Ministerul Educației și Cercetării al Republicii Moldova
Centrul de Excelență în Informatică şi Tehnologii Informaţionale
Catedra Informatică II

RAPORT

STUDIU INDIVIDUAL

NR. 2

Unitatea de curs: PROGRAMAREA ORIENTATĂ PE OBIECTE

Tema: Relații între clase

Grupa: W-2223
Eleva: Belîi Alexandra
Profesor: MUSTEAȚĂ Victoria

Chişinău, 2024

Cuprins

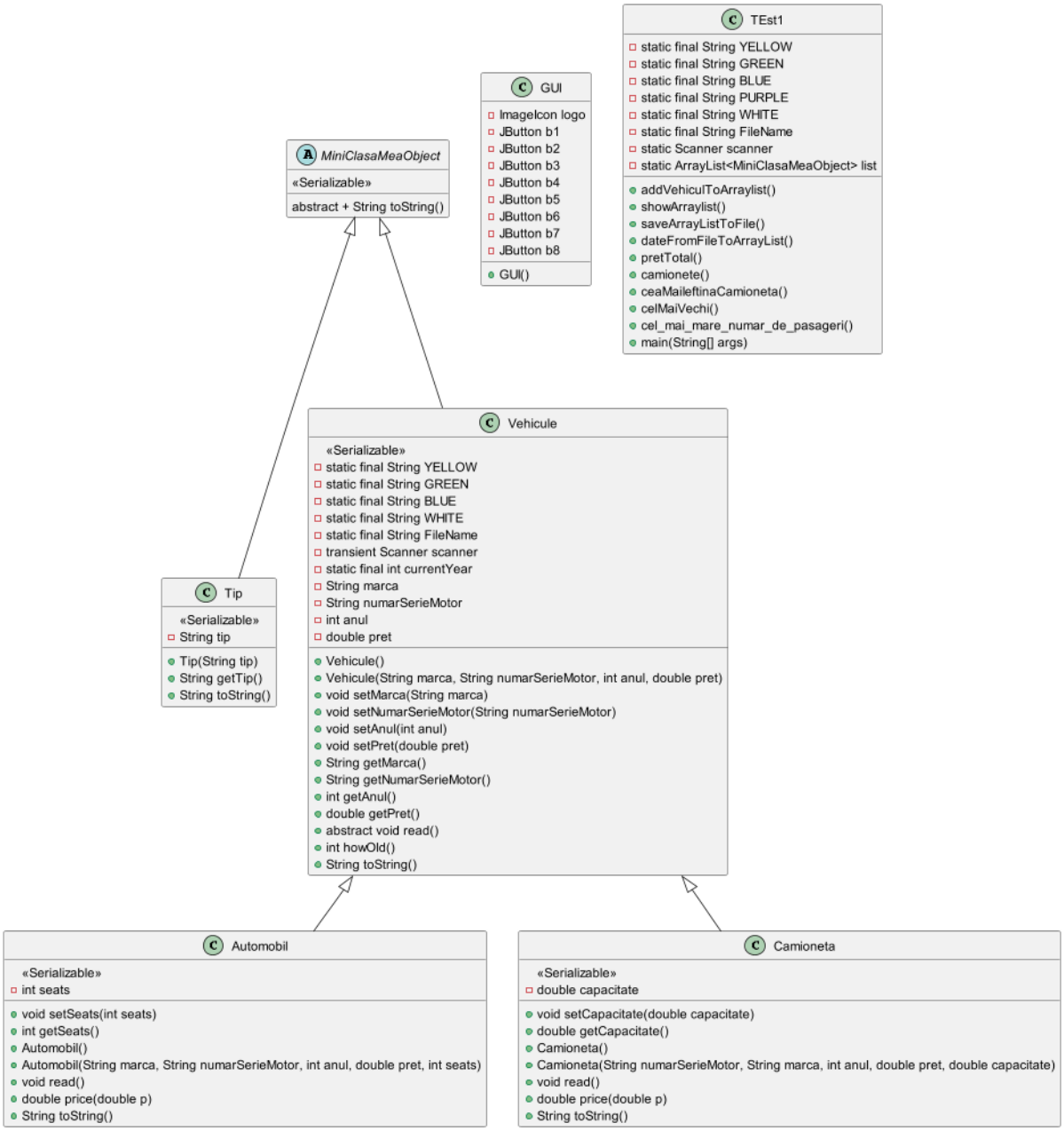
I.Enunțul problemei

Agenție Auto. Este necesar ca întro agenție auto atât să se ducă evidența datelor despre vehicule, cît și o clasificare a acestora. Elaboraţi în acest sens un program, dacă se cunoaște următoarea clasificare a produselor din compania dată: Vehicule sunt caracterizate de următoarea informație numărul de serie al motorului, marca , anul apariției și preț. În calitate de metode vor fi: citirea datelor de la tastatură, afişarea datelor la ecran, determinarea vechimii mașinii. Automobil care este derivată din Vehicule și la care se adaugă numărul de pasageri. În calitate de metode vor fi: citirea datelor de la tastatură, afişarea datelor la ecran, determinarea prețului conform formulei: pret=pret actual + 1000. Camionete derivată din Vehicule , la care se adaugă capacitate de transportare în tone. În calitate de metode vor fi: citirea datelor de la tastatură, afişarea datelor la ecran, determinarea prețului conform formulei:

$$Pret = \begin{cases} pret\ actual, dacă\ capacitatea < 3\ tone \\ pret\ actual + 1000\ lei, dacă\ 3\ tone \leq capacitate < 6\ tone \\ pret\ actual + 2000\ lei, dacă\ 6\ tone \leq capacitate < 10 \\ pret\ actual + 3000\ lei, dacă\ 10\ min \leq capacitate < 15 \\ pret\ actual + 5000\ lei\ dacă\ capacitatea \geq 15 \end{cases}$$

Fiecare dintre clase trebuie să conțină metode pentru a accesa datele respective. Scrieți și o clasă Test în care efectuați operaţii de adăugare a mașinilor, afișare a tuturor produselor din companie, determinarea prețului total al mașinilor din companie, afișați lista camionetelor, datele celea mai ieftină camionetă, automobilul cel mai vechi, automobilul cu cel mai mare număr de pasageri. Operați cu fișiere.

II. Schema proiectului



III. Codul sursă repartizat pe clase

1. Clasa MiniClasaMeaObject

```
import java.io.Serializable;
abstract public class MiniClasaMeaObject implements Serializable {}
```

2. Clasa Tip

```
import java.io.Serializable;

