**MINISTERUL EDUCAȚIEI AL ȘI CERCETĂRII AL REPUBLICII MOLDOVA**

**Centrul de Excelență în Informatică**

**și Tehnologii Informaționale**



**RAPORT**

**Practica de Instruire**

**SPECIALITATEA: Programarea și analiza produselor program**

**TEMA: Biletul 23**

**A elaborat elevul : Gînga Lilian**

**Grupa: P-2123**

**Conducatorul Practicii : Frunza Olga**

**Chișinău 2023**

Contents

[Sarcina 1: 3](#_Toc137762726)

[Listingul Programului: 3](#_Toc137762727)

[Rezultatele Execuției Programului 10](#_Toc137762728)

[Sarcina 2: 12](#_Toc137762729)

[Listingul Programului: 12](#_Toc137762730)

[Rezultatul Execuției Programului: 52](#_Toc137762731)

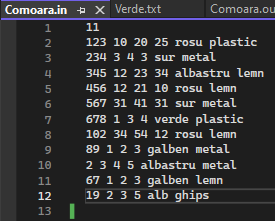
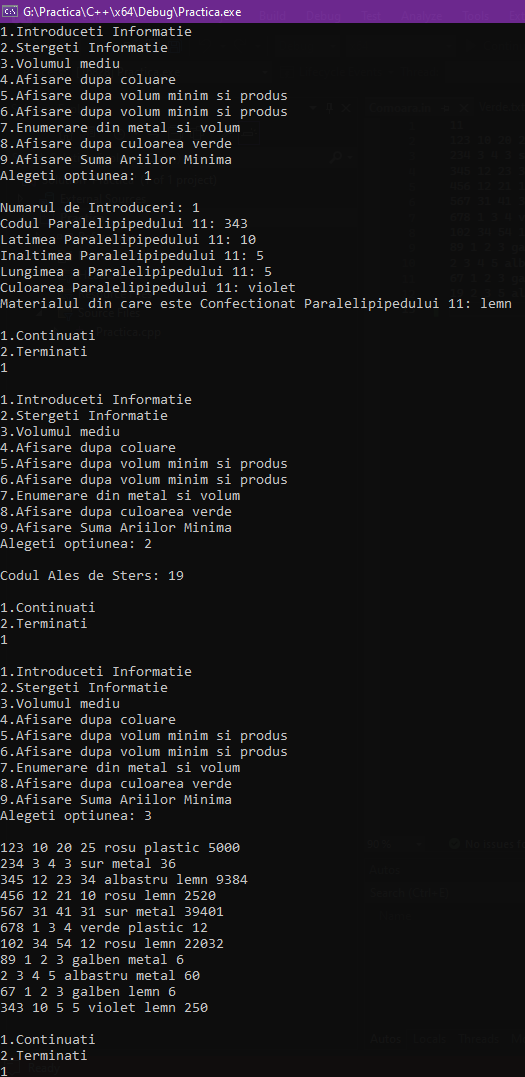
# Sarcina 1:

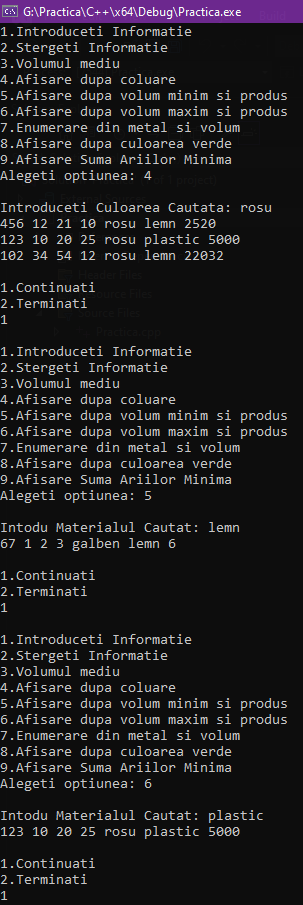
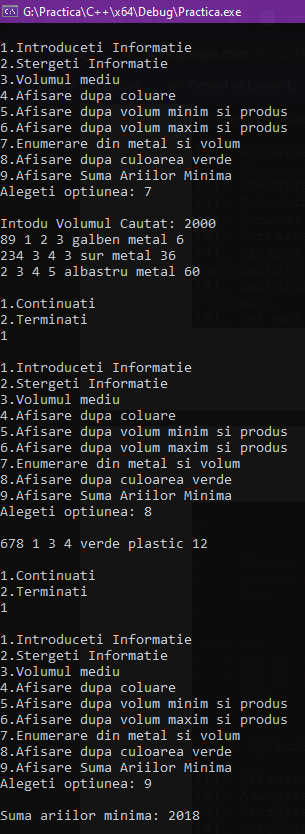
|  |
| --- |
| **Sarcina 1: *Implementați metoda de programare utilizând limbajul C++ care rezolvă următoarea sarcină:***  **23.** **Paralelipipede.**  Informaţiile  despre un set de ***n*** paralelipipede dreptunghiulare sunt înregistrate în  fişierul textual  **Comoare.in**.    Pe prima linie a  fişierului de intrare  se află  naturalul ***n***, iar pe următoarele ***n*** linii ale lui se conţin atributele dreptunghiurilor respective, fiecare  linie   ***i*** **+1** ( **1*≤ i ≤* *n***)  conţinând următoarele date, separate prin câte un spaţiu: **CPi** (Codul paralelipipedului i); cele 3 dimensiuni: ***a***, ***b***şi ***c***; **Culoare** (**r**oşie, **g**albenă, **v**erde, **a**lbastră), **Material**  ( Materialul din care e confecţionat: **l**emn, **m**etal, **c**arton, **p**lastic).  Să se creeze o aplicație care să realizeze, la solicitarea utilizatorului, următoarele prescripţii:  (*Design-ul aplicației vă aparține*).  **1**) Înregistrează un nou paralelipiped;  **2**) Exclude  din fişierul **Comoara.in** linia ce corespunde codului figurii, introdus de la tastatură;  **3** ) Determină volumul mediu  al  paralelipipedelor în studiu;  **4**) Afişează pe ecran atributele figurilor, de culoarea indicată de la tastatură, în ordinea crescătoare a volumelor acestora; sortarea datelor se va realiza prin metoda bulelor;  **5**) Determină figura de volum minim şi, respectiv, cea de volum maxim, confecţionate din materialul cu denumirea introdusă de la tastatură;  **6)** Enumără figurile confecţionate din metal, volumul căror nu depăşeşte valoarea introdusă de la tastatură;  **7**) Creează fişierul textual **Verde.txt** în care se vor copia acele linii ale fişierului de intrare, care corespund  figurilor de culoare verde;  **8**) **Rezolvă problema.** Vânătorii de comori au descoperit în una din încăperile închise ale unui castel medieval ***n***lingouri de aur de dimensiuni distincte. Fiecare lingou ***i*** **(1 ≤ *i* ≤ *n*** ) reprezintă un paralelipiped dreptunghiular cu dimensiunile ***xi , yi , zi***. Pentru a scoate lingourile la lumina zilei, vânătorii de comori trebuie să perforeze într-un zid de piatră unul sau mai multe orificii dreptunghiulare, care nu au puncte de tangenţă. Un lingou poate fi scos printr-un orificiu dreptunghiular doar atunci când lăţimea şi înălţimea orificiului sunt egale sau mai mari ca lăţimea şi înălţimea uneia din feţele dreptunghiulare ale paralelipipedului. Evident, lingoul poate fi rotit in orice fel.  Pentru a-şi uşura munca, vânătorii de comori doresc ca suma ariilor orificiilor ce trebuie perforate să fie cât mai mică.  **Sarcină**. Elaboraţi un subprogram care, cunoscând dimensiunile lingourilor de aur, calculează valoarea minimă ***S*** a sumei ariilor orificiilor de perforat, ce vor fi suficiente pentru a scoate toate lingourile.  **Date de intrare.** Fişierul text **Comoara.in**.  **Date de ieşire.** Fişierul text **Comoara.out** va conţine pe o singură linie numărul întreg ***S***− valoarea minimă a sumei ariilor orificiilor de perforat. |

## Listingul Programului:

|  |
| --- |
| //Practica.cpp  #include <iostream>  #include "logic.h"  using namespace std;  int main() {  menu();  } |
| // logic.h  #pragma once  #include <iostream>  #include <fstream>  #include <vector>  #include <algorithm>  using namespace std;  fstream file("Comoara.in");  ofstream out("Comoara.out");  ofstream grout("Verde.txt");  struct Paralelipiped {  string cod, culoare, material;  double x{}, y{}, z{}, volum{}, arie{};  } p[1000];  vector<string> colours{ "albastru", "rosu", "verde", "galben", "violet", "portocaliu", "cafeniu", "sur", "gri", "negru", "alb", "roz", "auriu", "argintiu", "cupru"};  vector<string> materials{"lemn", "carton", "metal", "plastic", "ghips"};  int n;  void menu();  void continueMessage();  void avgVolume();  void readFile() {  file >> n;  for (int index = 0; index < n; index++) {  file >> p[index].cod >> p[index].x >> p[index].y >> p[index].z >> p[index].culoare >> p[index].material;  }  avgVolume();  }  bool checkColour(string colour) {  bool check = false;  if (find(colours.begin(), colours.end(), colour) != colours.end()) check = true;  return check;  }  bool checkMaterial(string material) {  bool check = false;  if (find(materials.begin(), materials.end(), material) != materials.end()) check = true;  return check;  }  bool checkCode(string cod) {  bool check = false;  for (int index = 0; index < n; index++) {  if (p[index].cod == cod){  check = true;  break;  }  }  return check;  }  void insert() {  readFile();  ofstream fileOut("Comoara.in", ios::trunc);  int add;  cout << "Numarul de Introduceri: "; cin >> add;  for (int index = n; index < n + add; index++) {  do {  if (!p[index].cod.empty()) if (checkCode(p[index].cod)) cout << "Codul deja existent" << endl;  cout << "Codul Paralelipipedului " << index << ": ";  cin >> p[index].cod;  } while (checkCode(p[index].cod));  int i = 0;  do {  if(i != 0) if (p[index].x <= 0) cout << "Latimea nu poate fi 0 sau negativa" << endl;  cout << "Latimea Paralelipipedului " << index << ": ";  cin >> p[index].x;  i++;  } while (p[index].x <= 0);  i = 0;  do {  if (i != 0) if (p[index].y <= 0) cout << "Inaltimea nu poate fi 0 sau negativa" << endl;  cout << "Inaltimea Paralelipipedului " << index << ": ";  cin >> p[index].y;  i++;  } while (p[index].y <= 0);    i = 0;  do {  if (i != 0) if (p[index].z <= 0) cout << "Lungimea nu poate fi 0 sau negativa" << endl;  cout << "Lungimea a Paralelipipedului " << index << ": ";  cin >> p[index].z;  i++;  } while (p[index].z <= 0);  do {  if (!p[index].culoare.empty()) if (!checkMaterial(p[index].culoare)) cout << "Nu este o Culaore" << endl;  cout << "Culoarea Paralelipipedului " << index << ": ";  cin >> p[index].culoare;  } while (!checkColour(p[index].culoare));  do {  if (!p[index].material.empty()) if (!checkMaterial(p[index].material)) cout << "Nu este un Material" << endl;  cout << "Materialul din care este Confectionat Paralelipipedului " << index << ": ";  cin >> p[index].material;  } while (!checkMaterial(p[index].material));  }  n += add;  fileOut << n << endl;  for (int index = 0; index < n; index++) {  fileOut << p[index].cod << " " << p[index].x << " " << p[index].y << " " << p[index].z << " " << p[index].culoare << " " << p[index].material << endl;  }    continueMessage();  }  void deleteInfo() {  readFile();  ofstream fileOut("Comoara.in", ios::trunc);  string cod;  cout << "Codul Ales de Sters: "; cin >> cod;  int foundIndex=0;  for (int index = 0; index < n; index++) {  if (p[index].cod == cod) {  foundIndex = index;  break;  }  }  if (foundIndex == 0) cout << "Codul nu a fost gsit";  else {  if (foundIndex == n - 1) n--;  else {  for (int index = foundIndex; index < n - 1; index++) {  p[index].cod = p[index + 1].cod;  p[index].x = p[index + 1].x;  p[index].y = p[index + 1].y;  p[index].z = p[index + 1].z;  p[index].culoare = p[index + 1].culoare;  p[index].material = p[index + 1].material;  p[index].volum = p[index + 1].volum;  }  n--;  }  fileOut << n << endl;  for (int index = 0; index < n; index++) {  fileOut << p[index].cod << " " << p[index].x << " " << p[index].y << " " << p[index].z << " " << p[index].culoare << " " << p[index].material << endl;  }  }  continueMessage();  }  void avgVolume() {  for (int index = 0; index < n; index++) {  p[index].volum = p[index].x \* p[index].y \* p[index].z;  }  }  void writeAvg() {  avgVolume();  for (int index = 0; index < n; index++) {  cout << p[index].cod << " " << p[index].x << " " << p[index].y << " " << p[index].z << " " << p[index].culoare << " " << p[index].material << " " << p[index].volum << endl;  }  continueMessage();  }  void bubbleSort() {  avgVolume();  double aux;  string auxs;  for (int index = 0; index < n-1; index++) {  for (int jindex = 0; jindex < n-1; jindex++) {  if (p[jindex].volum > p[jindex + 1].volum) {  auxs = p[jindex + 1].cod;  p[jindex + 1].cod = p[jindex].cod;  p[jindex].cod = auxs;  aux = p[jindex + 1].x;  p[jindex + 1].x = p[jindex].x;  p[jindex].x = aux;  aux = p[jindex + 1].y;  p[jindex + 1].y = p[jindex].y;  p[jindex].y = aux;  aux = p[jindex + 1].z;  p[jindex + 1].z = p[jindex].z;  p[jindex].z = aux;  auxs = p[jindex + 1].culoare;  p[jindex + 1].culoare = p[jindex].culoare;  p[jindex].culoare = auxs;  auxs = p[jindex + 1].material;  p[jindex + 1].material = p[jindex].material;  p[jindex].material = auxs;  aux = p[jindex + 1].volum;  p[jindex + 1].volum = p[jindex].volum;  p[jindex].volum = aux;  }  }  }  }  void writeByColour() {  bubbleSort();  string colour;  do {  if(!colour.empty()) if(!checkColour(colour)) cout << "Nu este o Culaore" << endl;  cout << "Introduceti Culoarea Cautata: "; cin >> colour;  } while (!checkColour(colour));  for (int index = 0; index < n; index++) {  if (p[index].culoare == colour) {  cout << p[index].cod << " " << p[index].x << " " << p[index].y << " " << p[index].z << " " << p[index].culoare << " " << p[index].material << " " << p[index].volum << endl;  }  }  continueMessage();  }  double getMin(string material) {  double min = 999999;  for (int index = 0; index < n; index++) {  if (p[index].material == material) {  if (min > p[index].volum) min = p[index].volum;  }  }  return min;  }  double getMax(string material) {  double max = 0;  for (int index = 0; index < n; index++) {  if (p[index].material == material) {  if (max < p[index].volum) max = p[index].volum;  }  }  return max;  }  void minVolume() {  avgVolume();  string material;  cout << "Intodu Materialul Cautat: "; cin >> material;  double min = getMin(material);  for (int index = 0; index < n; index++) {  if (p[index].material == material && p[index].volum == min) {  cout << p[index].cod << " " << p[index].x << " " << p[index].y << " " << p[index].z << " " << p[index].culoare << " " << p[index].material << " " << p[index].volum << endl;  }  }  continueMessage();  }  void maxVolume() {  avgVolume();  string material;  do {  if (!material.empty()) if (!checkMaterial(material)) cout << "Nu este un Material" << endl;  cout << "Intodu Materialul Cautat: "; cin >> material;  } while (!checkMaterial(material));  double max = getMax(material);  for (int index = 0; index < n; index++) {  if (p[index].material == material && p[index].volum == max) {  cout << p[index].cod << " " << p[index].x << " " << p[index].y << " " << p[index].z << " " << p[index].culoare << " " << p[index].material << " " << p[index].volum << endl;  }  }  continueMessage();  }  void writeLessVolume() {  avgVolume();  double volum;  cout << "Intodu Volumul Cautat: "; cin >> volum;  for (int index = 0; index < n; index++) {  if (p[index].material == "metal") {  if (p[index].volum <= volum) {  cout << p[index].cod << " " << p[index].x << " " << p[index].y << " " << p[index].z << " " << p[index].culoare << " " << p[index].material << " " << p[index].volum << endl;  }  }  }  continueMessage();  }  void outGreen(string colour, int index) {  grout << p[index].cod << " " << p[index].x << " " << p[index].y << " " << p[index].z << " " << p[index].culoare << " " << p[index].material << " " << p[index].volum << endl;  }  void writeGreen() {  for (int index = 0; index < n; index++) {  if (p[index].culoare == "verde") {  cout << p[index].cod << " " << p[index].x << " " << p[index].y << " " << p[index].z << " " << p[index].culoare << " " << p[index].material << " " << p[index].volum << endl;  outGreen(p[index].culoare, index);  }  }  continueMessage();  }  double sumaMin(int index) {  if (index < 0) return 0;  p[index].arie = p[index].x \* p[index].y;  if (p[index].x \* p[index].z < p[index].arie) p[index].arie = p[index].x \* p[index].z;  if (p[index].y \* p[index].z < p[index].arie) p[index].arie = p[index].y \* p[index].z;  return p[index].arie + sumaMin(index - 1);  }  void problem() {  readFile();  double suma;  suma = sumaMin(n-1);  cout << "Suma ariilor minima: " << suma << endl;  out << suma;  continueMessage();  }  void menu() {  readFile();  int index;  cout << "1.Introduceti Informatie" << endl;  cout << "2.Stergeti Informatie" << endl;  cout << "3.Volumul mediu" << endl;  cout << "4.Afisare dupa coluare" << endl;  cout << "5.Afisare dupa volum minim si produs" << endl;  cout << "6.Afisare dupa volum minim si produs" << endl;  cout << "7.Enumerare din metal si volum" << endl;  cout << "8.Afisare dupa culoarea verde" << endl;  cout << "9.Afisare Suma Ariilor Minima" << endl;  cout << "Alegeti optiunea: ";  cin >> index;  cout << endl;  switch (index) {  case 1: insert();  break;  case 2: deleteInfo();  break;  case 3: writeAvg();  break;  case 4: writeByColour();  break;  case 5: minVolume();  break;  case 6: maxVolume();  break;  case 7: writeLessVolume();  break;  case 8: writeGreen();  break;  case 9: problem();  break;  default: "Optiunea nu exista";  break;  }  }  void continueMessage() {  int index;  cout << endl << "1.Continuati" << endl;  cout << "2.Terminati" << endl;  cin >> index;  cout<<endl;  switch (index) {  case 1: menu();  break;  default: cout << "";  break;  }  } |

