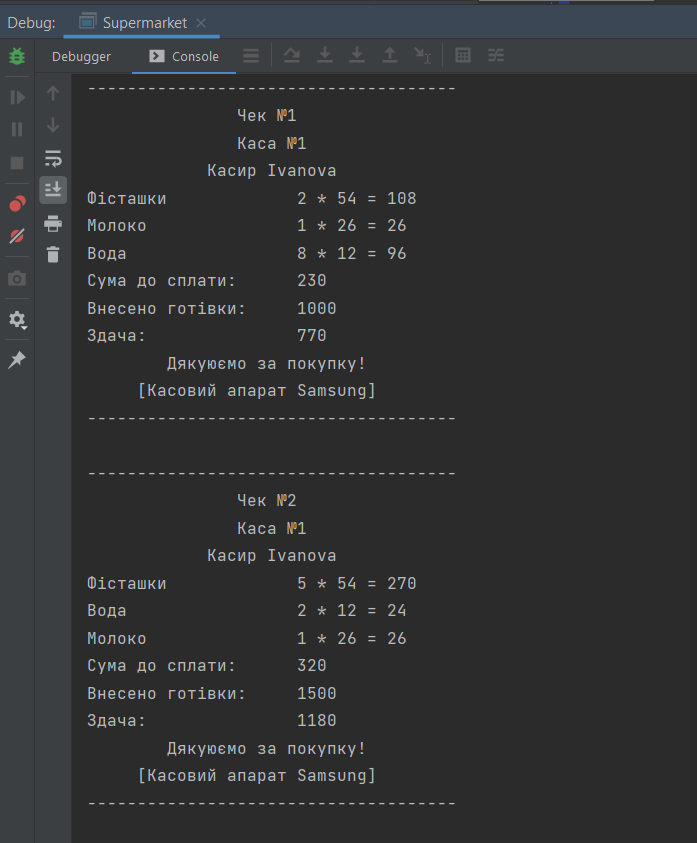
public class Samsung implements Ecr {  
 // Задаємо поля cashID і operatorName  
 private int cashID;  
 private String operatorName;  
  
 // Задаємо поля totalEcrCash і totalReceiptCash  
 private int totalEcrCash = 0;  
 private int totalReceiptCash = 0;  
  
 // Конструктор класу Samsung  
 public Samsung(int cashID, String operatorName) {  
 this.cashID = cashID;  
 this.operatorName = operatorName;  
 }  
  
 // Відкриваємо чек  
 @Override  
 public void openReceipt(int reseiptID) {  
 totalReceiptCash = 0;  
 System.*out*.println("-------------------------------------");  
 System.*out*.printf("%-14s %s%d \n","","Чек №", reseiptID);  
 System.*out*.printf("%-14s %s%d \n","","Каса №", cashID);  
 System.*out*.printf("%-11s %s%s \n","","Касир ", operatorName);  
 }  
  
 // Виводимо усі продукти з корзини  
 @Override  
 public void printGood(String name, int quantity, int price) {  
 int totalSum = quantity \* price;  
 totalReceiptCash += totalSum;  
 System.*out*.printf("%-20s %d \* %d = %d \n", name, quantity, price, totalSum);  
 }  
  
 // Закриваємо чек  
 @Override  
 public void closeReceipt(int cash) {  
 totalEcrCash += totalReceiptCash;  
 System.*out*.printf("%-20s %d \n", "Сума до сплати: ", totalReceiptCash);  
 System.*out*.printf("%-20s %d \n", "Внесено готівки: ", cash);  
 int change = cash - totalReceiptCash;  
 if (change != 0) {  
 System.*out*.printf("%-20s %d \n", "Здача: ", cash - totalReceiptCash);  
 }  
 System.*out*.printf("%-7s %s \n", "", "Дякуюємо за покупку!");  
 printDelimiter();  
 }  
  
 // Денний звіт з поточного касового апарату  
 @Override  
 public int dailyReport() {  
 return totalEcrCash;  
 }  
  