public class Tip extends MiniClasaMeaObject {
    public String tip;

    Tip(String tip){
        this.tip = tip;
    }

    public String getTip() {
        return tip;
    }

    @Override
    public String toString() {
        if(tip.equalsIgnoreCase("automobil"))
            return "\nTip{'" + tip + '\'' + '}' + "\nVariabile{Marca,
Numarul de serie, Anul aparitiei, Pretul, Numarul de pasageri}";
        else
            return "\nTip{'" + tip + '\'' + '}' + "\nVariabile{Marca,
Numarul de serie, Anul aparitiei, Pretul, Capacitatea transportarii}";
    }
}
```

3. Clasa Vehicule

```
import java.io.Serializable;
import java.time.LocalDate;
import java.util.Scanner;

public abstract class Vehicule extends MiniClasaMeaObject {
    public static final String YELLOW = "\u001B[33m";
    public static final String GREEN = "\u001B[32m";
    public static final String BLUE = "\033[1;94m";
    public static final String WHITE= "\033[0;97m";
    public static final String FileName = "Studiu2.ser";
    transient Scanner scanner = new Scanner(System.in); //îl instanțiem
aici și cum toate clasele vor extinde clasa aceasta, nu va fi nevoie să-l
mai instanțiem în altă parte
    static final int currentYear = LocalDate.now().getYear();
    private String marca, numarSerieMotor;
    private int anul;
    private double pret;

    //constructori
    Vehicule(){}
    Vehicule(String marca, String numarSerieMotor, int anul, double pret){
        this.marca = marca;
        this.numarSerieMotor = numarSerieMotor;
```

```
        this.anul = anul;
        this.pret = pret;
    }

    //setter
    void setMarca(String marca){
      this.marca = marca;
    }
    void setNumarSerieMotor(String numarSerieMotor){
        this.numarSerieMotor = numarSerieMotor;
    }
    void setAnul(int anul){
        this.anul = anul;
    }
    void setPret(double pret){
        this.pret = pret;
    }

    //getter
    String getMarca(){
    return marca;
    }
    String getNumarSerieMotor(){
        return numarSerieMotor;
    }
    int getAnul(){
        return anul;
    }
    double getPret(){
        return pret;
    }

    abstract void read();

    int howOld() {
        return currentYear - anul;
    }

    @Override
    public String toString(){
        return getMarca() + " " + getNumarSerieMotor() + " " + getAnul() +
" " + getPret();
    }
}
```