## Rezultatele Execuției Programului





# Sarcina 2:

|  |
| --- |
| **Sarcina 2: *Scrieți un program Java care modelează următoarea activitate:***  Creați un produs program pentru evidența activității unei companii de închiriere de mașini. Compania are mai multe filiale în țară. Fiecare filială este caracterizată prin nume, adresă, telefon, persoană de contact, etc.  Produsul program gestionează activitatea a unei companii care oferă spre închiriere mașini de diferite tipuri. Programul reține informații despre mașinile companiei, companiile cu care compania dată are contracte de colaborare (de exemplu, garaje), mașinile închiriate, veniturile companiei și bineînțeles date despre clienții companiei.  Mașinile sunt descrise prin date precum: producătorul, modelul, anul de fabricație, mărimea motorului, tipul de combustibil, numărul de pasageri, numărul de înmatriculare, prețul de cumpărare, data cumpărării, prețul de închiriere și detalii privind asigurarea mașinii.  Toate reparațiile importante asupra mașinilor companiei sunt făcute de companii cu care colaborează. Unele companii solicită plata serviciilor de service imediat după ce reparația a fost făcută, altele acceptă plata în rate a serviciilor.  Compania deține un portofoliu de clienți destul de stabil. Pentru clienții privilegiați oferă reduceri la închirierea de mașini. Acești clienți au de asemenea posibilitatea de a rezerva o mașină din timp. O mașină poate fi închiriată pentru o perioadă de timp de la o zi la un an. Plata pentru închirierea unei mașini se poate face cash sau cu credit card. Se acceptă orice tip de card. Datele importante despre clienți sunt numele, adresa, numărul de telefon, seria permisului de conducere, etc.  Produsul program trebuie  să ținută  evidența clară a fiecărei mașini, a celor închiriate, a celor aflate în reparații, etc. Se păstrează de asemenea evidența tuturor veniturilor și cheltuielilor firmei: cumpărarea de noi mașini, închirierea de mașini, cheltuieli de reparații, vânzarea unor mașini mai vechi din parcul auto (compania preferă să nu păstreze în parcul auto o mașină mai mult de un an), taxe de asigurare a mașinilor, etc. |