 // Друк друк марки апарату і роздільної лінії  
 public void printDelimiter() {  
 System.*out*.printf("%-4s %s \n","","[Касовий апарат Samsung]");  
 System.*out*.println("-------------------------------------\n");  
 }  
}

public class RF implements Ecr {  
 // Задаємо поля cashID і operatorName  
 private int cashID;  
 private String operatorName;  
  
 // Задаємо поля totalEcrCash і totalReceiptCash  
 private int totalEcrCash = 0;  
 private int totalReceiptCash = 0;  
  
 // Конструктор класу RF  
 public RF(int cashID, String operatorName) {  
 this.cashID = cashID;  
 this.operatorName = operatorName;  
 }  
  
 // Відкриваємо чек  
 @Override  
 public void openReceipt(int reseiptID) {  
 totalReceiptCash = 0;  
 System.*out*.println("-------------------------------------");  
 System.*out*.printf("%-14s %s%d \n","","Чек №", reseiptID);  
 System.*out*.printf("%-14s %s%d \n","","Каса №", cashID);  
 System.*out*.printf("%-9s %s%s \n","","Касир ", operatorName);  
 }  
  
 // Виводимо усі продукти з корзини  
 @Override  
 public void printGood(String name, int quantity, int price) {  
 int totalSum = quantity \* price;  
 totalReceiptCash += totalSum;  
 System.*out*.printf("%-20s %d \* %d = %d \n", name, quantity, price, totalSum);  
 }  
  
 // Закриваємо чек  
 @Override  
 public void closeReceipt(int cash) {  
 totalEcrCash += totalReceiptCash;  
 System.*out*.printf("%-20s %d \n", "Сума до сплати: ", totalReceiptCash);  
 System.*out*.printf("%-20s %d \n", "Внесено готівки: ", cash);  
 int change = cash - totalReceiptCash;  
 if (change != 0) {  
 System.*out*.printf("%-20s %d \n", "Здача: ", cash - totalReceiptCash);  
 }  
 System.*out*.printf("%-7s %s \n", "", "Дякуюємо за покупку!");  
 printDelimiter();  
 }  
  
 // Денний звіт з поточного касового апарату  
 @Override  
 public int dailyReport() {  
 return totalEcrCash;  
 }  
  
 // Друк друк марки апарату і роздільної лінії  
 public void printDelimiter() {  
 System.*out*.printf("%-7s %s \n","","[Касовий апарат RF]");  
 System.*out*.println("------------------------------------- \n");  
 }  
}

public class Good {  
 // Поля ціни та касира  
 private int price;  
 private String name;  
  
 // Конструктор класу Good  
 public Good(int price, String name) {  
 this.price = price;  
 this.name = name;  
 }  
  
 // Геттери і Сеттери  
 public int getPrice() {  
 return price;  
 }  
  
 public String getName() {  
 return name;  
 }  
}

public interface Ecr {  
  
 void openReceipt(int reseiptID);  
 void printGood(String name, int quantity, int price);  
 void closeReceipt(int cash);  
 int dailyReport();  
}

public class CashierARM {  
 // Універсально підключаємо касові апарати  
 private Ecr ecr;  
  
 // Створюємо масив товарів  
 private Good[] goodsForSale;  
  
 // Створюємо двовимірний масив для товарів вибраних покупцем  
 private int[][] userGoods;  
 private int userGoodsCount;  
  
 // Айдішнік чеку  
 private int receiptID;  
  
 // Реалізуємо масив товарів  
 public CashierARM() {  
 receiptID = 0;  
 goodsForSale = new Good[4];  
  
 // Заповнюємо товарами  
 goodsForSale[0] = new Good(17, "Апельсини");  
 goodsForSale[1] = new Good(54, "Фісташки");  
 goodsForSale[2] = new Good(26, "Молоко");  
 goodsForSale[3] = new Good(12, "Вода");  
 }  
  
 // Підключення апарату  
 public void connectErc(Ecr newEcr) {  
 receiptID = 0;  
 ecr = newEcr;  
 }  
  
 // Фінальний друк чеку  
 public void printReceipt(int cash) {  
 ecr.openReceipt(receiptID);  
 for (int[] userGoods : userGoods) {  
 Good good = goodsForSale[userGoods[0]];  
 int quantity = userGoods[1];  
 if (quantity != 0) {  
 ecr.printGood(good.getName(), quantity, good.getPrice());  
 }  
 }  
 ecr.closeReceipt(cash);  
 }  
  
