**// LIST**

List<Integer> list = new ArrayList<>(); // INTERFACE AS TYPE

//cand atribuim unei interfete efective o clasa

list.add(4);

list.add(3);

list.add(2);

list.add(1);

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

System.out.println(list.get(i));

System.out.println();

list.remove(2);

for( Integer i:list)

System.out.println(i);

System.out.println();

list.set(1, 9);

for( Iterator<Integer> it = list.iterator(); it.hasNext();) {

System.out.println(it.next());

}

**//SET**

Set<Car> set = new TreeSet<Car>();

set.add(c2);

c2.setCapacity(3000);

set.add(c);

set.add(car);

for(Car x : set)

{

System.out.println(x);

}

**//MAP**

// cautarea rapida se face dupa cheie

Map<Car, String> map = new HashMap<Car, String>();

map.put(c2, "Ionel Ionescu");

map.put(c, "Ioana Euuu");

Car c3 = null;

c3=(Car)c.clone();

map.put(c3, "Gigel Georgescu");

for( Car x : map.keySet())

{

System.out.printf("%s :", x.toString());

System.out.println(map.get(x));

}

**// citire de la tastatura**

Scanner scanner = new Scanner(System.in);

String yourName = "";

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

yourName = scanner.nextLine();

int yourAge =0;

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

yourAge=scanner.nextInt();

System.out.println("Name = "+ yourName + " Age=" +yourAge);

scanner.close();

**//scriere in fisier txt**

Car car = new Car("Renault", 90, "blue", 1500);

try {

FileOutputStream fileOutputStream= new FileOutputStream("car.txt");

OutputStreamWriter streamWriter = new OutputStreamWriter(fileOutputStream);

BufferedWriter writer = new BufferedWriter(streamWriter);

writer.write(car.getName());

writer.write(System.lineSeparator()); *// folosim line separetor deoarece*

*//separatorul depinde de sistemul de operare/platforma pe care ruleaza program Integer speed = car.getSpeed();*

*//int-ul nu are ToString dar Integer ul da*

writer.write(speed.toString());

writer.write(System.lineSeparator());

*// writer.write va arunca o exceptie de tipul ioException*

*// si ilocuim exceptia din catch care era de tipul FileNotFindException cu IOException*

*// ca sa le printa pe toate*

writer.write(car.getColor());

writer.write(System.lineSeparator());

Integer cap = car.getCapacity();

writer.write(cap.toString());

writer.close();

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

**// citire din fisiere txt**

try {

FileInputStream fileInputStream = new FileInputStream("car.txt");

InputStreamReader streamReader = new InputStreamReader(fileInputStream);

BufferedReader reader= new BufferedReader(streamReader);

String name = reader.readLine();

int speed = 0;

speed = Integer.parseInt(reader.readLine());

String color = reader.readLine();

int capacity = Integer.parseInt(reader.readLine());

reader.close();

Car c2 = new Car( name, speed, color, capacity);

System.out.println(c2);

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

**// FISIER BINAR**

// folosind prima modalitate scriind informatie cu informatie, fiecare camp nonstatic in parte

**//scriere**

try {

FileOutputStream binaryOutputStream = new FileOutputStream("car.bin");

DataOutputStream dataOutputStream = new DataOutputStream(binaryOutputStream);

dataOutputStream.writeUTF(car.getName()); *// stie exact unde sa se opreasca cu scrierea citirea*

dataOutputStream.writeInt(car.getSpeed());

dataOutputStream.writeUTF(car.getColor());

dataOutputStream.writeInt(car.getCapacity());

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

**//citire**

try {

FileInputStream binaryInputStream = new FileInputStream("car.bin");

DataInputStream dataInputStream = new DataInputStream(binaryInputStream);

String name=dataInputStream.readUTF();

int speed = dataInputStream.readInt();

String color = dataInputStream.readUTF();

int capacity = dataInputStream.readInt();

Car c3= new Car( name, speed, color, capacity);

System.out.println(c3);

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

**//folosirea Serializarii a intregii clase**

car.serialize();

try {

Car c4 = Car.deserialize();

System.out.println(c4);

} catch (ClassNotFoundException | IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

**//serializam intreaga clasa**

public void serialize() {

FileOutputStream fileOutputStream;

try {

fileOutputStream = new FileOutputStream("object.bin");

ObjectOutputStream stream = new ObjectOutputStream(fileOutputStream);

stream.writeObject(this);

stream.close();

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

**//deserializam clasa**

public static Car deserialize() throws IOException, ClassNotFoundException {

FileInputStream fileInputStream = new FileInputStream("object.bin");

ObjectInputStream objectinputStream = new ObjectInputStream(fileInputStream);

Car c = (Car)objectinputStream.readObject();

objectinputStream.close();

return c;

}

**//conectare la baza de date**

Connection connection = null;

try {

Class.forName("org.sqlite.JDBC");

connection = DriverManager.getConnection("jdbc:sqlite:database.db");

connection.setAutoCommit(false);

createTable(connection); // functii din clasa

insertValues(connection);

selectData(connection);

} catch (ClassNotFoundException e) {

e.printStackTrace();

} catch (SQLException e) {

e.printStackTrace();

}

finally {

if(connection != null) {

try {

connection.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

}

}

//**creare tabela**

public static void createTable(Connection connection) {

String sqlDrop = "DROP TABLE IF EXISTS employees";