4. Clasa Automobil

```
import java.io.*;
import java.util.InputMismatchException;

public class Automobil extends Vehicule {

    private int seats;
    void setSeats(int seats){
        this.seats = seats;
    }
    int getSeats(){
        return seats;
    }
    //constructori
```

```
    Automobil(){}
    Automobil(String marca, String numarSerieMotor, int anul, double pret,
int seats) {
        super(marca, numarSerieMotor, anul, pret);
        this.seats = seats;
    }

    @Override
     void read() {
        int an = -1;
        try {
            System.out.print("Marca vehiculului: ");
            setMarca(scanner.nextLine());

            System.out.print("Numărul de serie al motorului: ");
            setNumarSerieMotor(scanner.nextLine());


            while (true){
                System.out.print("Anul apariției: ");
                an = scanner.nextInt();
                if(an > 0 && an <currentYear){
                setAnul(an);
                scanner.nextLine();
                break;
                }
                else {
                    System.out.println("Valoarea trebuie sa apartina
intervalului [0-an curant]");
                }
            }

            System.out.print("Preț: ");
            setPret(price(scanner.nextDouble()));
            scanner.nextLine();

            System.out.print("Număr de pasageri: ");
            setSeats(scanner.nextInt());
            scanner.nextLine();
        }
        catch (InputMismatchException e){
            System.out.println("Ai introdus un tip de date nevalabil");
            System.out.println(YELLOW + "Mai încearcă" + WHITE);
            //scanner.nextLine();
            read();
        }

    }

    double price(double p){
        return p + 1000;
    }

    @Override
    public String toString(){
        return  getMarca() + " " + getNumarSerieMotor() + " " + getAnul() +
" " + getPret() + " " + getSeats();
    }
}
```

5. Clasa Camioneta

```
import java.io.*;
import java.util.InputMismatchException;

public class Camioneta extends Vehicule {
    private double capacitate;
    public void setCapacitate(double capacitate) {
        this.capacitate = capacitate;
    }
    public double getCapacitate() {
        return capacitate;
    }

    //constructori
    Camioneta(){}

    Camioneta(String numarSerieMotor, String marca, int anul, double pret,
double capacitate){
        super(numarSerieMotor, marca, anul, pret);
        this.capacitate = capacitate;
    }

    @Override
    void read() {
        int an;
        try {
            System.out.print("Marca vehiculului: ");
            setMarca(scanner.nextLine());

            System.out.print("Numărul de serie al motorului: ");
            setNumarSerieMotor(scanner.nextLine());

            while (true){
                System.out.print("Anul apariției: ");
                an = scanner.nextInt();
                if(an > 0 && an <currentYear){
                    setAnul(an);
                    scanner.nextLine();
                    break;
                }
                else {
                    System.out.println("Valoarea trebuie sa apartina
intervalului [0-an curant]");
                }
            }

            System.out.print("Preț: ");
            setPret(price(scanner.nextDouble()));
            scanner.nextLine();

            System.out.print("Capacitatea de transportare în tone: ");
            setCapacitate(scanner.nextDouble());
            scanner.nextLine();
        }
        catch (InputMismatchException e){
            System.out.println("Ai introdus un tip de date nevalabil");
            System.out.println(YELLOW + "Mai încearcă" + WHITE);
            read();
        }
    }
```

```
    double price(double p) {
        if(capacitate < 3)
            return p;
        else if (capacitate < 6)
            return p + 1000;
        else if (capacitate < 10)
            return p + 2000;
        else if (capacitate < 15)
            return p + 3000;
        else
            return p + 5000;
    }

    @Override
    public String toString(){
        return getMarca() + " " + getNumarSerieMotor() + " " + getAnul() +
" " + getPret() + " " + getCapacitate();
    }
}
```