 // Звіт  
 public int doReport() {  
 return ecr.dailyReport();  
 }  
  
 // Метод додавання товару  
 public void addGoodsToBascket(int id, int quantity) {  
 userGoods[userGoodsCount] [0] = id;  
 userGoods[userGoodsCount] [1] = quantity;  
 userGoodsCount++;  
 }  
  
 // Новий покупець  
 public void newBuyer() {  
 receiptID++;  
 userGoodsCount = 0;  
 userGoods = new int[4][2];  
 }  
}



**ЛАБОРАТОРНА РОБОТА №10**

**Узагальнені типи та колекції**

Мета роботи – навчитися працювати з групами об’єктів, використовуючи

колекції.

**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

Клас «квиток» має містити наступні поля:

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

**Код**

public class Ticket {  
 private int id;  
 private String lastName;  
 private typeOfAttraction typeOfAttraction;  
 private int price;  
 private String date;  
 enum typeOfAttraction {  
 *SHOOTING\_GALLERY*("Тир"),  
 *WHEEL\_REVIEWS*("Оглядове колесо"),  
 *ROLLER\_COASTER*("Амераеканські гірки"),  
 *BOATS*("Лодочки"),  
 *CARS*("Тачки");  
 private String typeOfAttraction;  
 typeOfAttraction(String typeOfAttraction) {  
 this.typeOfAttraction = typeOfAttraction;  
 }  
 public String getTypeOfAttraction() {  
 return typeOfAttraction;  
 }  
 @Override  
 public String toString() {  
 return typeOfAttraction;  
 }  
 }  
 public Ticket(int id, String lastName, Ticket.typeOfAttraction  
 typeOfAttraction, int price, String date) {  
 this.id = id;  
 this.lastName = lastName;  
 this.typeOfAttraction = typeOfAttraction;  
 this.price = price;  
 this.date = date;  
 }  
 public int getId() {  
 return id;  
 }  
 public String getLastName() {  
 return lastName;  
 }  
 public Ticket.typeOfAttraction getTypeOfAttraction() {  
 return typeOfAttraction;  
 }  
 public int getPrice() {  
 return price;  
 }  
 public String getDate() {  
 return date;  
 }  
 public void setLastName(String lastName) {  
 this.lastName = lastName;  
 }  
 public void setTypeOfAttraction(Ticket.typeOfAttraction  
 typeOfAttraction) {  
 this.typeOfAttraction = typeOfAttraction;  
 }  
 public void setPrice(int price) {  
 this.price = price;  
 }  
 public void setDate(String date) {  
 this.date = date;  
 }  
 @Override  
 public String toString() {  
 String ticket = String.*format*(" | ID квитка = %-5d || Прізвище = %-12s || Тип атракціону = %-17s || Ціна = %-5d || Дата придбання =%s |",  
 id,lastName, typeOfAttraction,price,date);  
 return ticket;  
 }  
}