String sqlCreate = "CREATE TABLE employees(id INTEGER PRIMARY KEY, " +

"name TEXT, birthdate LONG, address TEXT, salary REAL)";

Statement statement;

try {

statement = connection.createStatement();

statement.executeUpdate(sqlDrop);

statement.executeUpdate(sqlCreate);

statement.close();

connection.commit();

} catch (SQLException e) {

e.printStackTrace();

}

}

//inserare de valori

public static void insertValues(Connection connection) {

String sqlInsert = "INSERT INTO employees VALUES(1, 'Ionel Popescu', 1589874134752, " +"'Stefan cel Mare nr 20', 2000)";

String sqlInsertWithParams = "INSERT INTO employees(name, birthdate, address, salary) " +"VALUES(?, ?, ?, ?)";

try {

Statement statement = connection.createStatement();

statement.executeUpdate(sqlInsert);

statement.close();

connection.commit();

PreparedStatement preparedStatement =

connection.prepareStatement(sqlInsertWithParams);

preparedStatement.setString(1, "Gigel Ionescu");

preparedStatement.setLong(2, Date.valueOf("1995-05-17").getTime());

preparedStatement.setString(3, "Mihai Bravu nr 15");

preparedStatement.setDouble(4, 4000);

preparedStatement.executeUpdate();

preparedStatement.close();

connection.commit();

} catch (SQLException e) {

e.printStackTrace();

}

}

**//citire din baza de date**

public static void selectData(Connection connection) {

String sqlSelect = "SELECT \* FROM employees";

try {

Statement statement = connection.createStatement();

ResultSet rs = statement.executeQuery(sqlSelect);

while(rs.next()) {

int id = rs.getInt("id");

System.out.println("id: " + id);

String name = rs.getString("name");

System.out.println("name: " + name);

long birthDate = rs.getLong("birthdate");

System.out.println("birthdate: " + new Date(birthDate));

String address = rs.getString("address");

System.out.println("address: " + address);

double salary = rs.getDouble("salary");

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

}

rs.close();

statement.close();

} catch (SQLException e) {

e.printStackTrace();

}

//scriere in csv

import java.io.File;

import java.io.FileWriter;

import java.io.IOException;

import java.util.ArrayList;

import java.util.Collection;

import java.util.Collections;

import java.util.List;

class Absolvent implements Comparable{

public int idElev;

public String nume;

public double notaMatematica;

public double notaRomana;

public double mediaAnilor;

public Absolvent(int idElev, String nume, double notaMatematica,

double notaRomana, double mediaAnilor)

{

this.idElev = idElev;

this.nume= nume;

this.notaMatematica = notaMatematica;

this.notaRomana = notaRomana;

this.mediaAnilor=mediaAnilor;

}

@Override

public String toString() {

StringBuilder builder = new StringBuilder();

builder.append("Absolvent [idElev=");

builder.append(idElev);

builder.append(", nume=");

builder.append(nume);

builder.append(", notaMatematica=");

builder.append(notaMatematica);

builder.append(", notaRomana=");

builder.append(notaRomana);

builder.append(", mediaAnilor=");

builder.append(mediaAnilor);

builder.append("]");

return builder.toString();

}

public double MediaGenerala(Absolvent a)

{

return a.mediaAnilor\*0.2+ a.notaMatematica\*0.4+a.notaRomana\*0.4;

}

@Override

public int compareTo(Object o) {

Absolvent altul = (Absolvent) o;

return Double.compare(MediaGenerala(this),MediaGenerala(altul));

}

}

public class TestMain {

static void SalvareAbsolventi( String cale, List<Absolvent> absolventi)

{

if(new File(cale).getParentFile()!= null)

{

//ne asiguram ca acesta exista

new File(cale).getParentFile().mkdirs();

}

try(var fisier = new FileWriter(cale))

{

for(var a: absolventi)

{

fisier.write(a.toString());

fisier.write("\n");

}

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

public static void main(String[] args) {

List<Absolvent> absolventi = new ArrayList<Absolvent>();

Absolvent a1=new Absolvent(100, "Ion", 10,10,10);

Absolvent a2=new Absolvent(101, "Ana", 9,9,9);

Absolvent a3=new Absolvent(102, "Mihai", 8,8,8);

Absolvent a4=new Absolvent(103, "Maria",7,7,7);

absolventi.add(a1);

absolventi.add(a2);

absolventi.add(a3);

absolventi.add(a4);

for(var a:absolventi)

{

System.out.println(a);

}

Collections.sort(absolventi);

List<Absolvent> sortata=new ArrayList<Absolvent>();

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

for(var a:absolventi)

{

sortata.add(a);

}

for(var a:absolventi)

{

System.out.println(a + "<- Media generala=" + a.MediaGenerala(a));

}

final String cale="date\\absolventi.csv";

SalvareAbsolventi(cale, sortata);

}

}

**//citire din csv**

try (

// FileReader fileReader = new FileReader("studenti.csv");

// BufferedReader bufferedReader = new BufferedReader(fileReader)) {

// String linie;

// while ((linie = bufferedReader.readLine()) != null) {

// String[] t = linie.split(",");

// String type = t[0].trim();

// Float weight = Float.parseFloat(t[1].trim());

// System.out.println("Weight " + weight);

// System.out.println("Type " + type);

// }

// } catch (FileNotFoundException e) {

// e.printStackTrace();

// } catch (IOException e) {

// e.printStackTrace();

// }