6. Clasa GUI

```
import javax.swing.*;
import java.awt.*;


public class GUI extends JFrame {
    ImageIcon logo = new ImageIcon("logo.png");
    private JButton b1, b2, b3, b4, b5, b6, b7, b8;
    GUI(){
        JFrame frame = new JFrame("Meniu");
        frame.setIconImage(logo.getImage());
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setResizable(false);
        frame.setSize(500, 500);
        frame.setLocationRelativeTo(null); // centreaza meniul
        frame.setLayout(new GridLayout(8, 1));
        frame.setAlwaysOnTop(true);




        b1 = new JButton("1 Adaugă un vehicul în fișier");
b1.setFocusable(false); b1.addActionListener(e ->
{TEst1.addVehiculToArraylist(); TEst1.saveArrayListToFile();});
        b2 = new JButton("2 Afișează toate produsele din companie");
b2.setFocusable(false); b2.addActionListener(e -> TEst1.showArraylist());
        b3 = new JButton("3 Prețului total al mașinilor din companie");
b3.setFocusable(false); b3.addActionListener(e -> TEst1.pretTotal());
        b4 = new JButton("4 Afișați lista camionetelor");
b4.setFocusable(false); b4.addActionListener(e -> TEst1.camionete());
        b5 = new JButton("5 Afișează datele celea mai ieftină camionetă");
b5.setFocusable(false); b5.addActionListener(e ->
TEst1.ceaMaiIeftinaCamioneta());
        b6 = new JButton("6 Afișează datele automobilul cel mai vechi");
b6.setFocusable(false); b6.addActionListener(e -> TEst1.celMaiVechi());
        b7 = new JButton("7 Afișează datele automobilul cu cel mai mare
```

```
număr de pasageri"); b7.setFocusable(false); b7.addActionListener(e ->
TEst1.cel_mai_mare_numar_de_pasageri());
        b8 = new JButton("Cancel"); b8.setFocusable(false);
b8.addActionListener(e -> System.exit(0));




        frame.add(b1);
        frame.add(b2);
        frame.add(b3);
        frame.add(b4);
        frame.add(b5);
        frame.add(b6);
        frame.add(b7);
        frame.add(b8);






        frame.setVisible(true);
    }
}
```