## Listingul Programului:

|  |
| --- |
| // Main.java  public class Main {  public static void main(String[] args) {  CompanieInchiriere compInchit = new CompanieInchiriere();  compInchit.menu();  compInchit.createFrame();  } } |
| // CompanieInchiriere.java  import java.awt.\*; import java.util.\*; import java.io.\*; import javax.swing.\*; import java.awt.event.\*;  public class CompanieInchiriere {  private JFrame frame;  private final JFrame carFrame = new JFrame("MASINI");  private final JFrame clientFrame = new JFrame("CLIENTI");  private final JFrame repairFrame = new JFrame("COMPANII DE REPARERE");  private final JFrame cityFrame = new JFrame("FILIALE");  private final JFrame setCar = new JFrame();  private final JFrame setClient = new JFrame();  private final JFrame setRepair = new JFrame();  private final JFrame setCity = new JFrame();  private final JFrame setClientCar = new JFrame();  private final JFrame setRepairCar = new JFrame();  private final JFrame getCar = new JFrame();  private final JFrame getClient = new JFrame();  private final JFrame getRepair = new JFrame();  private final JFrame getCity = new JFrame();  private final JFrame getClientCar = new JFrame();  private final JFrame getRepairCar = new JFrame();  private final JFrame verifyCar = new JFrame();  private final JFrame convertBuy = new JFrame();  private final JFrame convertRent = new JFrame();  private final JFrame sumClientPaid = new JFrame();  private final JFrame numberCars = new JFrame();  private JLabel label;  private final JPanel panel = new JPanel();   private final JButton next = new JButton("Next");  private final JButton back = new JButton("Back");  private final JButton home = new JButton("Home");  private final JButton end = new JButton("End");  private final JButton carBtn = new JButton("Masini");  private final JButton clientBtn = new JButton("Clienti");  private final JButton repairBtn = new JButton("Companii de Reparatii");  private final JButton cityBtn = new JButton("Filiale");  private JButton setBtn;  private JButton getBtn;  private final JButton clientCarBtn = new JButton("Introduceti Informatii Despre Clienti si Masina");  private final JButton getClientCarBtn = new JButton("Afisati Informatia Despre Clienti si Masina");  private final JButton verifyBtn = new JButton("Verifica Masini");  private final JButton convertBuyBtn = new JButton("Converteste Pretul de Cumparare");  private final JButton convertRentBtn = new JButton("Converteste Pretul de Inchiriere");  private final JButton sumBtn = new JButton("Sumele achitate de Clienti");  private final JButton numBtn = new JButton("Afisare numarul Masini la Reparatie");  private final JButton repairCarBtn = new JButton("Introduceti Informatii Despre Companiile de Reparare si Masini");  private final JButton getRepairCarBtn = new JButton("Afisati Informatii Despre Companiile de Reparare si Masini");   private ActionListener[] actionListeners = new ActionListener[10000];  private JComboBox<String> buyCurrency;  private JComboBox<String> rentCurrency;   private final JLabel company = new JLabel("Producator:");  private final JLabel model = new JLabel("Model:");  private final JLabel fuel = new JLabel("Combustibil:");  private final JLabel buyDate = new JLabel("Data Cumparare:");  private final JLabel status = new JLabel("Situatie:");  private final JLabel yearCreated = new JLabel("Anul Fabricarii:");  private final JLabel nrPassangers = new JLabel("Numar Pasageri:");  private final JLabel code = new JLabel("Cod de Inmatriculare:");  private final JLabel capacity = new JLabel("Marimea Motorului:");  private final JLabel buyPrice = new JLabel("Pretul de Cumparare:");  private final JLabel rentPrice = new JLabel("Pretul de Inchiriere:");  private final JTextField companyText = new JTextField();  private final JTextField modelText = new JTextField();  private final JTextField fuelText = new JTextField();  private final JTextField buyDateText = new JTextField();  private final JTextField statusText = new JTextField();  private final JTextField yearCreatedText = new JTextField();  private final JTextField nrPassangersText = new JTextField();  private final JTextField codeText = new JTextField();  private final JTextField capacityText = new JTextField();  private final JTextField buyPriceText = new JTextField();  private final JTextField rentPriceText = new JTextField();   private JLabel name = new JLabel("Nume:");  private JLabel surname = new JLabel("Prenume:");  private final JLabel adress = new JLabel("Adresa:");  private final JLabel priviledge = new JLabel("Privilegiu:");  private final JLabel rentPeriod = new JLabel("Perioada Inchirierii:");  private final JLabel pay = new JLabel("Cu ce Plateste:");  private final JLabel phone = new JLabel("Numar de Telefon:");  private final JLabel license = new JLabel("Licenta de Condus:");  private final JTextField nameText = new JTextField();  private final JTextField surnameText = new JTextField();  private final JTextField adressText = new JTextField();  private final JTextField priviledgeText = new JTextField();  private final JTextField rentPeriodText = new JTextField();  private final JTextField payText = new JTextField();  private final JTextField phoneText = new JTextField();  private final JTextField licenseText = new JTextField();   private final JLabel carStatus = new JLabel("Achizitie: ");  private final JLabel period = new JLabel("Perioada Achizitiei: ");  private final JTextField carStatusText = new JTextField();  private final JTextField periodText = new JTextField();   private final JLabel nameContact = new JLabel("Numele Persoanei de Contact:");  private final JTextField nameContactText = new JTextField();   private final JLabel payment = new JLabel("Plata: ");  private final JTextField paymentText = new JTextField();  private JTextArea textArea;   private final JComboBox<String> filterBox = new JComboBox<>();  private final JComboBox<String> filteringBox = new JComboBox<>();  private JScrollPane scrollPane;   private final JButton convert = new JButton("Convertiti");  private final JTextField converted = new JTextField();   private final File companie = new File("Companie.txt");  private final Scanner scan = new Scanner(System.*in*);  private final ArrayList<Masina> listaMasini = new ArrayList<>();  private final ArrayList<Client> listaClienti = new ArrayList<>();  private final ArrayList<Filiala> listaFiliale = new ArrayList<>();  private final ArrayList<CompanieReparati> listaReparatii = new ArrayList<>();  private final ArrayList<ClientMasina> listaClientMasini = new ArrayList<>();  private final ArrayList<ReparareMasina> listaCompRepMasini = new ArrayList<>();  private final ArrayList<Double> achitare = new ArrayList<>();  private final Vector<String> currencyList = new Vector<>();  private final int[] num = {0, 0, 0, 0, 0};  private final int[] count = new int[10000];   public CompanieInchiriere() {readFile();}   private void readFile(){  try{  Scanner readFile = new Scanner(companie);  while(readFile.hasNext()) {  String delimiter = readFile.next();  switch (delimiter) {  case "m" -> autoList(readFile);  case "c" -> clientList(readFile);  case "cr" -> repairList(readFile);  case "f" -> citiesList(readFile);  case "lcm" -> clientAutoList(readFile);  case "lrm" -> repairAutoList(readFile);  }  }  }catch(FileNotFoundException fnfe){  System.*out*.println("File Not Found");  }  }   public void createFrame(){  createComboBox();  carFrame.dispose();  clientFrame.dispose();  repairFrame.dispose();  cityFrame.dispose();   panel.removeAll();   frame = new JFrame("MENU");  createBtn();  label = new JLabel("Alegeti Optiunea");  label.setBounds(154, 0, 200, 50);  label.setHorizontalAlignment(SwingConstants.*CENTER*);   frame.setSize(415, 230);  panel.setLayout(new GridLayout(2, 4));  panel.add(carBtn);  panel.add(clientBtn);  panel.add(label);  panel.add(repairBtn);  panel.add(cityBtn);  frame.add(panel);  frame.setResizable(false);   frame.setVisible(true);  frame.setLocationRelativeTo(null);  frame.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  }   private void createBtn(){  carBtn.setBounds(0, 50, 200, 70);  clientBtn.setBounds(200, 50, 200, 70);  repairBtn.setBounds(0, 120, 200, 70);  cityBtn.setBounds(200, 120, 200, 70);   carBtn.addActionListener(e-> carMenu(frame));  clientBtn.addActionListener(e -> clientelMenu(frame));  repairBtn.addActionListener(e-> repairMenu(frame));  cityBtn.addActionListener(e -> citiesMenu(frame));  }   private void createComboBox(){  if(!(buyCurrency == null)) buyCurrency.removeAllItems();  currencyList.add("MDL");  currencyList.add("EUR");  currencyList.add("USD");   buyCurrency = new JComboBox<>(currencyList);  buyCurrency.setBounds(300, 239, 75, 25);  rentCurrency = new JComboBox<>(currencyList);  rentCurrency.setBounds(300, 264, 75, 25);   }   *//start of class* static class ClientMasina{  private String nume, prenume, achizitie;  private int codMasina, perioada;  ClientMasina(){}   public ClientMasina(String nume, String prenume, String achizitie, int codMasina, int perioada) {  this.nume = nume;  this.prenume = prenume;  this.achizitie = achizitie;  this.codMasina = codMasina;  this.perioada = perioada;  }   public void setMasina(File file){  Scanner scan = new Scanner(System.*in*);  this.nume = scan.nextLine();  this.prenume = scan.nextLine();  this.achizitie = scan.nextLine();  this.codMasina = scan.nextInt();  this.perioada = scan.nextInt();  try{  FileWriter fileWriter = new FileWriter(file, true);  BufferedWriter bufferedWriter = new BufferedWriter(fileWriter);  bufferedWriter.write("lcm "+this.nume+" "+this.prenume+" "+this.achizitie+" "+this.codMasina+" "+this.perioada+"\n");  bufferedWriter.close();  }catch(IOException e){  e.printStackTrace();  }  }   public String getNume() {  return nume;  }   public void setNume(String nume) {  this.nume = nume;  }   public String getPrenume() {  return prenume;  }   public void setPrenume(String prenume) {  this.prenume = prenume;  }   public String getAchizitie() {  return achizitie;  }   public void setAchizitie(String achizitie) {  this.achizitie = achizitie;  }   public int getCodMasina() {  return codMasina;  }   public void setCodMasina(int codMasina) {  this.codMasina = codMasina;  }   public int getPerioada() {  return perioada;  }   public void setPerioada(int perioada) {  this.perioada = perioada;  }   @Override  public String toString() {  return "ClientMasina{" +  "nume='" + nume + '\'' +  ", prenume='" + prenume + '\'' +  ", achizitie='" + achizitie + '\'' +  ", codMasina=" + codMasina +  ", perioada=" + perioada +  '}';  }  }   static class ReparareMasina{  private String denumire;  private int codMasina, perioada;   public ReparareMasina() {}   public ReparareMasina(String denumire, int codMasina, int perioada) {  this.denumire = denumire;  this.codMasina = codMasina;  this.perioada = perioada;  }   public void setReparareMasina(File file){  Scanner scan = new Scanner(System.*in*);  this.denumire = scan.nextLine();  this.codMasina = scan.nextInt();  this.perioada = scan.nextInt();  try{  FileWriter fileWriter = new FileWriter(file, true);  BufferedWriter bufferedWriter = new BufferedWriter(fileWriter);  bufferedWriter.write("lrm "+this.denumire+" "+this.codMasina+" "+this.perioada+"\n");  bufferedWriter.close();  }catch(IOException e){  e.printStackTrace();  }  }   public String getDenumire() {  return denumire;  }   public void setDenumire(String denumire) {  this.denumire = denumire;  }   public int getCodMasina() {  return codMasina;  }   public void setCodMasina(int codMasina) {  this.codMasina = codMasina;  }   public int getPerioada() {  return perioada;  }   public void setPerioada(int perioada) {  this.perioada = perioada;  }   @Override  public String toString() {  return "ReparareMasina{" +  "denumire='" + denumire + '\'' +  ", codMasina=" + codMasina +  ", perioada=" + perioada +  '}';  }  }  *//end of classs* private void autoList(Scanner scan){  Masina car = new Masina(scan.next(), scan.next(), scan.next(), scan.next(), scan.next(), scan.nextInt(), scan.nextInt(), scan.nextInt(), scan.nextDouble(), scan.nextDouble(), scan.next(), scan.nextDouble(), scan.next());  listaMasini.add(car);  }   private void clientList(Scanner scan){  Client client = new Client(scan.next(), scan.next(), scan.next(), scan.next(), scan.next(), scan.next(), scan.nextInt(), scan.nextInt());  listaClienti.add(client);  }   private void repairList(Scanner scan){  CompanieReparati compRepar = new CompanieReparati(scan.next(), scan.next());  listaReparatii.add(compRepar);  }   private void citiesList(Scanner scan){  Filiala filiala = new Filiala(scan.next(), scan.next(), scan.next(), scan.next(), scan.nextInt());  listaFiliale.add(filiala);  }   private void clientAutoList(Scanner scan){  ClientMasina clientMasina = new ClientMasina(scan.next(), scan.next(), scan.next(), scan.nextInt(), scan.nextInt());  listaClientMasini.add(clientMasina);  }   private void repairAutoList(Scanner scan) {  ReparareMasina reparareMasina = new ReparareMasina(scan.next(), scan.nextInt(), scan.nextInt());  listaCompRepMasini.add(reparareMasina);  }   public void getCompDetails(){  System.