import org.w3c.dom.ls.LSOutput;  
import java.util.\*;  
public class Tickets {  
 List<Ticket> ticketsList = new ArrayList<Ticket>();  
 public void generationTickets() {  
 ticketsList.add(new Ticket(1, "Senuk",  
 Ticket.typeOfAttraction.*BOATS*, 30, "27.04.2021"));  
 ticketsList.add(new Ticket(2, "Kotelevich",  
 Ticket.typeOfAttraction.*CARS*, 45, "30.04.2021"));  
 ticketsList.add(new Ticket(3, "Perenko",  
 Ticket.typeOfAttraction.*SHOOTING\_GALLERY*, 50, "01.05.2021"));  
 ticketsList.add(new Ticket(4, "Larin",  
 Ticket.typeOfAttraction.*WHEEL\_REVIEWS*, 35, "01.05.2021"));  
 ticketsList.add(new Ticket(5, "Moyseinko",  
 Ticket.typeOfAttraction.*WHEEL\_REVIEWS*, 35, "01.05.2021"));  
 System.*out*.println("\n------------------Інформаційна система квитків:-------------------------- \n");  
 devision();  
 printAllTickets();  
 }  
 public void printAllTickets() {  
 for (Ticket currentTicket : ticketsList) {  
 System.*out*.println(currentTicket);  
 }  
 }  
 public void sortTicketsByLastName() {  
 Collections.*sort*(ticketsList, new sortByLastName());  
 System.*out*.println("\n----------------------Відбулося сортування за прізвищем:---------------------- \n");  
 devision();  
 printAllTickets();  
 }  
 public void sortTicketsByType() {  
 Collections.*sort*(ticketsList, new sortByTypeOfAttraction());  
 System.*out*.println("\n--------------------------------Відбулося сортування за типом: ----------------\n");  
 devision();  
 printAllTickets();  
 }  
 public void addNewTicket(String lastName, Ticket.typeOfAttraction  
 typeOfAttraction, int price, String date) {  
 int id = ticketsList.size() + 1;  
 ticketsList.add(new Ticket(id, lastName, typeOfAttraction,  
 price, date));  
 }  
 public void editTickets(int id, String lastName,  
 Ticket.typeOfAttraction typeOfAttraction, int price, String date) {  
 for (Ticket currentTicket : ticketsList) {  
 if (id == currentTicket.getId()) {  
 System.*out*.println("\n----------Редагування квитка:------------\n" +  
 currentTicket);  
 devision();  
 currentTicket.setLastName(lastName);  
 currentTicket.setTypeOfAttraction(typeOfAttraction);  
 currentTicket.setPrice(price);  
 currentTicket.setDate(date);  
 System.*out*.println("\n-------------Квиток було змінено на: ---------------\n" +  
 currentTicket);  
 }  
 }  
 }  
 public void deleteTickets(int id) {  
 System.*out*.println("\n-------------------Видаляємо квиток: ------------------\n");  
 for (Iterator<Ticket> it = ticketsList.iterator();  
 it.hasNext(); ) {  
 Ticket currentTicket = it.next();  
 if (id == currentTicket.getId()) {  
 it.remove();  
 break;  
 }  
 }  
 }  
 public void devision() {  
 System.*out*.println(" ==================================================================================================================================================");  
 }  
 public void printTicketAttraction(Ticket.typeOfAttraction  
 typeOfAttraction){  
 System.*out*.println("\n---------------------Виведення квитків одного типу: ---------------------------\n");  
 devision();  
 for (Ticket currentTicket:ticketsList) {  
  
 if(currentTicket.getTypeOfAttraction().equals(typeOfAttraction)){  
 System.*out*.println(currentTicket);  
 }  
 }  
 }  
}

import java.util.Comparator;  
public class sortByLastName implements Comparator<Ticket>{  
 @Override  
 public int compare(Ticket ticket1, Ticket ticket2) {  
 return ticket1.getLastName().compareTo(ticket2.getLastName());  
 }  
}

import java.util.Comparator;  
public class sortByTypeOfAttraction implements Comparator<Ticket>{  
 @Override  
 public int compare(Ticket ticket1, Ticket ticket2) {  
 return  
 ticket1.getTypeOfAttraction().compareTo(ticket2.getTypeOfAttraction())  
 ;  
 }  
}

public class Main {  
 public static void main(String[] args) {  
 Tickets kassa = new Tickets();  
 kassa.generationTickets();  
 kassa.devision();  
 kassa.addNewTicket("Didenko", Ticket.typeOfAttraction.*BOATS*,  
 32, "15.05.2021");  
  
 kassa.addNewTicket("Panku", Ticket.typeOfAttraction.*BOATS*, 32,  
 "15.05.2021");  
 kassa.sortTicketsByLastName();  
 kassa.devision();  
 kassa.editTickets(4, "Gnatenko", Ticket.typeOfAttraction.*CARS*,  
 45, "24.04.2021");  
 kassa.devision();  
 kassa.sortTicketsByType();  
 kassa.devision();  
 kassa.deleteTickets(4);  
 kassa.devision();  
 kassa.printAllTickets();  
 kassa.devision();  
 kassa.printTicketAttraction(Ticket.typeOfAttraction.*BOATS*);  
 kassa.devision();  
  
 }  
}