7. Clasa TEst1

```
import java.io.*;
import java.util.*;
import static java.awt.Color.*;
import static java.lang.Double.MAX_VALUE;

public class TEst1 {
    public static final String YELLOW = "\u001B[33m";
    public static final String GREEN = "\u001B[32m";
    public static final String BLUE = "\033[1;94m";
    public static final String PURPLE =  "\u001B[35m";
    public static final String WHITE= "\033[0;97m";
    public static final String FileName = "Studiu2.ser";

    static Scanner scanner = new Scanner(System.in);
    public static ArrayList<MiniClasaMeaObject> list = new
ArrayList<MiniClasaMeaObject>();

    public static void addVehiculToArraylist(){
        System.out.println("\nVrei să introduci un "+ YELLOW + "automobil "
+ WHITE + "sau o " + YELLOW + "camioneta" + WHITE);
        String s = scanner.nextLine();
        while (true){
            if (s.equalsIgnoreCase("automobil")){
                Tip tip = new Tip(s);
                list.add(tip);
                //System.out.println("Tipul vehiculului adaugat in array
list");
                Automobil automobil = new Automobil();
                automobil.read();
                list.add(automobil);
                System.out.println(GREEN + "\nAutomobilul a fost adăugat cu
succes." + WHITE); //în arraylist
                break;
```

```
            }
            else if (s.equalsIgnoreCase("camioneta")){
                Tip tip = new Tip(s);
                list.add(tip);
                //System.out.println("Tipul vehiculului adaugat in array
list");
                Camioneta camioneta = new Camioneta();
                camioneta.read();
                list.add(camioneta);
                System.out.println(GREEN + "\nCamioneta a fost adăugată cu
succes." + WHITE); //în arraylist
                break;
            }
            else{
            System.out.println(PURPLE + "\nMai încearcă" + WHITE);
            s = scanner.nextLine();
            }
        }

    }

    public static void showArraylist(){
        if (list.isEmpty())
            System.out.println(PURPLE + "\nNu exista vehicule." + WHITE);
        for (int i = 0; i<list.size(); i+=2){
            System.out.println(list.get(i).toString());
            System.out.println(list.get(i+1).toString());
        }
    }

    public static void saveArrayListToFile(){
        try (FileOutputStream fos = new FileOutputStream(FileName);
             ObjectOutputStream oos = new ObjectOutputStream(fos);) {

            oos.writeObject(list);
            //System.out.println("ArrayListSavedToFile");

        } catch (FileNotFoundException e) {
            System.err.println("\nFile not found : ");
            throw new RuntimeException(e);
        } catch (IOException e) {
            System.err.println("\nError while writing data : ");
            throw new RuntimeException(e);
        }
    }

    static public void dateFromFileToArrayList(){
        try(ObjectInputStream cin = new ObjectInputStream(new
FileInputStream(FileName))) {
            list = (ArrayList<MiniClasaMeaObject>)cin.readObject();
        }
        catch (EOFException eof){}
        catch (IOException | ClassNotFoundException ex ){}
    }

    static void pretTotal(){
        double total = 0;
        if (list.isEmpty())
            System.out.println(PURPLE + "\nNu exista vehicule în companie,
respectiv prețul total e 0." + WHITE);
        else{
```

```
            for (int i = 1; i<list.size(); i+=2){
                Vehicule vehicul = (Vehicule) list.get(i);
                total += vehicul.getPret();
            }
            System.out.println(BLUE + "\nPreţului total al maşinilor din
companie: " + total + WHITE);
        }
    }

    static void camionete(){
        boolean ok = false;
        if (list.isEmpty())
            System.out.println(PURPLE + "\nNu exista vehicule în companie,
respectiv nu-s nici camionete." + WHITE);
        else{
            for (int i = 0; i<list.size(); i+=2){
                Tip tip = (Tip) list.get(i);
                if (tip.getTip().equalsIgnoreCase("camioneta")){
                    ok = true; break;
                }
            }

            if (ok == false)
                System.out.println(PURPLE + "\nNu sunt camionete" + WHITE);
            else{
                System.out.println(BLUE + "\nCamionete:" + WHITE);
                System.out.println("Variabile{Marca, Numarul de serie, Anul
aparitiei, Pretul, Capacitatea transportarii}");
                for (int i = 0; i<list.size(); i+=2){
                    Tip tip = (Tip) list.get(i);
                    if (tip.getTip().equalsIgnoreCase("camioneta")){
                        System.out.println(list.get(i+1).toString());
                        ok = true;
                    }
                }
            }
        }
    }

    static void ceaMaiIeftinaCamioneta(){
        Camioneta camioneta1 =  null;
        boolean ok = false;
        double min = MAX_VALUE;
        if (list.isEmpty())
            System.out.println(PURPLE + "\nNu exista vehicule în companie,
respectiv nu-s nici camionete." + WHITE);
        else{
            for (int i = 0; i<list.size(); i+=2){
                Tip tip = (Tip) list.get(i);
                if (tip.getTip().equalsIgnoreCase("camioneta")){
                    Camioneta camioneta = (Camioneta)  list.get(i+1);
                    if(camioneta.getPret() < min){
                        min = camioneta.getPret();
                        camioneta1 = camioneta;
                    }
                    ok = true;
                }
            }
        }

            if (ok == false)
```

```
                System.out.println(PURPLE + "\nNu sunt camionete, respectiv
nu exista una cea mai ieftina" + WHITE);
            else{
                System.out.println(BLUE + "\nCea mai ieftina camioneta: " +
WHITE);
                System.out.println("Variabile{Marca, Numarul de serie, Anul
aparitiei, Pretul, Capacitatea transportarii}");
                System.out.println(camioneta1.toString());
            }
    }

    static void celMaiVechi(){
        Automobil automobil =  null;
        boolean ok = false;
        int agemax = 0;
        if (list.isEmpty())
            System.out.println(PURPLE + "\nNu exista vehicule în companie,
respectiv nu-s nici automobile." + WHITE);
        else{
            for (int i = 0; i<list.size(); i+=2){
                Tip tip = (Tip) list.get(i);
                if (tip.getTip().equalsIgnoreCase("automobil")){
                    Automobil automobil1 = (Automobil)  list.get(i+1);
                    if(automobil1.howOld() > agemax){
                        agemax = automobil1.howOld();
                        automobil = automobil1;
                    }
                    ok = true;
                }
            }

        }
        if (ok == false)
            System.out.println(PURPLE + "\nNu sunt automobile" + WHITE);
        else {
            System.out.println(BLUE + "\nCel mai vechi automobil are " +
agemax + " ani: " + WHITE);
            System.out.println("Variabile{Marca, Numarul de serie, Anul
aparitiei, Pretul, Capacitatea transportarii}");
            System.out.println(automobil.toString());
        }
    }

    static void cel_mai_mare_numar_de_pasageri(){
        Automobil automobil =  null;
        boolean ok = false;
        int mare = 0;
        if (list.isEmpty())
            System.out.println(PURPLE + "\nNu exista vehicule în companie,
respectiv nu-s nici automobile." + WHITE);
        else{
            for (int i = 0; i<list.size(); i+=2){
                Tip tip = (Tip) list.get(i);
                if (tip.getTip().equalsIgnoreCase("automobil")){
                    Automobil automobil1 = (Automobil)  list.get(i+1);
                    if(automobil1.getSeats() > mare){
                        mare = automobil1.getSeats();
                        automobil = automobil1;
                    }
                    ok = true;
                }
```

```
            }
        }

        if (ok == false)
            System.out.println(PURPLE + "\nNu sunt automobile" + WHITE);
        else{
            System.out.println(BLUE + "\nAutomobilul cu cel mai mare numar
de pasageri: " + WHITE);
            System.out.println("Variabile{Marca, Numarul de serie, Anul
aparitiei, Pretul, Numarul de pasageri}");
            System.out.println(automobil.toString());
        }
    }



    public static void main(String[] args) throws IOException {
        dateFromFileToArrayList();
        GUI gui = new GUI();

    }
}
```