*out*.print("Ce doresti sa aflii:\n1.Masini\n2.Clienti\n3.Companiie de Reparatii\n4.Filiale\nOptiune:");  int index = scan.nextByte();   switch(index){  case 1 -> returnAuto();  case 2 -> returnClient();  case 3 -> returnRepair();  case 4 -> returnCities();  }  }   public void introduAuto(){  Masina car = new Masina();  car.setAuto(companie);  listaMasini.add(car);  }   public void introduClient(){  Client client = new Client();  client.setClient(companie);  listaClienti.add(client);  }   public void introduCompRep(){  CompanieReparati compRep = new CompanieReparati();  compRep.setCompRep(companie);  listaReparatii.add(compRep);  }   public void introduFiliala(){  Filiala filiala = new Filiala();  filiala.setFiliala(companie);  listaFiliale.add(filiala);  }   public void introduClientMasina(){  ClientMasina clientMasina = new ClientMasina();  clientMasina.setMasina(companie);  listaClientMasini.add(clientMasina);  }   public void introduRepMasina(){  ReparareMasina reparareMasina = new ReparareMasina();  reparareMasina.setReparareMasina(companie);  listaCompRepMasini.add(reparareMasina);  }   public void returnAuto() {  for(Masina entity : listaMasini){  System.*out*.println(entity.toString());  }  }   public void returnClient() {  for(Client entity : listaClienti){  System.*out*.println(entity.toString());  }  }   public void returnClientCar(){  for(ClientMasina entity : listaClientMasini){  System.*out*.println(entity.toString());  }  }   public void returnRepair() {  for(CompanieReparati entity : listaReparatii){  System.*out*.println(entity.toString());  }  }   public void returnRepairCar() {  for(ReparareMasina entity : listaCompRepMasini){  System.*out*.println(entity.toString());  }  }  public void returnCities() {  for(Filiala entity : listaFiliale){  System.*out*.println(entity.toString());  }  }   public void sumaAchitata(){  double suma = 0, convert = 1, procent = 1;  for(ClientMasina client : listaClientMasini){  for(Masina masina : listaMasini){  achitare.add(0.0);  if(client.getCodMasina() == masina.getNumarInmatriculare()) {   if(masina.getPrCumpValuta().equals("EUR") || masina.getPrInchValuta().equals("EUR")) convert = 19.0;  else if(masina.getPrCumpValuta().equals("USD") || masina.getPrInchValuta().equals("USD")) convert = 17.8;   if(listaClienti.get(checkClient(client.getNume())).getPrivilegiu().equals("advanced")) procent = 0.2;   if (checkClient(client.getNume())+1 > 0) suma = achitare.get(checkClient(client.getNume()));  if (client.getAchizitie().equals("cumparare")) suma += masina.getPretCumparare() \* convert \* procent;  if (client.getAchizitie().equals("inchiriere")) suma += masina.getPretCumparare() \* convert \* procent;   achitare.set(checkClient(client.getNume()), suma);  }  }  }  }   public int checkClient(String name){  int index = 0;  for(Client client : listaClienti){  if(client.getNume().equals(name)) break;  index++;  }  return index;  }   public void verifAuto(){  for(Masina car : listaMasini){  switch (car.getSituatie()) {  case "vanduta" -> num[0]++;  case "inchiriata" -> num[1]++;  case "reparatie" -> num[2]++;  case "depozitat" -> num[3]++;  case "parcata" -> num[4]++;  }  } }   public void autoRepair(){  for(ReparareMasina comp : listaCompRepMasini){  for(Masina masina : listaMasini){  if(comp.getCodMasina() == masina.getNumarInmatriculare()) {  count[checkRepair(comp.getDenumire())]++;  }  }  }  }   public int checkRepair(String name){  int index = 0;  for(CompanieReparati rep : listaReparatii){  if(rep.getDenumire().equals(name)) break;  index++;  }  return index;  }   public void menu(){  System.*out*.print("1.Masini\n2.Clienti\n3.Companiie de Reparatii\n4.Filiale\nOptiune: ");  int index = scan.nextInt();  switch (index){  case 1 -> masinaMenu();  case 2 -> clientMenu();  case 3 -> companieReparatiMenu();  case 4 -> filialaMenu();  }  }   private void masinaMenu() {  System.*out*.print("1.Introduceti Informatii despre Masini\n2.Afisare informatie despre Masini\n3.Verifica Masini\n4.Converteste Pretul de Cumparare\n5.Converteste Pretul de Inchiriere\nOptiune: ");  int index = scan.nextInt();  switch (index){  case 1 -> introduAuto();  case 2 -> returnAuto();  case 3 -> {  verifAuto();  System.*out*.printf("Vadute: %d\nInchiriate: %d\nLa Reparatie: %d\nIn Parc: %d\nDepozitate: %d\n", num[0],num[1],num[2],num[3],num[4]);  }  default -> {  System.*out*.print("Introduceti indexul masinii: ");  int indexL = scan.nextInt();  listaMasini.get(indexL).menu(index);  }  }  }   private void clientMenu() {  System.*out*.print("1.Introduceti Informatii despre Clienti\n2.Introduceti Informatii despre Clienti si Masini\n3.Afisare informatie despre Clienti\n4.Afisare informatie despre Clienti si Masini\n5.Sumele achitate de Clienti\nOptiune: ");  int index = scan.nextInt();  switch (index){  case 1 -> introduClient();  case 2 -> introduClientMasina();  case 3 -> returnClient();  case 4 -> returnClientCar();  case 5 -> {  sumaAchitata();  for(int i = 0; i < achitare.size(); i++){  if(i >= listaClientMasini.size()) break;  System.*out*.println(listaClientMasini.get(i).getNume() +" "+ listaClientMasini.get(i).getPrenume() +": "+ achitare.get(checkClient(listaClientMasini.get(i).getNume())));  }  }  }  }   private void companieReparatiMenu() {  System.*out*.print("1.Introduceti Informatii despre Compania de Reparatii\n2.Introduceti Informatii despre Compania de Reparatii si Masini\n3.Afisare informatie despre Compania de Reparatii\n4.Afisare informatie despre Compania de Reparatii si Masini\n5.Afisare numarul Masini la Reparatie\nOptiune: ");  int index = scan.nextInt();  switch (index){  case 1 -> introduCompRep();  case 2 -> introduRepMasina();  case 3 -> returnRepair();  case 4 -> returnRepairCar();  case 5 -> {  autoRepair();  for(int i = 0; i < listaCompRepMasini.size(); i++){  if(count[checkRepair(listaCompRepMasini.get(i).getDenumire())] == 0) break;  System.*out*.println(listaReparatii.get(checkRepair(listaCompRepMasini.get(i).getDenumire())).getDenumire() +": "+ count[checkRepair(listaCompRepMasini.get(i).getDenumire())]);  }  }  }  }   private void filialaMenu() {  System.*out*.print("1.Introduceti Informatii despre Filiala\n2.Afisare informatie despre Filiala\nOptiune: ");  int index = scan.nextInt();  switch (index){  case 1 -> introduFiliala();  case 2 -> returnCities();  }  }   *//GUI MENUS* private void createOtherBtn(JFrame Jframe, int width, int height) {  next.setBounds(width - (width - 7), height - 85, width / 5, height / 10);  home.setBounds(width - (width - 107), height - 85, width / 5, height / 10);  back.setBounds(width - (width - 207), height - 85, width / 5, height / 10);  end.setBounds(width - (width - 307), height - 85, width / 5, height / 10);   actionListeners = next.getActionListeners();  for (ActionListener listener : actionListeners) {  next.removeActionListener(listener);  }  actionListeners = back.getActionListeners();  for (ActionListener listener : actionListeners) {  back.removeActionListener(listener);  }  actionListeners = home.getActionListeners();  for (ActionListener listener : actionListeners) {  home.removeActionListener(listener);  }  actionListeners = end.getActionListeners();  for (ActionListener listener : actionListeners) {  end.removeActionListener(listener);  }   home.addActionListener(e -> {  panel.removeAll();  frame.dispose();  createFrame();  });  end.addActionListener(e -> System.*exit*(0));  back.addActionListener(e -> *Jframe*.dispose());   Jframe.add(next);  Jframe.add(home);  Jframe.add(end);  Jframe.add(back);  }   private void carMenu(JFrame Jframe) {  panel.removeAll();   convertRent.dispose();  convertBuy.dispose();  frame.dispose();  clientFrame.dispose();  repairFrame.dispose();  cityFrame.dispose();   setBtn = new JButton("Introduceti Informatii despre Masini");  getBtn = new JButton("Afisare informatie despre Masini");   setBtn.setBounds(0, 0, 200, 70);  getBtn.setBounds(200, 0, 200, 70);  verifyBtn.setBounds(0, 70, 200, 70);  convertBuyBtn.setBounds(200, 70, 200, 70);  convertRentBtn.setBounds(0, 140, 200, 70);   actionListeners = setBtn.getActionListeners();  for (ActionListener listener : actionListeners) {  setBtn.removeActionListener(listener);  }  actionListeners = getBtn.getActionListeners();  for (ActionListener listener : actionListeners) {  getBtn.removeActionListener(listener);  }  actionListeners = verifyBtn.getActionListeners();  for (ActionListener listener : actionListeners) {  verifyBtn.removeActionListener(listener);  }  actionListeners = convertBuyBtn.getActionListeners();  for (ActionListener listener : actionListeners) {  convertBuyBtn.removeActionListener(listener);  }  actionListeners = convertRentBtn.getActionListeners();  for (ActionListener listener : actionListeners) {  convertRentBtn.removeActionListener(listener);  }   setBtn.addActionListener(e -> setCar());  getBtn.addActionListener(e -> filterCar());  verifyBtn.addActionListener(e -> verifyCar());  convertBuyBtn.addActionListener(e -> filterBuyLicense());  convertRentBtn.addActionListener(e -> filterRentLicense());   panel.setLayout(new GridLayout(4, 4));  carFrame.setSize(415, 320);   panel.add(setBtn);  panel.add(getBtn);  panel.add(verifyBtn);  panel.add(convertBuyBtn);  panel.add(convertRentBtn);   createOtherBtn(carFrame, carFrame.getWidth(), carFrame.getHeight());  next.addActionListener(e -> clientelMenu(*Jframe*));  back.addActionListener(e -> citiesMenu(*Jframe*));   carFrame.add(panel);   carFrame.setResizable(false);  carFrame.setVisible(true);  carFrame.setLocationRelativeTo(null);  carFrame.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  }   private void clientelMenu(JFrame Jframe) {  panel.removeAll();   clientFrame.remove(next);  clientFrame.remove(back);   if(Jframe != null) Jframe.dispose();  frame.dispose();  carFrame.dispose();  repairFrame.dispose();  cityFrame.dispose();   setBtn = new JButton("Introduceti Informatii despre Clienti");  getBtn = new JButton("Afisare informatie despre Clienti");   setBtn.setBounds(0, 0, 200, 70);  clientCarBtn.setBounds(200, 0, 200, 70);  getBtn.setBounds(0, 70, 200, 70);  getClientCarBtn.setBounds(200,70,200,70);  sumBtn.setBounds(0, 140, 200, 70);   actionListeners = convertRentBtn.getActionListeners();  for (ActionListener listener : actionListeners) {  convertRentBtn.removeActionListener(listener);  }  actionListeners = clientCarBtn.getActionListeners();  for (ActionListener listener : actionListeners) {  clientCarBtn.removeActionListener(listener);  }  actionListeners = getBtn.getActionListeners();  for (ActionListener listener : actionListeners) {  getBtn.removeActionListener(listener);  }  actionListeners = getClientCarBtn.getActionListeners();  for (ActionListener listener : actionListeners) {  getClientCarBtn.removeActionListener(listener);  }  actionListeners = sumBtn.getActionListeners();  for (ActionListener listener : actionListeners) {  sumBtn.removeActionListener(listener);  }   setBtn.addActionListener(e -> setClient());  clientCarBtn.addActionListener(e -> setClientCar());  getBtn.addActionListener(e -> filterClient());  getClientCarBtn.addActionListener(e -> filterClientCar());  sumBtn.addActionListener(e -> clientSumPaid());   clientFrame.setSize(415, 325);   createOtherBtn(clientFrame, clientFrame.getWidth(), clientFrame.getHeight());   next.addActionListener(e -> repairMenu(*Jframe*));  back.addActionListener(e -> carMenu(*Jframe*));   clientFrame.add(setBtn);  clientFrame.add(clientCarBtn);  clientFrame.add(getBtn);  clientFrame.add(getClientCarBtn);  clientFrame.add(sumBtn);  clientFrame.add(panel);   clientFrame.setResizable(false);  clientFrame.setVisible(true);  clientFrame.setLocationRelativeTo(null);  clientFrame.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  }   private void repairMenu(JFrame Jframe) {  repairFrame.remove(next);  repairFrame.remove(back);   panel.setLayout(null);  panel.removeAll();   if(Jframe != null) Jframe.dispose();  Jframe.dispose();  carFrame.dispose();  clientFrame.dispose();  cityFrame.dispose();   setBtn = new JButton("Introduceti Informatii despre Compania de Reparatii");  getBtn = new JButton("Afisare informatie despre Compania de Reparatii");   setBtn.setBounds(0, 0, 200, 70);  repairCarBtn.setBounds(200, 0, 200, 70);  getBtn.setBounds(0, 70, 200, 70);  getRepairCarBtn.setBounds(200, 70, 200, 70);  numBtn.setBounds(0, 140, 200, 70);   actionListeners = setBtn.getActionListeners();  for(ActionListener listener : actionListeners){  setBtn.removeActionListener(listener);  }  actionListeners = repairCarBtn.getActionListeners();  for(ActionListener listener : actionListeners){  repairCarBtn.removeActionListener(listener);  }  actionListeners = getBtn.getActionListeners();  for(ActionListener listener : actionListeners){  getBtn.removeActionListener(listener);  }  actionListeners = getRepairCarBtn.getActionListeners();  for(ActionListener listener : actionListeners){  getRepairCarBtn.removeActionListener(listener);  }  actionListeners = numBtn.getActionListeners();  for(ActionListener listener : actionListeners){  numBtn.removeActionListener(listener);  }   setBtn.addActionListener(e -> setRepair());  repairCarBtn.addActionListener(e -> setRepairCar());  getBtn.addActionListener(e -> filterRepair());  getRepairCarBtn.addActionListener(e -> filterRepairCar());  numBtn.addActionListener(e -> numberCars());   repairFrame.setSize(415, 325);   createOtherBtn(repairFrame, repairFrame.getWidth(), repairFrame.getHeight());   next.addActionListener(e -> citiesMenu(*Jframe*));  back.addActionListener(e -> clientelMenu(*Jframe*));   panel.add(setBtn);  panel.add(repairCarBtn);  panel.add(getBtn);  panel.add(getRepairCarBtn);  panel.add(numBtn);  repairFrame.add(panel);   repairFrame.setResizable(false);  repairFrame.setVisible(true);  repairFrame.setLocationRelativeTo(null);  repairFrame.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  }   private void citiesMenu(JFrame Jframe) {  cityFrame.remove(next);  cityFrame.remove(back);   panel.removeAll();  panel.setLayout(null);   if(Jframe != null) Jframe.dispose();  frame.dispose();  clientFrame.dispose();  repairFrame.dispose();  carFrame.dispose();   setBtn = new JButton("Introduceti Informatii despre Filiala");  getBtn = new JButton("Afisare informatie despre Filiala");   setBtn.setBounds(0, 0, 200, 70);  getBtn.setBounds(200, 0, 200, 70);   actionListeners = setBtn.getActionListeners();  for(ActionListener listener : actionListeners){  setBtn.removeActionListener(listener);  }  actionListeners = getBtn.getActionListeners();  for(ActionListener listener : actionListeners){  getBtn.removeActionListener(listener);  }   setBtn.addActionListener(e -> setCity());  getBtn.addActionListener(e -> filterCity());   cityFrame.setSize(415, 200);   createOtherBtn(cityFrame, cityFrame.getWidth(), cityFrame.getHeight());   next.addActionListener(e -> carMenu(*Jframe*));  back.addActionListener(e -> repairMenu(*Jframe*));   panel.add(setBtn);  panel.add(getBtn);  cityFrame.add(panel);   cityFrame.setResizable(false);  cityFrame.setVisible(true);  cityFrame.setLocationRelativeTo(null);  cityFrame.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  }   private void setCar(){  panel.removeAll();  panel.setLayout(null);  carFrame.dispose();   createComboBox();   setCar.setTitle("Introduceti Informatii Despre Masini");  setCar.setSize(415, 375);   company.setBounds(25, 0, 100, 50);  model.setBounds(40, 25, 100, 50);  fuel.setBounds(25, 50, 100, 50);  buyDate.setBounds(15, 75, 100, 50);  status.setBounds(35, 100, 100, 50);  yearCreated.setBounds(15, 125, 100, 50);  nrPassangers.setBounds(15, 150, 100, 50);  code.setBounds(0, 175, 150, 50);  capacity.setBounds(8, 200, 150, 50);  buyPrice.setBounds(0, 225, 150, 50);  rentPrice.setBounds(8, 250, 150, 50);   companyText.setBounds(130, 14, 245, 25);  modelText.setBounds(130, 39, 245, 25);  fuelText.setBounds(130, 64, 245, 25);  buyDateText.setBounds(130, 89, 245, 25);  statusText.setBounds(130, 114, 245, 25);  yearCreatedText.setBounds(130, 139, 245, 25);  nrPassangersText.setBounds(130, 164, 245, 25);  codeText.setBounds(130, 189, 245, 25);  capacityText.setBounds(130, 214, 245, 25);  buyPriceText.setBounds(130, 239, 170, 25);  buyPriceText.setEditable(true);  rentPriceText.setBounds(130, 264, 170, 25);   panel.add(company);  panel.add(model);  panel.add(fuel);  panel.add(buyDate);  panel.add(status);  panel.add(yearCreated);  panel.add(nrPassangers);  panel.add(code);  panel.add(capacity);  panel.add(buyPrice);  panel.add(rentPrice);   panel.add(companyText);  panel.add(modelText);  panel.add(fuelText);  panel.add(buyDateText);  panel.add(statusText);  panel.add(yearCreatedText);  panel.add(nrPassangersText);  panel.add(codeText);  panel.add(capacityText);  panel.add(buyPriceText);  panel.add(rentPriceText);   panel.add(buyCurrency);  panel.add(rentCurrency);   createOtherBtn(setCar, setCar.getWidth(), setCar.getHeight());  next.addActionListener(e -> {  try{  FileWriter fileWriter = new FileWriter(companie, true);  BufferedWriter bufferedWriter = new BufferedWriter(fileWriter);   if(!(companyText.getText().equals("") || companyText.getText().equals("") || fuelText.getText().equals("") ||  buyDateText.getText().equals("") || statusText.getText().equals("") || yearCreatedText.getText().equals("") ||  nrPassangersText.getText().equals("") || codeText.getText().equals("") || capacityText.getText().equals("") ||  buyPriceText.getText().equals("") || rentPriceText.getText().equals(""))){  bufferedWriter.write("m " + companyText.getText() + " " + modelText.getText() + " " + fuelText.getText() + " " +  buyDateText.getText() + " " + statusText.getText() + " " + yearCreatedText.getText() + " " + nrPassangersText.getText() + " " +  codeText.getText() + " " + capacityText.getText() + " " + buyPriceText.getText() + " " + buyCurrency.getSelectedItem() + " " +  rentPriceText.getText() + " " + rentCurrency.getSelectedItem() + "\n");  }  companyText.setText("");  modelText.setText("");  fuelText.setText("");  buyDateText.setText("");  statusText.setText("");  yearCreatedText.setText("");  nrPassangersText.setText("");  codeText.setText("");  capacityText.setText("");  buyPriceText.setText("");  rentPriceText.setText("");   bufferedWriter.close();  fileWriter.close();  }catch(IOException ioe){  ioe.printStackTrace();  }  });  setCar.add(panel);   back.addActionListener(e -> carMenu(setCar));   setCar.setVisible(true);  setCar.setResizable(false);  setCar.setLocationRelativeTo(null);  setCar.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  }   private void setClient(){  panel.removeAll();  panel.setLayout(null);  clientFrame.dispose();   setClient.setTitle("Introduceti Informatii Despre Clienti");  setClient.setSize(415, 315);   name.setBounds((setClient.getWidth()\*10)/100, 0, 100, 50);  surname.setBounds(30, 25, 100, 50);  adress.setBounds(35, 50, 100, 50);  priviledge.setBounds(30, 75, 100, 50);  rentPeriod.setBounds(10, 100, 125, 50);  pay.setBounds(20, 125, 100, 50);  phone.setBounds(10, 150, 125, 50);  license.setBounds(7, 175, 150, 50);   nameText.setBounds(130, 14, 245, 25);  surnameText.setBounds(130, 39, 245, 25);  adressText.setBounds(130, 64, 245, 25);  priviledgeText.setBounds(130, 89, 245, 25);  rentPeriodText.setBounds(130, 114, 245, 25);  payText.setBounds(130, 139, 245, 25);  phoneText.setBounds(130, 164, 245, 25);  licenseText.setBounds(130, 189, 245, 25);   panel.add(name);  panel.add(surname);  panel.add(adress);  panel.add(priviledge);  panel.add(rentPeriod);  panel.add(pay);  panel.add(phone);  panel.add(license);  panel.add(nameText);  panel.add(surnameText);  panel.add(adressText);  panel.add(priviledgeText);  panel.add(rentPeriodText);  panel.add(payText);  panel.add(phoneText);  panel.add(licenseText);   createOtherBtn(setClient, setClient.getWidth(), setClient.getHeight());  next.addActionListener(e -> {  try {  FileWriter fileWriter = new FileWriter(companie, true);  BufferedWriter bufferedWriter = new BufferedWriter(fileWriter);   if (!(nameText.getText().equals("") || surnameText.getText().equals("") || adressText.getText().equals("") ||  priviledgeText.getText().equals("") || rentPeriodText.getText().equals("") || payText.getText().equals(""))) {  bufferedWriter.write("c " + nameText.getText() + " " + surnameText.getText() + " " + adressText.getText()  + " " + priviledgeText.getText() + " " + rentPeriodText.getText() + " " + payText.getText() + " " +  phoneText.getText() + " " + licenseText.getText() + "\n");  }   nameText.setText("");  surnameText.setText("");  adressText.setText("");  priviledgeText.setText("");  rentPeriodText.setText("");  payText.setText("");   bufferedWriter.close();  fileWriter.close();  } catch (IOException ioe) {  ioe.printStackTrace();  }  });  setClient.add(panel);   back.addActionListener(e -> clientelMenu(setClient));   setClient.setVisible(true);  setClient.setResizable(false);  setClient.setLocationRelativeTo(null);  setClient.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  }   private void setClientCar(){  panel.removeAll();  panel.setLayout(null);  clientFrame.dispose();   setClientCar.setTitle("Introduceti Informatii Despre Clienti si Masini");  setClientCar.setSize(415, 250);   name.setBounds((setClientCar.getWidth()\*10)/100, 0, 100, 50);  surname.setBounds(30, 25, 100, 50);  carStatus.setBounds(35, 50, 100, 50);  code.setBounds(0, 75, 150, 50);  period.setBounds(10, 100, 150, 50);   nameText.setBounds(130, 14, 245, 25);  surnameText.setBounds(130, 39, 245, 25);  carStatusText.setBounds(130, 64, 245, 25);  codeText.setBounds(130, 89, 245, 25);  periodText.setBounds(130, 114, 245, 25);   panel.add(name);  panel.add(surname);  panel.add(carStatus);  panel.add(code);  panel.add(period);  panel.add(nameText);  panel.add(surnameText);  panel.add(carStatusText);  panel.add(codeText);  panel.add(periodText);   createOtherBtn(setClientCar, setClientCar.getWidth(), setClientCar.getHeight());  next.addActionListener(e -> {  try{  FileWriter fileWriter = new FileWriter(companie, true);  BufferedWriter bufferedWriter = new BufferedWriter(fileWriter);   if(!(nameText.getText().equals("") || surnameText.getText().equals("") || carStatusText.getText().equals("")  || codeText.getText().equals("") || periodText.getText().equals(""))){  bufferedWriter.write("lcm " + nameText.getText() + " " + surnameText.getText() + " " + carStatusText.getText() + " " +  codeText.getText() + " " + periodText.getText() + "\n");  }   nameText.setText("");  surnameText.setText("");  carStatusText.setText("");  codeText.setText("");  periodText.setText("");   bufferedWriter.close();  fileWriter.close();  }catch(IOException ioe){  ioe.printStackTrace();  }  });   setClientCar.add(panel);   back.addActionListener(e -> clientelMenu(setClientCar));   setClientCar.setVisible(true);  setClientCar.setResizable(false);  setClientCar.setLocationRelativeTo(null);  setClientCar.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  }   private void setRepair(){  panel.removeAll();  panel.setLayout(null);  repairFrame.dispose();   setRepair.setTitle("Introduceti Informatii Despre Companii de Reparare");   name = new JLabel("Denumire: ");   name.setBounds(26, 0, 100, 50);  payment.setBounds(35, 25, 100, 50);  nameText.setBounds(130, 14, 245, 25);  paymentText.setBounds(130, 39, 245, 25);   setRepair.setSize(415, 200);   createOtherBtn(setRepair, setRepair.getWidth(), setRepair.getHeight());  next.addActionListener(e -> {  try{  FileWriter fileWriter = new FileWriter(companie, true);  BufferedWriter bufferedWriter = new BufferedWriter(fileWriter);   if(!(nameText.getText().equals("") || paymentText.getText().equals("") )){  bufferedWriter.write("cr " + nameText.getText() + " " + paymentText.getText() + "\n");  }   name.setText("");  paymentText.setText("");   bufferedWriter.close();  fileWriter.close();  }catch(IOException ioe){  ioe.printStackTrace();  }  });   panel.add(name);  panel.add(payment);  panel.add(nameText);  panel.add(paymentText);  setRepair.add(panel);   back.addActionListener(e -> repairMenu(setRepair));   setRepair.setVisible(true);  setRepair.setResizable(false);  setRepair.setLocationRelativeTo(null);  setRepair.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  }   private void setRepairCar(){  panel.removeAll();  panel.setLayout(null);  repairFrame.dispose();   setRepairCar.setTitle("Introduceti Informatii Despre Companie de Reparare si Masini");  setRepairCar.setSize(415, 375);   name = new JLabel("Denumire: ");   name.setBounds(26, 0, 100, 50);  code.setBounds(0, 25, 150, 50);  period.setBounds(10, 50, 150, 50);  nameText.setBounds(130, 14, 245, 25);  codeText.setBounds(130, 39, 245, 25);  periodText.setBounds(130, 64, 245, 25);   createOtherBtn(setRepairCar, setRepairCar.getWidth(), setRepairCar.getHeight());  next.addActionListener(e -> {  try{  FileWriter fileWriter = new FileWriter(companie, true);  BufferedWriter bufferedWriter = new BufferedWriter(fileWriter);   if(!(nameText.getText().equals("") || codeText.getText().equals("") || periodText.getText().equals(""))){  bufferedWriter.write("lrm " + nameText.getText() + " " + codeText.getText() + " " + periodText.getText() + "\n");  }   nameText.setText("");  codeText.setText("");  periodText.setText("");   bufferedWriter.close();  fileWriter.close();  }catch(IOException ioe){  ioe.printStackTrace();  }  });   panel.add(name);  panel.add(code);  panel.add(period);  panel.add(nameText);  panel.add(codeText);  panel.add(periodText);  setRepairCar.add(panel);   back.addActionListener(e -> repairMenu(setRepairCar));   setRepairCar.setVisible(true);  setRepairCar.setResizable(false);  setRepairCar.setLocationRelativeTo(null);  setRepairCar.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  }   private void setCity(){  panel.removeAll();  panel.setLayout(null);  cityFrame.dispose();   setCity.setSize(415, 375);   name = new JLabel("Denumire: ");  surname = new JLabel("Prenumele Persoanei de Contact: ");   name.setBounds(30, 0, 100, 50);  adress.setBounds(40, 25, 100, 50);  nameContact.setBounds(10, 50, 200, 50);  surname.setBounds(0, 75, 200, 50);  phone.setBounds(15, 100, 150, 50);  nameText.setBounds(130, 14, 245, 25);  adressText.setBounds(130, 39, 245, 25);  nameContactText.setBounds(200, 64, 175, 25);  surnameText.setBounds(200, 89, 175, 25);  phoneText.setBounds(130, 114, 245, 25);   createOtherBtn(setCity, setCity.getWidth(), setCity.getHeight());  next.addActionListener(e -> {  try{  FileWriter fileWriter = new FileWriter(companie, true);  BufferedWriter bufferedWriter = new BufferedWriter(fileWriter);   if(!(nameText.getText().equals("") || adressText.getText().equals("") || nameContactText.getText().equals("") ||  surnameText.getText().equals("") || phoneText.getText().equals("") )){  bufferedWriter.write("f " + nameText.getText() + " " + adressText.getText() + " " + nameContactText.getText() + " " +  surnameText.getText() + " " + phoneText.getText() + "\n");  }   nameText.setText("");  adressText.setText("");  nameContactText.setText("");  surnameText.setText("");  phoneText.setText("");   bufferedWriter.close();  fileWriter.close();  }catch(IOException ioe){  ioe.printStackTrace();  }  });   panel.add(name);  panel.add(adress);  panel.add(nameContact);  panel.add(surname);  panel.add(phone);  panel.add(nameText);  panel.add(adressText);  panel.add(nameContactText);  panel.add(surnameText);  panel.add(phoneText);  setCity.add(panel);   back.addActionListener(e -> citiesMenu(setCity));   setCity.setVisible(true);  setCity.setResizable(false);  setCity.setLocationRelativeTo(null);  setCity.