IV.Teste

Încercarea de a adăuga un automobile „din viitor”.

```
automobil
Marca vehiculului: X
Numărul de serie al motorului: x
Anul apariției: 2045
Valoarea trebuie sa apartina intervalului [0-an curant]
Anul apariției: 2003
Preț: 13340
Număr de pasageri: 5
```

Afișarea toturor automobilelor din companie.

```
Tip{'automobil'}
Variabile{Marca, Numarul de serie, Anul aparitiei, Pretul, Numarul de pasageri}
Mazda M1 2000 201000.0 5

Tip{'camioneta'}
Variabile{Marca, Numarul de serie, Anul aparitiei, Pretul, Capacitatea transportarii}
BMW B7 2020 120000.0 12.0

Tip{'camioneta'}
Variabile{Marca, Numarul de serie, Anul aparitiei, Pretul, Capacitatea transportarii}
Citroen Z1 2011 12.0 9.0

Tip{'automobil'}
Variabile{Marca, Numarul de serie, Anul aparitiei, Pretul, Numarul de pasageri}
X x 2003 14340.0 5
```

Încercarea de a adăuga un autovehicul care nu e automobil sau camioneta.

```
Vrei să introduci un automobil sau o camioneta
auto

Mai încearcă
automobil
Marca vehiculului:
```

Afișarea camionetelor.

```
Camionete:
Variabile{Marca, Numarul de serie, Anul aparitiei, Pretul, Capacitatea transportarii}
BMW B7 2020 120000.0 12.0
Citroen Z1 2011 12.0 9.0
```

Afișarea celui mai vechi automobile.

```
Cel mai vechi automobil are 24 ani:
Variabile{Marca, Numarul de serie, Anul aparitiei, Pretul, Capacitatea transportarii}
Mazda M1 2000 201000.0 5
```

Afișarea datelor automobilului cu cel mai mare număr de pasageri.

```
Automobilul cu cel mai mare numar de pasageri:
Variabile{Marca, Numarul de serie, Anul aparitiei, Pretul, Numarul de pasageri}
Mazda M1 2000 201000.0 5
```

Prețul total al mașinilor din companie.

```
Prețului total al mașinilor din companie: 335352.0
```

V.Concluzie

Bună ziua, hello, здравствуйте!