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  }   private void getCar(String item) {  panel.removeAll();  panel.setLayout(null);  carFrame.dispose();   getCar.setTitle("Afisarea Informatii Despre Masini");  getCar.setSize(300, 375);   getCar.setLayout(new FlowLayout());   textArea = new JTextArea(15, 23);  textArea.setEditable(false);   scrollPane = new JScrollPane(textArea);  scrollPane.setVerticalScrollBarPolicy(JScrollPane.*VERTICAL\_SCROLLBAR\_ALWAYS*);  scrollPane.setHorizontalScrollBarPolicy(JScrollPane.*HORIZONTAL\_SCROLLBAR\_AS\_NEEDED*);   panel.setLayout(new BorderLayout());  panel.add(scrollPane, BorderLayout.*CENTER*);   for (Masina masina : listaMasini) {  if (item == null || item.equals("All") || item.equals(masina.getProducator()) ||  item.equals(masina.getModel()) || item.equals(masina.getCombustibil()) || item.equals(masina.getdataCumparare())  || item.equals(masina.getSituatie()) || item.equals(String.*valueOf*(masina.getAnulFabricat()))  || item.equals(String.*valueOf*(masina.getNumarPasageri())) || item.equals(String.*valueOf*(masina.getNumarInmatriculare()))  || item.equals(String.*valueOf*(masina.getMarimeaMotorului())) || item.equals(String.*valueOf*(masina.getPretCumparare()))  || item.equals(String.*valueOf*(masina.getPretInchiriere()))) {  textArea.append("Producator: " + masina.getProducator() +  "\nModel: " + masina.getModel() +  "\nTip Combustibil: " + masina.getCombustibil() +  "\nData de Cumparare: " + masina.getdataCumparare() +  "\nSituatie: " + masina.getSituatie() +  "\nAnul Fabricarii: " + masina.getAnulFabricat() +  "\nNumarul de Pasageri: " + masina.getNumarPasageri() +  "\nNumarul de Inmatriculare: " + masina.getNumarInmatriculare() +  "\nMarimea Motorului: " + masina.getMarimeaMotorului() +  "\nPretul de Cumparare: " + masina.getPretCumparare() + " " + masina.getPrCumpValuta() +  "\nPretul de Inchiriere: " + masina.getPretInchiriere() + " " + masina.getPrInchValuta() + "\n");  }   }   createOtherBtn(getCar, getCar.getWidth(), getCar.getHeight());  next.addActionListener(e -> {  verifyCar();  });  back.addActionListener(e -> carMenu(getCar));  getCar.add(panel);  getCar.add(filterBox);  getCar.add(filteringBox);   getCar.setVisible(true);  getCar.setResizable(false);  getCar.setLocationRelativeTo(null);  getCar.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  }   private void filterCar(){  filterBox.setBounds(0, 0, 10, 10);  filterBox.addItem("All");  filterBox.addItem("Producator");  filterBox.addItem("Model");  filterBox.addItem("Tip Combustibil");  filterBox.addItem("Data de Cumparare");  filterBox.addItem("Situatie");  filterBox.addItem("Anul Fabricarii");  filterBox.addItem("Numarul de Pasager");  filterBox.addItem("Numar de Inmatriculare");  filterBox.addItem("Marime Motorului");  filterBox.addItem("Pretul de Cumparare");  filterBox.addItem("Pretul de Inchiriere");   filterBox.addActionListener(e -> filterCar((String) Objects.*requireNonNull*(filterBox.getSelectedItem())));   getCar("All");  }   private void filterCar(String item){  filteringBox.removeAllItems();  filteringBox.setBounds(10, 0, 10, 10);  filteringBox.addItem("All");   if(item.equals("Producator")){  for(Masina car : listaMasini){  filteringBox.addItem(car.getProducator());  }  }   if(item.equals("Model")){  for(Masina car : listaMasini){  filteringBox.addItem(car.getModel());  }  }   if(item.equals("Tip Combustibil")){  for(Masina car : listaMasini){  filteringBox.addItem(car.getCombustibil());  }  }   if(item.equals("Data de Cumparare")){  for(Masina car : listaMasini){  filteringBox.addItem(car.getdataCumparare());  }  }   if(item.equals("Situatie")){  for(Masina car : listaMasini){  filteringBox.addItem(car.getSituatie());  }  }   if(item.equals("Anul Fabricarii")){  for(Masina car : listaMasini){  filteringBox.addItem(String.*valueOf*(car.getAnulFabricat()));  }  }   if(item.equals("Numarul de Pasager")){  for(Masina car : listaMasini){  filteringBox.addItem(String.*valueOf*(car.getNumarPasageri()));  }  }   if(item.equals("Numar de Inmatriculare")){  for(Masina car : listaMasini){  filteringBox.addItem(String.*valueOf*(car.getNumarInmatriculare()));  }  }   if(item.equals("Marime Motorului")){  for(Masina car : listaMasini){  filteringBox.addItem(String.*valueOf*(car.getMarimeaMotorului()));  }  }   if(item.equals("Pretul de Cumparare")){  for(Masina car : listaMasini){  filteringBox.addItem(String.*valueOf*(car.getPretCumparare()));  }  }   if(item.equals("Pretul de Inchiriere")){  for(Masina car : listaMasini){  filteringBox.addItem(String.*valueOf*(car.getPretInchiriere()));  }  }   filteringBox.addActionListener(e -> getCar((String) filteringBox.getSelectedItem()));   getCar("All");  }   private void getClient(String item) {  panel.removeAll();  panel.setLayout(null);  clientFrame.dispose();   getClient.setTitle("Afisarea Informatii Despre Client");  getClient.setSize(300, 375);   getClient.setLayout(new FlowLayout());   textArea = new JTextArea(15, 23);  textArea.setEditable(false);   scrollPane = new JScrollPane(textArea);  scrollPane.setVerticalScrollBarPolicy(JScrollPane.*VERTICAL\_SCROLLBAR\_ALWAYS*);  scrollPane.setHorizontalScrollBarPolicy(JScrollPane.*HORIZONTAL\_SCROLLBAR\_AS\_NEEDED*);   panel.setLayout(new BorderLayout());  panel.add(scrollPane, BorderLayout.*CENTER*);   for (Client client : listaClienti) {  if (item == null || item.equals("All") || item.equals(client.getNume()) ||  item.equals(client.getPrenume()) || item.equals(client.getAdresa())  || item.equals(client.getPrivilegiu()) || item.equals(String.*valueOf*(client.getPerioadaInchirierii())) ||  item.equals(client.getPlata()) || item.equals(String.*valueOf*(client.getTelefon())) ||  item.equals(String.*valueOf*(client.getPermisConducere()))) {  textArea.append("Nume: " + client.getNume() +  "\nPrenume: " + client.getPrenume() +  "\nAdresa: " + client.getAdresa() +  "\nPrivilegiu: " + client.getPrivilegiu() +  "\nPerioada de Inchiriere: " + client.getPerioadaInchirierii() +  "\nPlata: " + client.getPlata() +  "\nNumar de Telefon: " + client.getTelefon() +  "\nPermis de Conducere: " + client.getPermisConducere() + "\n");  }   }   createOtherBtn(getClient, getClient.getWidth(), getClient.getHeight());  next.addActionListener(e -> getClientCar("All"));  back.addActionListener(e -> clientelMenu(getClient));  getClient.add(panel);  getClient.add(filterBox);  getClient.add(filteringBox);   getClient.setVisible(true);  getClient.setResizable(false);  getClient.setLocationRelativeTo(null);  getClient.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  }   private void filterClient(){  filterBox.setBounds(0, 0, 10, 10);  filterBox.addItem("All");  filterBox.addItem("Nume");  filterBox.addItem("Prenume");  filterBox.addItem("Adresa");  filterBox.addItem("Privilegiu");  filterBox.addItem("Perioada de Inchiriere");  filterBox.addItem("Plata");  filterBox.addItem("Numar de Telefon");  filterBox.addItem("Permis de Conducere");   filterBox.addActionListener(e -> filterClient((String) Objects.*requireNonNull*(filterBox.getSelectedItem())));   getClient("All");  }   private void filterClient(String item){  filteringBox.removeAllItems();  filteringBox.setBounds(10, 0, 10, 10);  filteringBox.addItem("All");   if(item.equals("Nume")){  for(Client client : listaClienti){  filteringBox.addItem(client.getNume());  }  }   if(item.equals("Prenume")){  for(Client client : listaClienti){  filteringBox.addItem(client.getPrenume());  }  }   if(item.equals("Adresa")){  for(Client client : listaClienti){  filteringBox.addItem(client.getAdresa());  }  }   if(item.equals("Privilegiu")){  for(Client client : listaClienti){  filteringBox.addItem(client.getPrivilegiu());  }  }   if(item.equals("Perioada de Inchiriere")){  for(Client client : listaClienti){  filteringBox.addItem(client.getPerioadaInchirierii());  }  }   if(item.equals("Plata")){  for(Client client : listaClienti){  filteringBox.addItem(client.getPlata());  }  }   if(item.equals("Numar de Telefon")){  for(Client client : listaClienti){  filteringBox.addItem(String.*valueOf*(client.getTelefon()));  }  }   if(item.equals("Permis de Conducere")){  for(Client client : listaClienti){  filteringBox.addItem(String.*valueOf*(client.getPermisConducere()));  }  }   filteringBox.addActionListener(e -> getClient((String) filteringBox.getSelectedItem()));   getClient("All");  }   private void getRepair(String item) {  panel.removeAll();  panel.setLayout(null);  repairFrame.dispose();   getRepair.setTitle("Afisarea Informatii Despre Companii de Reparare");  getRepair.setSize(300, 375);   getRepair.setLayout(new FlowLayout());   textArea = new JTextArea(15, 23);  textArea.setEditable(false);   scrollPane = new JScrollPane(textArea);  scrollPane.setVerticalScrollBarPolicy(JScrollPane.*VERTICAL\_SCROLLBAR\_ALWAYS*);  scrollPane.setHorizontalScrollBarPolicy(JScrollPane.*HORIZONTAL\_SCROLLBAR\_AS\_NEEDED*);   panel.setLayout(new BorderLayout());  panel.add(scrollPane, BorderLayout.*CENTER*);   for (CompanieReparati companieReparati : listaReparatii) {  if (item == null || item.equals("All") || item.equals(companieReparati.getDenumire()) ||  item.equals(companieReparati.getPlataServiciilor())) {  textArea.append("Denumire: " + companieReparati.getDenumire() +  "\nPlata Serviciilor: " + companieReparati.getPlataServiciilor() + "\n");  }   }   createOtherBtn(getRepair, getRepair.getWidth(), getRepair.getHeight());  next.addActionListener(e -> getRepairCar("All"));  back.addActionListener(e -> repairMenu(getRepair));  getRepair.add(panel);  getRepair.add(filterBox);  getRepair.add(filteringBox);   getRepair.setVisible(true);  getRepair.setResizable(false);  getRepair.setLocationRelativeTo(null);  getRepair.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  }   private void filterRepair(){  filterBox.setBounds(0, 0, 10, 10);  filterBox.addItem("All");  filterBox.addItem("Denumire");  filterBox.addItem("Plata Serviciilor");   filterBox.addActionListener(e -> filterRepair((String) Objects.*requireNonNull*(filterBox.getSelectedItem())));   getRepair("All");  }   private void filterRepair(String item){  filteringBox.removeAllItems();  filteringBox.setBounds(10, 0, 10, 10);  filteringBox.addItem("All");   if(item.equals("Denumire")){  for(CompanieReparati companieReparati : listaReparatii){  filteringBox.addItem(companieReparati.getDenumire());  }  }   if(item.equals("Plata Serviciilor")){  for(CompanieReparati companieReparati : listaReparatii){  filteringBox.addItem(companieReparati.getPlataServiciilor());  }  }    filteringBox.addActionListener(e -> getRepair((String) filteringBox.getSelectedItem()));   getRepair("All");  }   private void getCity(String item) {  panel.removeAll();  panel.setLayout(null);  cityFrame  .dispose();   getCity.setTitle("Afisarea Informatii Despre Filiale");  getCity.setSize(300, 375);   getCity.setLayout(new FlowLayout());   textArea = new JTextArea(15, 23);  textArea.setEditable(false);   scrollPane = new JScrollPane(textArea);  scrollPane.setVerticalScrollBarPolicy(JScrollPane.*VERTICAL\_SCROLLBAR\_ALWAYS*);  scrollPane.setHorizontalScrollBarPolicy(JScrollPane.*HORIZONTAL\_SCROLLBAR\_AS\_NEEDED*);   panel.setLayout(new BorderLayout());  panel.add(scrollPane, BorderLayout.*CENTER*);   for (Filiala filiala : listaFiliale) {  if (item == null || item.equals("All") || item.equals(filiala.getDenumire()) ||  item.equals(filiala.getAdresa()) || item.equals(filiala.getNumePersContact()) ||  item.equals(filiala.getPrenumePersContact()) || item.equals(String.*valueOf*(filiala.getTelefon()))) {  textArea.append("Denumire: " + filiala.getDenumire() +  "\nAdreasa: " + filiala.getAdresa() +  "\nNumele Persoanei de Contact" + filiala.getNumePersContact() +  "\nPrenumele Persoanei de Contact: " + filiala.getPrenumePersContact() +  "\nTelefon" + filiala.getTelefon() + "\n");  }   }   createOtherBtn(getCity, getCity.getWidth(), getCity.getHeight());  next.addActionListener(e -> setCity());  back.addActionListener(e -> citiesMenu(getCity));  getCity.add(panel);  getCity.add(filterBox);  getCity.add(filteringBox);   getCity.setVisible(true);  getCity.setResizable(false);  getCity.setLocationRelativeTo(null);  getCity.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  }   private void filterCity(){  filterBox.setBounds(0, 0, 10, 10);  filterBox.addItem("All");  filterBox.addItem("Denumire");  filterBox.addItem("Adreasa");  filterBox.addItem("Numele Persoanei de Contact");  filterBox.addItem("Prenumele Persoanei de Contact");  filterBox.addItem("Telefon");   filterBox.addActionListener(e -> filterCity((String) Objects.*requireNonNull*(filterBox.getSelectedItem())));   getCity("All");  }   private void filterCity(String item){  filteringBox.removeAllItems();  filteringBox.setBounds(10, 0, 10, 10);  filteringBox.addItem("All");   if(item.equals("Denumire")){  for(Filiala filiala : listaFiliale){  filteringBox.addItem(filiala.getDenumire());  }  }   if(item.equals("Adreasa")){  for(Filiala filiala : listaFiliale){  filteringBox.addItem(filiala.getAdresa());  }  }  if(item.equals("Numele Persoanei de Contact")){  for(Filiala filiala : listaFiliale){  filteringBox.addItem(filiala.getNumePersContact());  }  }   if(item.equals("Prenumele Persoanei de Contact")){  for(Filiala filiala : listaFiliale){  filteringBox.addItem(filiala.getPrenumePersContact());  }  }   if(item.equals("Telefon")){  for(Filiala filiala : listaFiliale){  filteringBox.addItem(String.*valueOf*(filiala.getTelefon()));  }  }    filteringBox.addActionListener(e -> getCity((String) filteringBox.getSelectedItem()));   getCity("All");  }   private void getClientCar(String item) {  panel.removeAll();  panel.setLayout(null);  clientFrame.dispose();   getClientCar.setTitle("Afisarea Informatii Despre Client");  getClientCar.setSize(300, 375);   getClientCar.setLayout(new FlowLayout());   textArea = new JTextArea(15, 23);  textArea.setEditable(false);   scrollPane = new JScrollPane(textArea);  scrollPane.setVerticalScrollBarPolicy(JScrollPane.*VERTICAL\_SCROLLBAR\_ALWAYS*);  scrollPane.setHorizontalScrollBarPolicy(JScrollPane.*HORIZONTAL\_SCROLLBAR\_AS\_NEEDED*);   panel.setLayout(new BorderLayout());  panel.add(scrollPane, BorderLayout.*CENTER*);   for (int index = 0; index < listaClientMasini.size(); index++) {  if(item == null || item.equals("All") || item.equals(listaClientMasini.get(index).getNume()) ||  item.equals(listaClientMasini.get(index).getPrenume()) || item.equals(listaClientMasini.get(index).getAchizitie())  || item.equals(String.*valueOf*(listaClientMasini.get(index).getCodMasina())) ||  item.equals(String.*valueOf*(listaClientMasini.get(index).getPerioada()))){   textArea.append("Nume: " + listaClientMasini.get(index).getNume() +  "\nPrenume: " + listaClientMasini.get(index).getPrenume() +  "\nAchizitie: " + listaClientMasini.get(index).getAchizitie() +  "\nNumar de Inmatriculare: " + listaClientMasini.get(index).getCodMasina() +  "\nPerioada de Inchirier: " + listaClientMasini.get(index).getPerioada() + "\n");  }   }   createOtherBtn(getClientCar, getClientCar.getWidth(), getClientCar.getHeight());  next.addActionListener(e -> clientSumPaid());  back.addActionListener(e -> clientelMenu(getClientCar));  getClientCar.add(panel);  getClientCar.add(filterBox);  getClientCar.add(filteringBox);   getClientCar.setVisible(true);  getClientCar.setResizable(false);  getClientCar.setLocationRelativeTo(null);  getClientCar.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  }   private void filterClientCar(){  filterBox.setBounds(0, 0, 10, 10);  filterBox.addItem("All");  filterBox.addItem("Nume");  filterBox.addItem("Prenume");  filterBox.addItem("Achizitie");  filterBox.addItem("Numar de Inmatriculare");  filterBox.addItem("Perioada de Inchiriere");   filterBox.addActionListener(e -> filterClientCar((String) Objects.*requireNonNull*(filterBox.getSelectedItem())));   getClientCar("All");  }   private void filterClientCar(String item){  filteringBox.removeAllItems();  filteringBox.setBounds(10, 0, 10, 10);  filteringBox.addItem("All");   if(item.equals("Nume")){  for(ClientMasina clientMasina : listaClientMasini){  filteringBox.addItem(clientMasina.getNume());  }  }   if(item.equals("Prenume")){  for(ClientMasina clientMasina : listaClientMasini){  filteringBox.addItem(clientMasina.getPrenume());  }  }   if(item.equals("Achizitie")){  for(ClientMasina clientMasina : listaClientMasini){  filteringBox.addItem(clientMasina.getAchizitie());  }  }   if(item.equals("Numar de Inmatriculare")){  for(ClientMasina clientMasina : listaClientMasini){  filteringBox.addItem(String.*valueOf*(clientMasina.getCodMasina()));  }  }   if(item.equals("Perioada de Inchiriere")){  for(ClientMasina clientMasina : listaClientMasini){  filteringBox.addItem(String.*valueOf*(clientMasina.getPerioada()));  }  }   filteringBox.addActionListener(e -> getClientCar((String) filteringBox.getSelectedItem()));   getClientCar("All");  }   private void getRepairCar(String item) {  panel.removeAll();  panel.setLayout(null);  repairFrame.dispose();   getRepairCar.setTitle("Afisarea Informatii Despre Companii de Reparare");  getRepairCar.setSize(300, 375);   getRepairCar.setLayout(new FlowLayout());   textArea = new JTextArea(15, 23);  textArea.setEditable(false);   scrollPane = new JScrollPane(textArea);  scrollPane.setVerticalScrollBarPolicy(JScrollPane.*VERTICAL\_SCROLLBAR\_ALWAYS*);  scrollPane.setHorizontalScrollBarPolicy(JScrollPane.*HORIZONTAL\_SCROLLBAR\_AS\_NEEDED*);  panel.setLayout(new BorderLayout());  panel.add(scrollPane, BorderLayout.*CENTER*);   for (int index = 0; index < listaCompRepMasini.size(); index++) {  if(item == null || item.equals("All") || item.equals(listaCompRepMasini.get(index).getDenumire()) ||  item.equals(String.*valueOf*(listaCompRepMasini.get(index).getCodMasina())) ||  item.equals(String.*valueOf*(listaCompRepMasini.get(index).getPerioada()))){  textArea.append("Denumire: " + listaCompRepMasini.get(index).getDenumire() +  "\nNumar de Inmatriculare: " + listaCompRepMasini.get(index).getCodMasina() +  "\nPerioada de Reparare:" + listaCompRepMasini.get(index).getPerioada() + "\n");  }   }   createOtherBtn(getRepairCar, getRepairCar.getWidth(), getRepairCar.getHeight());  next.addActionListener(e -> numberCars());  back.addActionListener(e -> repairMenu(getRepairCar));  getRepairCar.add(panel);  getRepairCar.add(filterBox);  getRepairCar.add(filteringBox);   getRepairCar.setVisible(true);  getRepairCar.setResizable(false);  getRepairCar.setLocationRelativeTo(null);  getRepairCar.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  }   private void filterRepairCar(){  filterBox.setBounds(0, 0, 10, 10);  filterBox.addItem("All");  filterBox.addItem("Denumire");  filterBox.addItem("Numar de Inmatriculare");  filterBox.addItem("Perioada de Reparare");   filterBox.addActionListener(e -> filterRepairCar((String) Objects.*requireNonNull*(filterBox.getSelectedItem())));   getRepairCar("All");  }   private void filterRepairCar(String item){  filteringBox.removeAllItems();  filteringBox.setBounds(10, 0, 10, 10);  filteringBox.addItem("All");   if(item.equals("Denumire")){  for(ReparareMasina reparareMasina : listaCompRepMasini){  filteringBox.addItem(reparareMasina.getDenumire());  }  }   if(item.equals("Numar de Inmatriculare")){  for(ReparareMasina reparareMasina : listaCompRepMasini){  filteringBox.addItem(String.*valueOf*(reparareMasina.getCodMasina()));  }  }   if(item.equals("Perioada de Reparare")){  for(ReparareMasina reparareMasina : listaCompRepMasini){  filteringBox.addItem(String.*valueOf*(reparareMasina.getPerioada()));  }  }    filteringBox.addActionListener(e -> getRepairCar((String) filteringBox.getSelectedItem()));   getRepairCar("All");  }   private void verifyCar() {  verifAuto();   panel.removeAll();  panel.setLayout(null);  carFrame.dispose();   verifyCar.setTitle("Verificarea Masiniilor");  verifyCar.setSize(300, 175);   verifyCar.setLayout(new FlowLayout());   textArea = new JTextArea(5, 23);  textArea.setEditable(false);   scrollPane = new JScrollPane(textArea);  scrollPane.setVerticalScrollBarPolicy(JScrollPane.*VERTICAL\_SCROLLBAR\_ALWAYS*);  scrollPane.setHorizontalScrollBarPolicy(JScrollPane.*HORIZONTAL\_SCROLLBAR\_AS\_NEEDED*);   panel.setLayout(new BorderLayout());  panel.add(scrollPane, BorderLayout.*CENTER*);   textArea.append("Vandute: " + num[0] +  "\nInchiriate: " + num[1] +  "\nReparare: " + num[2] +  "\nDepozitate: " + num[3] +  "\nParcate: " + num[4]);   createOtherBtn(verifyCar, verifyCar.getWidth(), verifyCar.getHeight());  next.addActionListener(e -> filterBuyLicense());  back.addActionListener(e -> carMenu(verifyCar));  verifyCar.add(panel);   verifyCar.setVisible(true);  verifyCar.setResizable(false);  verifyCar.setLocationRelativeTo(null);  verifyCar.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  }   private void convertBuy(String item) {  panel.removeAll();  panel.setLayout(new GridLayout(1, 3));  carFrame.dispose();   convertBuy.setTitle("Convertirea Pretului de Cumparare");  convertBuy.setSize(300, 135);  convertBuy.setLayout(new FlowLayout());   buyPriceText.setBounds(0, 50, 100, 25);  buyPriceText.setEditable(false);  if(!item.equals(" ")) {  for (Masina car : listaMasini) {  if (car.getNumarInmatriculare() == Integer.*parseInt*(item)) {  buyPriceText.setText(car.getPretCumparare() + " " + car.getPrCumpValuta());  convert.addActionListener(e -> convert(*car*.getPretCumparare(), *car*.getPrCumpValuta(), (String) buyCurrency.getSelectedItem()));  }  }  }   converted.setEditable(false);   panel.add(buyPriceText);  panel.add(convert);  panel.add(converted);   createOtherBtn(convertBuy, convertBuy.getWidth(), convertBuy.getHeight());  next.addActionListener(e -> filterRentLicense());  back.addActionListener(e-> {  buyPriceText.setText("");  converted.setText("");  carMenu(convertBuy);});  convertBuy.add(filterBox);  convertBuy.add(buyCurrency);  convertBuy.add(panel);   convertBuy.setVisible(true);  convertBuy.setResizable(false);  convertBuy.setLocationRelativeTo(null);  convertBuy.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  }   private void filterBuyLicense(){  filterBox.removeAllItems();   filterBox.setBounds(50, 35, 75, 25);  filterBox.addItem(" ");  for(Masina car : listaMasini){  filterBox.addItem(String.*valueOf*(car.getNumarInmatriculare()));  }   filterBox.addActionListener(e -> {  Object selectedItem = filterBox.getSelectedItem();  if (selectedItem != null) {  String selectedValue = selectedItem.toString();  convertBuy.remove(filterBox);  convertBuy.remove(buyCurrency);  convertBuy(selectedValue);  }  });  convertBuy(" ");  }   private void convert(Double amount, String currency, String convertCurrency){  switch(currency){  case "MDL" -> {  switch (convertCurrency) {  case "MDL" -> converted.setText(amount + " MDL");  case "EUR" -> converted.setText(amount \* 19.0 + " EUR");  case "USD" -> converted.setText(amount \* 17.8 + " USD");  }  }  case "EUR" -> {  switch (convertCurrency){   case "MDL" -> converted.setText(amount \* 0.053 + " MDL");  case "EUR" -> converted.setText(amount + " EUR");  case "USD" -> converted.setText(amount \* 1.07 + " USD");  }  }  case "USD" -> {  switch (convertCurrency){   case "MDL" -> converted.setText(amount \* 0.056 + " MDL");  case "EUR" -> converted.setText(amount \* 0.94 + " EUR");  case "USD" -> converted.setText(amount + " USD");  }  }  }  }   private void convertRent(String item) {  panel.removeAll();  panel.setLayout(new GridLayout(1, 3));  carFrame.dispose();   convertRent.setTitle("Convertirea Pretului de Inchiriere");  convertRent.setSize(300, 135);  convertRent.setLayout(new FlowLayout());   buyPriceText.setBounds(0, 50, 100, 25);  buyPriceText.setEditable(false);  if(!item.equals(" ")) {  for (Masina car : listaMasini) {  if (car.getNumarInmatriculare() == Integer.*parseInt*(item)) {  buyPriceText.setText(car.getPretInchiriere() + " " + car.getPrInchValuta());  convert.addActionListener(e -> convert(*car*.getPretInchiriere(), *car*.getPrInchValuta(), (String) buyCurrency.getSelectedItem()));  }  }  }   converted.setEditable(false);   panel.add(buyPriceText);  panel.add(convert);  panel.add(converted);   createOtherBtn(convertRent, convertRent.getWidth(), convertRent.getHeight());  next.addActionListener(e -> setCar());  back.addActionListener(e ->{  buyPriceText.setText("");  converted.setText("");  carMenu(convertRent);});  convertRent.add(filterBox);  convertRent.add(buyCurrency);  convertRent.add(panel);   convertRent.setVisible(true);  convertRent.setResizable(false);  convertRent.setLocationRelativeTo(null);  convertRent.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  }   private void filterRentLicense(){  filterBox.removeAllItems();   filterBox.setBounds(50, 35, 75, 25);  filterBox.addItem(" ");  for(Masina car : listaMasini){  filterBox.addItem(String.*valueOf*(car.getNumarInmatriculare()));  }   filterBox.addActionListener(e -> {  Object selectedItem = filterBox.getSelectedItem();  if (selectedItem != null) {  String selectedValue = selectedItem.toString();  convertBuy.remove(filterBox);  convertRent.remove(buyCurrency);  convertRent(selectedValue);  }  });  convertRent(" ");  }  private void clientSumPaid(){  sumaAchitata();   panel.removeAll();  panel.setLayout(null);  clientFrame.dispose();   sumClientPaid.setTitle("Sumele Achitate de Clienti");  sumClientPaid.setSize(300, 175);   sumClientPaid.setLayout(new FlowLayout());   textArea = new JTextArea(5, 23);  textArea.setEditable(false);   scrollPane = new JScrollPane(textArea);  scrollPane.setVerticalScrollBarPolicy(JScrollPane.*VERTICAL\_SCROLLBAR\_ALWAYS*);  scrollPane.setHorizontalScrollBarPolicy(JScrollPane.*HORIZONTAL\_SCROLLBAR\_AS\_NEEDED*);   panel.setLayout(new BorderLayout());  panel.add(scrollPane, BorderLayout.*CENTER*);   for(int index = 0; index < achitare.size(); index++){  if(index >= listaClientMasini.size()) break;  textArea.append(listaClientMasini.get(index).getNume() +" "+ listaClientMasini.get(index).getPrenume() +": "+ achitare.get(checkClient(listaClientMasini.get(index).getNume())) + "\n");  }   createOtherBtn(sumClientPaid, sumClientPaid.getWidth(), sumClientPaid.getHeight());  next.addActionListener(e -> setClient());  back.addActionListener(e -> clientelMenu(sumClientPaid));  sumClientPaid.add(panel);   sumClientPaid.setVisible(true);  sumClientPaid.setResizable(false);  sumClientPaid.setLocationRelativeTo(null);  sumClientPaid.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  }   private void numberCars() {  autoRepair();   panel.removeAll();  panel.setLayout(null);  repairFrame.dispose();   numberCars.setTitle("Numarul de Masini la Companiile de Reparetie");  numberCars.setSize(300, 175);   numberCars.setLayout(new FlowLayout());   textArea = new JTextArea(5, 23);  textArea.setEditable(false);   scrollPane = new JScrollPane(textArea);  scrollPane.setVerticalScrollBarPolicy(JScrollPane.*VERTICAL\_SCROLLBAR\_ALWAYS*);  scrollPane.setHorizontalScrollBarPolicy(JScrollPane.*HORIZONTAL\_SCROLLBAR\_AS\_NEEDED*);   panel.setLayout(new BorderLayout());  panel.add(scrollPane, BorderLayout.*CENTER*);   for(int index = 0; index < listaCompRepMasini.size(); index++){  if(count[checkRepair(listaCompRepMasini.get(index).getDenumire())] == 0) break;  textArea.append(listaReparatii.get(checkRepair(listaCompRepMasini.get(index).getDenumire())).getDenumire() +": "+ count[checkRepair(listaCompRepMasini.get(index).getDenumire())] + "\n");  }   createOtherBtn(numberCars, numberCars.getWidth(), numberCars.getHeight());  next.addActionListener(e -> setRepair());  back.addActionListener(e -> repairMenu(numberCars));   numberCars.add(panel);   numberCars.setVisible(true);  numberCars.setResizable(false);  numberCars.setLocationRelativeTo(null);  numberCars.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  } } |