Aici Alexandra care predă al doilea studiu la Java.

Deeeeeci. Luând în considerare calvarul emoțional prin care am trecut cu primul studiu mi-am zis de la început că ăsta va fi ALFEL. Ghicim ce? CHIAR a fost.

Am început studiul relativ instant după ce l-am prezentat pe primul. Și totul părea floricele până într-o zi când mi-am dat seama că va trebui să serializez obiecte din clase diferite în același fișier. Apoi cum naiba să le deserializez dacă nu voi ști de ce tip sunt.!?

Am făcut oleacă de research, nu am găsit ceva prea folositor. Am întrebat profesoara și nu am fost mega satisfăcută (tot normal, eu în 89% din viață iubesc să îmi complic existența, dacă metoda care mi se propune e prea simplă, nu-mi place!).

La un moment am crezut că va fi exact ca studiul nr.1 cu lacrimi, mental breakdown și zeci de momente în care să-mi regret decizia de a merge în IT, respectiv de a veni la CEITI(sper că această informație rămâne confidențială, dacă nu, aia e, îmi asum).

Apropo la început eram nerăbdătoare să scriu concluzia. Acum mi s-a cam luat pentru că duminică (pe 14 parcă) (deși în casă era șantier) mi-a reușit studiul. Nu zic că a fost ușor, dar parcă a mers bine. Găseam ușor ieșiri din situație. Apropo, să mor eu. Primeam eroare și NU ÎNȚELEGEAM DE UNDE. Și dau niște sout să văd unde se rupe firul și opaaaa, Scanner îmi dădea erroare și decid eu să caut și iarăși opaaaa, DA CLASA SCANNER NU IMPLEMENTEAZĂ SERIALIZAREA..... Oribil, dar asta demonstrează încă o dată că TEORIA e MEGA importantă. Efectiv detalii mici cu relevanță majoră.

Ce să zic, sunt chiar mulțumită de studiul acesta.

Dragoș a zis: „Your study is a work of art" . Plâng. Efectiv.

Acest studiu are și o scurtă aplicabilitate GUI. Nu e wow, da e ok. Nu am avut inspirație de design și deci nu m-am aventurat prea mult, dar luând în considerare că pot face schimbări oricând la aspect, dacă aveți propuneri, eu îl fac mai frumi repejor.

De data aceasta cumva nu am pierdut variantele lucrative făcând schimbări. Poate pentru că am lucrat mai atent, poate din noroc. Nu știu, totul se bazează pe circumstanțe. Acum ele au fost de partea mea... Măcar ele. Heh.

În concluzie chiar mi-a plăcut să lucrez la acest studiu. Chiar am învățat chestii noi, dar și mai important, am reușit să scriu cod cu soluții proprii gândite. Problem solving-ul chiar a fost la nivel(cred eu).

Sieg Heil, eu mă duc la somn, da, și concluzia la studiul doi e scrisă după miezul nopții.

Bibliografie

https://www.geeksforgeeks.org/serialization-in-java/

https://stackoverflow.com/questions/30013292/how-do-i-write-multiple-objects-to-the-serializable-file-and-read-them-when-the

https://howtodoinjava.com/java/collections/arraylist/serialize-deserialize-arraylist/

https://stackoverflow.com/questions/15799541/notserializableexception-on-java-scanner

https://www.youtube.com/watch?v=xk4_1vDrzzo&t=23874s

https://stackoverflow.com/questions/11600213/why-doesnt-java-lang-object-implement-the-serializable-interface

https://drive.google.com/drive/folders/1Rs3PIHuLQONXDQeJymcKFC63yl31bhjE?usp=drive_link

https://www.javatpoint.com/downcasting-with-instanceof-operator