| // Masina.java  import java.io.\*; import java.util.\*; public class Masina {  private String producator, model, combustibil, dataCumparare, situatie, prCumpValuta, prInchValuta, filiala, compRepar;  private int anulFabricat, numarPasageri, numarInmatriculare;  private double marimeaMotorului, pretCumparare, pretInchiriere;   public Masina(){}   public Masina(String producator, String model, String combustibil, String dataCumparare, String situatie, int anulFabricat, int numarPasageri, int numarInmatriculare, double marimeaMotorului, double pretCumparare, String prCumpValuta, double pretInchiriere, String prInchValuta) {  this.producator = producator;  this.model = model;  this.combustibil = combustibil;  this.dataCumparare = dataCumparare;  this.situatie = situatie;  this.anulFabricat = anulFabricat;  this.numarPasageri = numarPasageri;  this.numarInmatriculare = numarInmatriculare;  this.marimeaMotorului = marimeaMotorului;  this.pretCumparare = pretCumparare;  this.prCumpValuta = prCumpValuta;  this.pretInchiriere = pretInchiriere;  this.prInchValuta = prInchValuta;  }   public void menu(int index){  switch(index){  case 4 -> convertBuyCurrency();  case 5 -> convertRentCurrency();  }  }   public void setAuto(File file){  Scanner scan = new Scanner(System.*in*);  this.producator = scan.nextLine();  this.model = scan.nextLine();  this.combustibil = scan.nextLine();  this.dataCumparare = scan.nextLine();  this.situatie = scan.nextLine();  this.anulFabricat = scan.nextInt();  this.numarPasageri = scan.nextInt();  this.numarInmatriculare = scan.nextInt();  this.marimeaMotorului = scan.nextDouble();  this.pretCumparare = scan.nextDouble();  this.prCumpValuta = scan.nextLine();  this.pretInchiriere = scan.nextDouble();  this.prInchValuta = scan.nextLine();  try{  FileWriter fileWriter = new FileWriter(file, true);  BufferedWriter bufferedWriter = new BufferedWriter(fileWriter);  bufferedWriter.write("m "+this.producator+" "+this.model+" "+this.combustibil+" "+this.dataCumparare  +" "+this.situatie+" "+this.anulFabricat+" "+this.numarPasageri+" "+this.numarInmatriculare  +" "+this.marimeaMotorului+" "+this.pretCumparare +" "+this.prCumpValuta+" "+this.pretInchiriere  +" "+this.prInchValuta+"\n");  bufferedWriter.close();  }catch(IOException e){  e.printStackTrace();  }  }   public void convertBuyCurrency(){  Scanner scan = new Scanner(System.*in*);  System.*out*.printf(this.pretCumparare+" "+this.prCumpValuta+"\nIn ce doriti ca convertiti: ");  String convCurrency = scan.nextLine();  switch (convCurrency){  case "MDL" -> convertMDL(this.pretCumparare, this.prCumpValuta);  case "EUR" -> convertEUR(this.pretCumparare, this.prCumpValuta);  case "USD" -> convertUSD(this.pretCumparare, this.prCumpValuta);  }  }   public void convertRentCurrency(){  Scanner scan = new Scanner(System.*in*);  System.*out*.printf(this.pretInchiriere+" "+this.prInchValuta+"\nIn ce doriti ca convertiti: ");  String convCurrency = scan.nextLine();  switch (convCurrency){  case "MDL" -> convertMDL(this.pretInchiriere, this.prInchValuta);  case "EUR" -> convertEUR(this.pretInchiriere, this.prInchValuta);  case "USD" -> convertUSD(this.pretInchiriere, this.prInchValuta);  }  }   public void convertMDL(double sum, String currency){  switch (currency){  case "EUR" -> {  sum \*= 19.0;  System.*out*.println(sum + " MDL");  }  case "USD" -> {  sum \*= 17.8;  System.*out*.println(sum + " MDL");  }  }  }   public void convertEUR(double sum, String currency){  switch (currency){  case "MDL" -> {  sum \*= 19.00;  System.*out*.println(sum + " EUR");  }  case "USD" -> {  sum \*= 1.07;  System.*out*.println(sum + " EUR");  }  }  }   public void convertUSD(double sum, String currency){  switch (currency){  case "MDL" -> {  sum \*= 17.80;  System.*out*.println(sum + " USD");  }  case "EUR" -> {  sum \*= 0.94;  System.*out*.println(sum + " USD");  }  }  }   public String getSituatie() {  return situatie;  }   public void setSituatie(String situatie) {  this.situatie = situatie;  }   public String getProducator() {  return producator;  }   public void setProducator(String producator) {  this.producator = producator;  }   public String getModel() {  return model;  }   public void setModel(String model) {  this.model = model;  }   public String getCombustibil() {  return combustibil;  }   public void setCombustibil(String combustibil) {  this.combustibil = combustibil;  }   public String getdataCumparare() {  return dataCumparare;  }   public void setdataCumparare(String dataCumparare) {  this.dataCumparare = dataCumparare;  }   public int getAnulFabricat() {  return anulFabricat;  }   public void setAnulFabricat(int anulFabricat) {  this.anulFabricat = anulFabricat;  }   public int getNumarPasageri() {  return numarPasageri;  }   public void setNumarPasageri(int numarPasageri) {  this.numarPasageri = numarPasageri;  }   public int getNumarInmatriculare() {  return numarInmatriculare;  }   public void setNumarInmatriculare(int numarInmatriculare) {  this.numarInmatriculare = numarInmatriculare;  }   public double getMarimeaMotorului() {  return marimeaMotorului;  }   public void setMarimeaMotorului(double marimeaMotorului) {  this.marimeaMotorului = marimeaMotorului;  }   public double getPretCumparare() {  return pretCumparare;  }   public void setPretCumparare(double pretCumparare) {  this.pretCumparare = pretCumparare;  }   public double getPretInchiriere() {  return pretInchiriere;  }   public void setPretInchiriere(double pretInchiriere) {  this.pretInchiriere = pretInchiriere;  }   public String getPrCumpValuta() {  return prCumpValuta;  }   public void setPrCumpValuta(String prCumpValuta) {  this.prCumpValuta = prCumpValuta;  }   public String getPrInchValuta() {  return prInchValuta;  }   public void setPrInchValuta(String prInchValuta) {  this.prInchValuta = prInchValuta;  }   @Override  public String toString() {  return "Masina{" +  "producator='" + producator + '\'' +  ", model='" + model + '\'' +  ", combustibil='" + combustibil + '\'' +  ", dataCumparare='" + dataCumparare + '\'' +  ", situatie='" + situatie + '\'' +  ", prCumpValuta='" + prCumpValuta + '\'' +  ", prInchValuta='" + prInchValuta + '\'' +  ", anulFabricat=" + anulFabricat + '\'' +  ", numarPasageri=" + numarPasageri + '\'' +  ", numarInmatriculare=" + numarInmatriculare + '\'' +  ", marimeaMotorului=" + marimeaMotorului + '\'' +  ", pretCumparare=" + pretCumparare + '\'' +  ", pretInchiriere=" + pretInchiriere + '\'' +  '}';  } } |
| // Client.java  import java.io.\*; import java.util.Scanner;  public class Client {  private String nume, prenume, adresa, privilegiu, perioadaInchirierii, plata;  private int telefon, permisConducere;   public Client(){}   public Client(String nume, String prenume, String adresa, int telefon, int permisConducere) {  this.nume = nume;  this.prenume = prenume;  this.adresa = adresa;  this.privilegiu = "fara";  this.perioadaInchirierii = "0";  this.plata = "cash";  this.telefon = telefon;  this.permisConducere = permisConducere;  }   public Client(String nume, String prenume, String adresa, String privilegiu, String perioadaInchirierii, String plata, int telefon, int permisConducere) {  this.nume = nume;  this.prenume = prenume;  this.adresa = adresa;  this.privilegiu = privilegiu;  this.perioadaInchirierii = perioadaInchirierii;  this.plata = plata;  this.telefon = telefon;  this.permisConducere = permisConducere;  }   public void setClient(File file){  Scanner scan = new Scanner(System.*in*);  this.nume = scan.nextLine();  this.prenume = scan.nextLine();  this.adresa = scan.nextLine();  this.privilegiu = scan.nextLine();  this.perioadaInchirierii = scan.nextLine();  this.plata = scan.nextLine();  this.telefon = scan.nextInt();  this.permisConducere = scan.nextInt();  try{  FileWriter fileWriter = new FileWriter(file, true);  BufferedWriter bufferedWriter = new BufferedWriter(fileWriter);  bufferedWriter.write("c "+this.nume+" "+this.prenume+" "+this.adresa+" "+this.privilegiu  +" "+this.perioadaInchirierii+" "+this.plata+" "+this.telefon+" "+this.permisConducere+"\n");  bufferedWriter.close();  }catch(IOException e){  e.printStackTrace();  }  }   public String getNume() {  return nume;  }   public void setNume(String nume) {  this.nume = nume;  }   public String getPrenume() {  return prenume;  }   public void setPrenume(String prenume) {  this.prenume = prenume;  }   public String getAdresa() {  return adresa;  }   public void setAdresa(String adresa) {  this.adresa = adresa;  }   public String getPrivilegiu() {  return privilegiu;  }   public void setPrivilegiu(String privilegiu) {  this.privilegiu = privilegiu;  }   public String getPlata() {  return plata;  }   public void setPlata(String plata) {  this.plata = plata;  }   public int getTelefon() {  return telefon;  }   public void setTelefon(int telefon) {  this.telefon = telefon;  }   public int getPermisConducere() {  return permisConducere;  }   public void setPermisConducere(int permisConducere) {  this.permisConducere = permisConducere;  }   public String getPerioadaInchirierii() {  return perioadaInchirierii;  }   public void setPerioadaInchirierii(String perioadaInchirierii) {  this.perioadaInchirierii = perioadaInchirierii;  }   @Override  public String toString() {  return "Client{" +  "nume='" + nume + '\'' +  ", prenume='" + prenume + '\'' +  ", adresa='" + adresa + '\'' +  ", privilegiu='" + privilegiu + '\'' +  ", perioadaInchirierii='" + perioadaInchirierii + '\'' +  ", plata='" + plata + '\'' +  ", telefon=" + telefon +  ", permisConducere=" + permisConducere +  '}';  } } |
| //CompanieReparati.java  import java.io.\*; import java.util.Scanner;  public class CompanieReparati {  private String denumire, plataServiciilor;   public CompanieReparati(){}   public CompanieReparati(String denumire) {  this.denumire = denumire;  this.plataServiciilor = "imediata";  }   public CompanieReparati(String denumire, String plataServiciilor) {  this.denumire = denumire;  this.plataServiciilor = plataServiciilor;  }   public void setCompRep(File file){  Scanner scan = new Scanner(System.*in*);  this.denumire = scan.nextLine();  this.plataServiciilor = scan.nextLine();  try{  FileWriter fileWriter = new FileWriter(file, true);  BufferedWriter bufferedWriter = new BufferedWriter(fileWriter);  bufferedWriter.write("cr "+this.denumire+" "+this.plataServiciilor+"\n");  bufferedWriter.close();  }catch(IOException e){  e.printStackTrace();  }  }   public String getDenumire() {  return denumire;  }   public void setDenumire(String denumire) {  this.denumire = denumire;  }   public String getPlataServiciilor() {  return plataServiciilor;  }   public void setPlataServiciilor(String plataServiciilor) {  this.plataServiciilor = plataServiciilor;  }   @Override  public String toString() {  return "CompanieReparati{" +  "denumire='" + denumire + '\'' +  ", plataServiciilor='" + plataServiciilor + '\'' +  '}';  } } |
| // Filiala.java  import java.io.\*; import java.util.Scanner;  public class Filiala {  private String denumire, adresa, numePersContact, prenumePersContact;  private int telefon;   public Filiala(){}   public Filiala(String denumire, String adresa, String numePersContact, String prenumePersContact, int telefon) {  this.denumire = denumire;  this.adresa = adresa;  this.numePersContact = numePersContact;  this.prenumePersContact = prenumePersContact;  this.telefon = telefon;  }   public void setFiliala(File file){  Scanner scan = new Scanner(System.*in*);  this.denumire = scan.nextLine();  this.adresa = scan.nextLine();  this.numePersContact = scan.nextLine();  this.prenumePersContact = scan.nextLine();  this.telefon = scan.nextInt();  try{  FileWriter fileWriter = new FileWriter(file, true);  BufferedWriter bufferedWriter = new BufferedWriter(fileWriter);  bufferedWriter.write("f "+this.denumire+" "+this.adresa+" "+this.numePersContact+" "+this.prenumePersContact+" "+this.telefon+"\n");  bufferedWriter.close();  }catch(IOException e){  e.printStackTrace();  }  }   public String getDenumire() {  return denumire;  }   public void setDenumire(String denumire) {  this.denumire = denumire;  }   public String getAdresa() {  return adresa;  }   public void setAdresa(String adresa) {  this.adresa = adresa;  }   public String getNumePersContact() {  return numePersContact;  }   public void setNumePersContact(String numePersContact) {  this.numePersContact = numePersContact;  }   public String getPrenumePersContact() {  return prenumePersContact;  }   public void setPrenumePersContact(String prenumePersContact) {  this.prenumePersContact = prenumePersContact;  }   public int getTelefon() {  return telefon;  }   public void setTelefon(int telefon) {  this.telefon = telefon;  }   @Override  public String toString() {  return "Filiala{" +  "denumire='" + denumire + '\'' +  ", adresa='" + adresa + '\'' +  ", numePersContact='" + numePersContact + '\'' +  ", prenumePersContact='" + prenumePersContact + '\'' +  ", telefon=" + telefon +  '}';  } } |

## Rezultatul Execuției Programului:

