Практична робота № 3

**Тема**: Програмування на Java.

**Мета**: Навчитись розробляти об'єктно-орієнтовані програми з використанням мови програмування Java

Завдання:

**Завдання 1:**

Написати програму, що обчислює корінь N-ої степені з числа p (N - ціле, p - дійсне) . Програма повинна використовувати цикл while. Вхідні дані поступають з клавіатури. Результат виводиться на екран. Встановити точність 4 цифри. Передбачити обробку помилок.

**Завдання 2:**

Написати програму, що друкує символи. Використовувати цикл while. У рядок виводиться номер, символ, шестнадцатерічний і восьмерічний код. Передбачити обробку помилок і можливість виведення даних у файл (output.txt).

**Завдання 3:** Написати програму перевірки чи є число простим, встановити ширину поля 10 символів. Заповнити замість пропусків символом «#».

**Завдання 4:**

У квадратній матриці розміром 5х5, заповненою випадковими цілими числами з діапазону (-50,+50) в секторах 2 і 4 знайти кількість негативних елементів. Сектор включає діагональні елементи. Дії з визначення кількості негативних елементів реалізувати з використанням функцій (методів) з параметрами. Масив може заповнюватися як уручну, так і автоматично (з файлу або з використанням функції random()). Передбачити обробку помилок.

**Завдання 5:**

На квадратному клітчастому листі паперу (розміру 100x100 кліток) намальовано декілька прямокутників. Кожен прямокутник складається з цілих кліток, різні прямокутники не накладаються один на одного і не стикаються. Заданий масив розміром 100x100, в якому Aij=1, якщо клітка (i,j) належить якому-небудь прямокутнику і Aij=0 інакше. Написати програму, яка рахує число прямокутників.

**Завдання 6:**

Зведення про наявність матеріалів на складі, включають наступні дані: найменування матеріалу (можуть повторюватися), постачальник, одиниці вимірювання, дата придбання, кількість, ціна однієї одиниці, вся вартість (розраховується по формулі). Вибрати матеріали, придбані за попередній місяць у одного і того ж постачальника і злічити всю вартість матеріалів, придбаних у цього постачальника.

**Завдання 7:**

Створити в графічному режимі систему класів у заданій предметній області: Мережа магазинів (людина, покупка, продавець, магазин)

**Хід роботи:**

Завдання 1:

Код програми:

Main.java:

package task1;

import java.io.IOException;

import java.io.BufferedReader;

import java.io.InputStreamReader;

public class Task1 {

static double mabs(double x){ return (x < 0)? -x : x; }

public static void main(String[] args) throws IOException {

BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));

System.out.print("Enter number: ");

double num = Double.parseDouble(reader.readLine());

System.out.print("Enter degree: ");

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

double eps = 0.0001;

double root = num / rootDegree;

double rn = num;

int count = 0;

while(mabs(root - rn) >= eps)

{

rn = num;

for(int i = 1; i < rootDegree; i++){

rn = rn / root;

}

root = 0.5 \* ( rn + root);

count++;

}

System.out.println("Root: " + Double.toString(root));

System.out.println("Number of iteration: " + Integer.toString(count));

}

}

Приклад роботи програми:

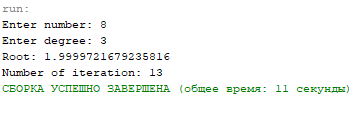


Рисунок 1 – Вивід програми

Завдання 2:

Код програми:

Main.java:

**package task2;**

**import java.io.IOException;**

**import java.io.FileWriter;**

**public class Task2 {**

**public static void main(String[] args) {**

**int firstSymbol = 33;**

**int lastSymbol = 256;**

**try (FileWriter writer = new FileWriter("result.txt", false))**

**{**

**while(firstSymbol < lastSymbol)**

**{**

**char ch = (char) firstSymbol;**

**String oct = Integer.toOctalString(firstSymbol);**

**String hex = Integer.toHexString(firstSymbol);**

**System.out.printf("%03d %3s %5s %4s%n", firstSymbol, ch, oct, hex);**

**writer.write(firstSymbol +"\t"+ ch +"\t"+ oct +"\t"+ hex +"\n");**

**firstSymbol++;**

**}**

**writer.flush();**

**}**

**catch(IOException ex)**

**{**

**System.out.println(ex.getMessage());**

**}**

**}**

**}**

Приклад роботи програми:

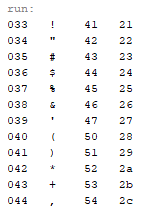


Рисунок 2 – Вивід програми

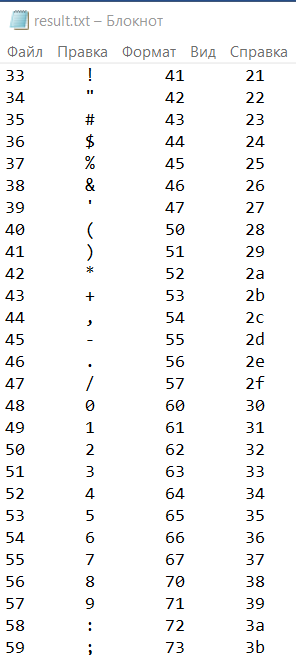


Рисунок 3 – Запис у файл

Завдання 3:

Код програми:

Main.java:

**package task3;**

**import java.io.IOException;**

**import java.io.BufferedReader;**

**import java.io.InputStreamReader;**

**public class Task3 {**

**static boolean IsSimple(int number)**

**{**

**for(int i = 2; i\*i <= number; i++)**

**{**

**if(number % i == 0) return false;**

**}**

**return true;**

**}**

**public static void main(String[] args) throws IOException**

**{**

**BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));**

**System.out.print("Enter number: ");**

**int num = Integer.parseInt(reader.readLine());**

**if(IsSimple(num))**

**{**

**System.out.println("Number " + String.format("%10s", num).replace(' ', '#') + " is simple");**

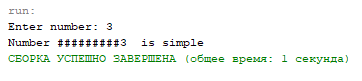
**}**

**else**

**System.out.println("Number is not simple");**

**}**

**}**

Приклад роботи програми:  
  
Рисунок 4 – Вивід програми

Завдання 4:

Код програми:

Main.java:

**package task4;**

**import java.io.BufferedReader;**

**import java.io.InputStreamReader;**

**public class Task4 {**

**static boolean IsValid(int number)**

**{**

**if(number >= -51 && number <= 51)**

**return true;**

**return false;**

**}**

**static String NegativeInSectors(int [][] Array)**

**{**

**int endOfTwoSector = 2;**

**int endOfFourSector = 4;**

**int twoSectorCounter = 0;**

**int fourSectorCounter = 0;**

**for(int i = 0; i < endOfTwoSector; i++)**

**for(int j = 0; j < endOfTwoSector; j++)**

**if(Array[i][j] < 0)**

**twoSectorCounter++;**

**for(int i = 0; i < endOfFourSector; i++)**

**for(int j = 0; j < endOfFourSector; j++)**

**if(Array[i][j] < 0)**

**fourSectorCounter++;**

**String info = "Negative number in two sector: " + Integer.toString(twoSectorCounter) +**

**"\nNegative number in four sector: " + Integer.toString(fourSectorCounter) + "\n";**

**return info;**

**}**

**public static void main(String[] args) throws Exception {**

**BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));**

**final int row = 5;**

**final int column = 5;**

**int [][] Array = new int[5][5];**

**System.out.print("1 - random\n2 - enter with keyboard\nother - exit\n");**

**System.out.print("Enter number: ");**

**int menu = Integer.parseInt(reader.readLine());**

**switch(menu)**

**{**

**case 1:**

**for(int i = 0; i < row; i++)**

**for(int j = 0; j < column; j++)**

**Array[i][j] = (int) Math.round((Math.random() \* 100) - 50);**

**break;**

**case 2:**

**for(int i = 0; i < row; i++)**

**for(int j = 0; j < column; j++)**

**{**

**while(true)**

**{**

**System.out.print("[" + (i+1) + "]" + "[" + (j+1) + "]: ");**

**int number = Integer.parseInt(reader.readLine());**

**if(IsValid(number))**

**{**

**Array[i][j] = number;**

**break;**

**}**

**else**

**{**

**System.out.println("Enter number >= 51 and <= 51");**

**}**

**}**

**}**

**default: break;**

**}**

**for(int i = 0; i < row; i++)**

**{**

**for(int j = 0; j < column; j++)**

**System.out.printf("%5s", Array[i][j]);**

**System.out.println();**

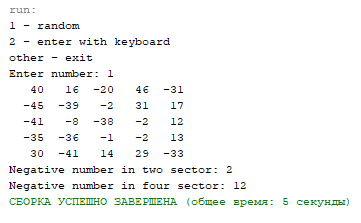
**}**

**System.out.print(NegativeInSectors(Array));**

**}**

**}**

Приклад роботи програми:

  
Рисунок 5 – Вивід програми

Завдання 5:

Код програми:

Main.java

**package task5;**

**import java.io.BufferedReader;**

**import java.io.InputStreamReader;**

**public class Task5 {**

**public static void main(String[] args) throws Exception**

**{**

**BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));**

**int size = 6;**

**int [][] Array = new int[6][6];**

**int count = 0;**

**for(int i = 0; i < size; i++)**

**for(int j = 0; j < size; j++)**

**{**

**Array[0][j] = 0;**

**Array[size - 1][j] = 0;**

**Array[i][0] = 0;**

**Array[i][size - 1] = 0;**

**}**

**for(int i = 1; i < (size - 1); i++)**

**for(int j = 1; j < (size - 1); j++)**

**{**

**System.out.print("[" + i + "]" + "[" + j + "]: ");**

**Array[i][j] = Integer.parseInt(reader.readLine());**

**}**

**for(int i = 0; i < size; i++)**

**{**

**for(int j = 0; j < size; j++)**

**System.out.printf("%2s", Array[i][j]);**

**System.out.println();**

**}**

**for(int i = 1; i < (size - 1); i++)**

**for(int j = 1; j < (size - 1); j++)**

**if(Array[i][j] == 1 && Array[i][j-1] == 0 && Array[i-1][j] == 0 && Array[i-1][j-1] == 0)**

**{**

**Array[i][j] = 1;**

**count += 1;**

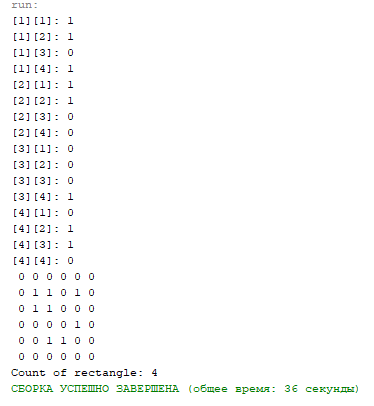
**}**

**System.out.println("Count of rectangle: " + count);**

**}**

**}**

Приклад роботи програми:

  
Рисунок 6 – Вивід програми

Завдання 6:

Код програми:

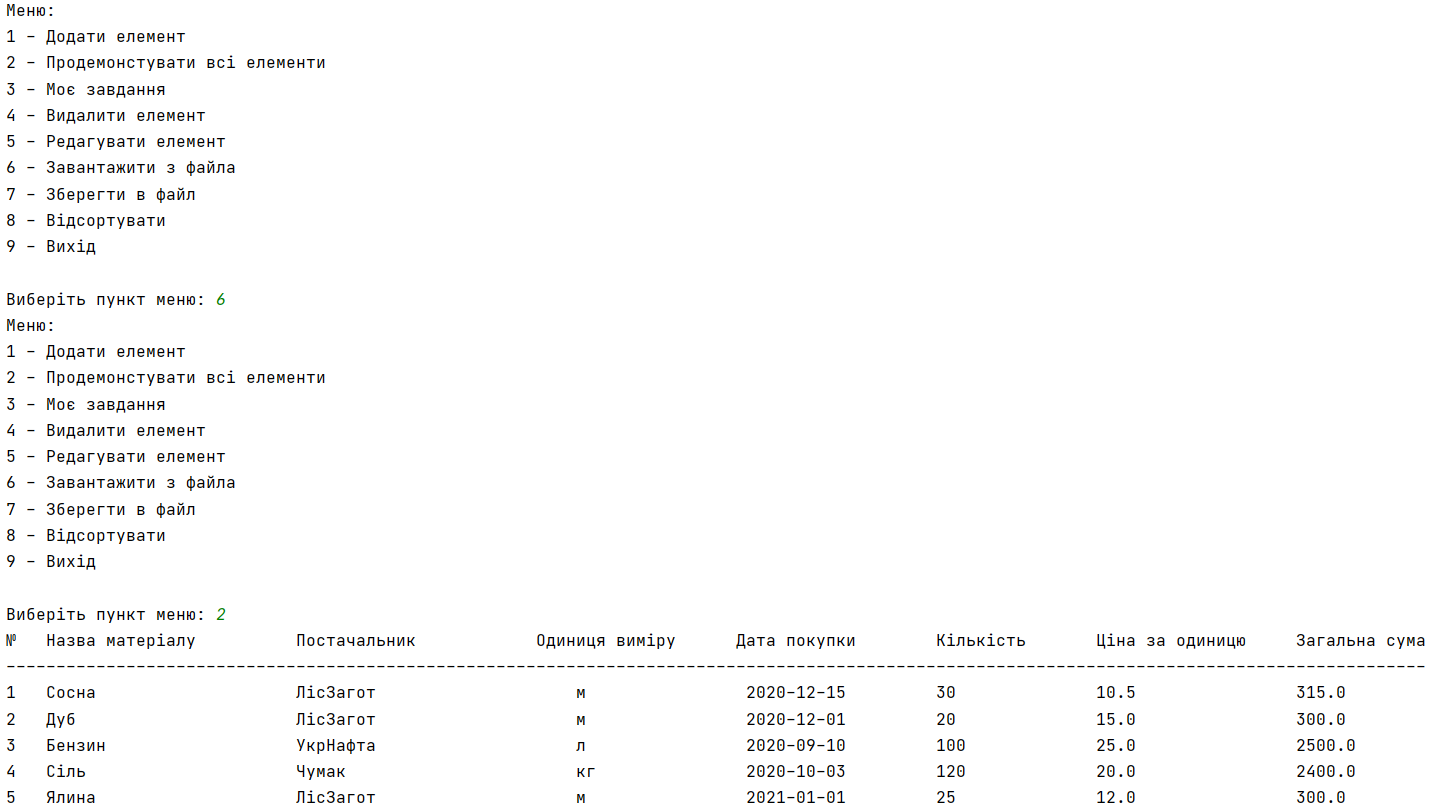
Main.java

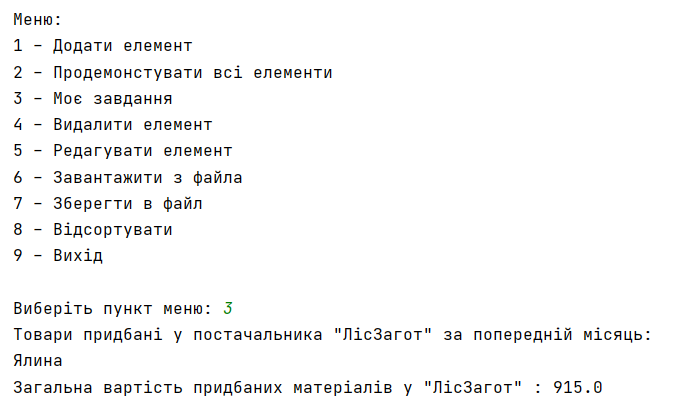
package com.company;  
  
import javax.swing.\*;  
import javax.swing.table.DefaultTableModel;  
import java.awt.\*;  
import java.awt.event.KeyAdapter;  
import java.awt.event.KeyEvent;  
import java.awt.event.KeyListener;  
import java.io.FileInputStream;  
import java.io.FileOutputStream;  
import java.io.ObjectInputStream;  
import java.io.ObjectOutputStream;  
import java.util.ArrayList;  
import java.util.Collections;  
  
import static javax.swing.JOptionPane.*showMessageDialog*;  
  
public class Window extends JFrame {  
  
 JTabbedPane tabbedPane = new JTabbedPane();  
 JPanel panel1 = new JPanel();  
 JPanel panel1Lb = new JPanel();  
 JPanel panel1Btn = new JPanel();  
 JPanel panel2 = new JPanel();  
 JPanel panel2Lb = new JPanel();  
 JPanel panel2Btn = new JPanel();  
 JPanel panel3 = new JPanel();  
 JPanel panel3Lb = new JPanel();  
 JPanel panel3Btn = new JPanel();  
 JPanel panel4 = new JPanel();  
 JPanel panel4Btn = new JPanel();  
 JPanel panel4Tab = new JPanel();  
  
 *//Tab 1 textField* JTextField storeNTF = new JTextField();  
 JTextField storeCityTF = new JTextField();  
 *//Tab 2 textField* JTextField sellerLastNameTF = new JTextField();  
 JTextField sellerFirstNameTF = new JTextField();  
 JTextField sellerExperienceTF = new JTextField();  
 *//Tab 3 textField* JTextField goodsIdTF = new JTextField();  
 JTextField goodsTotalTF = new JTextField();  
  
 *//Tab 1 comboBox* JComboBox sellerComboBoxPanel1 = new JComboBox();  
 *//Tab2 comboBox* JComboBox storeComboBoxPanel2 = new JComboBox();  
 JComboBox sellerComboBoxPanel2 = new JComboBox();  
 *//Tab3 comboBox* JComboBox storeComboBoxPanel3 = new JComboBox();  
 JComboBox sellerComboBoxPanel3 = new JComboBox();  
 JComboBox purchaseComboBoxPanel3 = new JComboBox();  
  
 *//Tab 1 buttons* JButton sellerAddButton = new JButton("Додати");  
 JButton sellerDelButton = new JButton("Видалити");  
 JButton sellerEditButton = new JButton("Редагувати");  
 JButton sellerLoadButton = new JButton("Завантажити з файла");  
 JButton sellerSaveButton = new JButton("Зберегти у файл");  
 *//Tab 2 buttons* JButton storeAddButton = new JButton("Додати");  
 JButton storeDelButton = new JButton("Видалити");  
 JButton storeEditButton = new JButton("Редагувати");  
 JButton storeLoadButton = new JButton("Завантажити з файла");  
 JButton storeSaveButton = new JButton("Зберегти у файл");  
 JButton addSellerButton = new JButton("Додати продавця");  
 *//Tab 3 buttons* JButton purchaseAddButton = new JButton("Додати");  
 JButton purchaseDelButton = new JButton("Видалити");  
 JButton purchaseEditButton = new JButton("Редагувати");  
 JButton purchaseLoadButton = new JButton("Завантажити з файла");  
 JButton purchaseSaveButton = new JButton("Зберегти у файл");  
 *//Tab 4 buttons* JButton sortButton = new JButton("Сортувати");  
  
 ArrayList<Store> chainStores = new ArrayList<>();  
 ArrayList<Seller> sellers = new ArrayList<>();  
 ArrayList<Purchase> purchases = new ArrayList<>();  
  
 String[] col = {"№ магазину", "Місто", "Прізвище продавця", "Ім'я продавця", "Стаж", "ID покупки", "Сума покупки"};  
 DefaultTableModel tableModel = new DefaultTableModel(col, 0);  
  
 JTable table = new JTable(tableModel);  
  
 public Window() {  
 super("Мережа магазинів");  
 this.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
 this.setBounds(400, 100, 850, 400);  
 this.add(tabbedPane);  
  
 tabbedPane.addTab("Продавець", panel1);  
 tabbedPane.addTab("Магазин", panel2);  
 tabbedPane.addTab("Покупка", panel3);  
 tabbedPane.addTab("Таблиця", panel4);  
  
 *//Tab 1 layouts* panel1.setLayout(new GridLayout(2, 1, 1, 5));  
 panel1Lb.setLayout(new GridLayout(4, 2, 1, 5));  
 panel1Btn.setLayout(new GridLayout(2, 3, 1, 5));  
 *//Tab 2 layouts* panel2.setLayout(new GridLayout(2, 1, 1, 5));  
 panel2Lb.setLayout(new GridLayout(4, 2, 1, 5));  
 panel2Btn.setLayout(new GridLayout(2, 3, 1, 5));  
 *//Tab 3 layout* panel3.setLayout(new GridLayout(2, 1, 1, 5));  
 panel3Lb.setLayout(new GridLayout(5, 2, 1, 5));  
 panel3Btn.setLayout(new GridLayout(2, 3, 1, 5));  
 *//Tab 4* panel4.setLayout(new GridLayout(2, 1, 1, 5));  
 panel4Tab.setLayout(new GridLayout(1, 1, 1,5));  
 panel4Btn.setLayout(new GridLayout(3, 1, 1,5));  
  
 *//Tab 1 add* panel1Lb.add(new JLabel("Прізвище продавця", SwingConstants.*CENTER*));  
 panel1Lb.add(sellerLastNameTF);  
 panel1Lb.add(new JLabel("Ім'я продавця", SwingConstants.*CENTER*));  
 panel1Lb.add(sellerFirstNameTF);  
 panel1Lb.add(new JLabel("Стаж продавця", SwingConstants.*CENTER*));  
 panel1Lb.add(sellerExperienceTF);  
 panel1Lb.add(new JLabel("Список продавців", SwingConstants.*CENTER*));  
 panel1Lb.add(sellerComboBoxPanel1);  
 panel1.add(panel1Lb);  
 panel1.add(panel1Btn);  
 panel1Btn.add(sellerAddButton);  
 panel1Btn.add(sellerDelButton);  
 panel1Btn.add(sellerEditButton);  
 panel1Btn.add(sellerSaveButton);  
 panel1Btn.add(sellerLoadButton);  
 *//Tab 2 add* panel2Lb.add(new JLabel("Список продавців", SwingConstants.*CENTER*));  
 panel2Lb.add(sellerComboBoxPanel2);  
 panel2Lb.add(new JLabel("Номер магазину", SwingConstants.*CENTER*));  
 panel2Lb.add(storeNTF);  
 panel2Lb.add(new JLabel("Місто", SwingConstants.*CENTER*));  
 panel2Lb.add(storeCityTF);  
 panel2Lb.add(new JLabel("Список магазинів", SwingConstants.*CENTER*));  
 panel2Lb.add(storeComboBoxPanel2);  
 panel2.add(panel2Lb);  
 panel2.add(panel2Btn);  
 panel2Btn.add(storeAddButton);  
 panel2Btn.add(storeDelButton);  
 panel2Btn.add(storeEditButton);  
 panel2Btn.add(storeSaveButton);  
 panel2Btn.add(storeLoadButton);  
 panel2Btn.add(addSellerButton);  
 *//Tab 3 add* panel3Lb.add(new JLabel("Список магазинів", SwingConstants.*CENTER*));  
 panel3Lb.add(storeComboBoxPanel3);  
 panel3Lb.add(new JLabel("Список продавців", SwingConstants.*CENTER*));  
 panel3Lb.add(sellerComboBoxPanel3);  
 panel3Lb.add(new JLabel("ID покупки", SwingConstants.*CENTER*));  
 panel3Lb.add(goodsIdTF);  
 panel3Lb.add(new JLabel("Загальна сума покупки", SwingConstants.*CENTER*));  
 panel3Lb.add(goodsTotalTF);  
 panel3Lb.add(new JLabel("Список покупок", SwingConstants.*CENTER*));  
 panel3Lb.add(purchaseComboBoxPanel3);  
 panel3.add(panel3Lb);  
 panel3.add(panel3Btn);  
 panel3Btn.add(purchaseAddButton);  
 panel3Btn.add(purchaseDelButton);  
 panel3Btn.add(purchaseEditButton);  
 panel3Btn.add(purchaseSaveButton);  
 panel3Btn.add(purchaseLoadButton);  
  
 storeNTF.addKeyListener(intKeyListener);  
 sellerExperienceTF.addKeyListener(intKeyListener);  
 goodsIdTF.addKeyListener(intKeyListener);  
 goodsTotalTF.addKeyListener(doubleKeyListener);  
  
 storeNTF.setHorizontalAlignment(JTextField.*CENTER*);  
 storeCityTF.setHorizontalAlignment(JTextField.*CENTER*);  
 sellerLastNameTF.setHorizontalAlignment(JTextField.*CENTER*);  
 sellerFirstNameTF.setHorizontalAlignment(JTextField.*CENTER*);  
 sellerExperienceTF.setHorizontalAlignment(JTextField.*CENTER*);  
 goodsIdTF.setHorizontalAlignment(JTextField.*CENTER*);  
 goodsTotalTF.setHorizontalAlignment(JTextField.*CENTER*);  
  
 table.setRowHeight(24);  
 panel4Tab.add(table);  
 panel4Tab.add(new JScrollPane(table));  
 panel4Tab.add(table.getTableHeader());  
 table.setDefaultEditor(Object.class, null);  
 panel4Btn.add(sortButton);  
 panel4.add(panel4Tab);  
 panel4.add(panel4Btn);  
  
  
 *//Seller* sellerAddButton.addActionListener(ae -> {  
 if(!sellerLastNameTF.getText().isEmpty()  
 && !sellerFirstNameTF.getText().isEmpty()  
 && !sellerExperienceTF.getText().isEmpty()) {  
 sellers.add(new Seller(sellerLastNameTF.getText(), sellerFirstNameTF.getText(),  
 Integer.*parseInt*(sellerExperienceTF.getText())));  
  
 sellerLastNameTF.setText("");  
 sellerFirstNameTF.setText("");  
 sellerExperienceTF.setText("");  
 UpdateSellerPanel12();  
 }  
 else  
 *showMessageDialog*(null, "Заповніть всі поля вводу значеннями");  
 });  
 sellerDelButton.addActionListener(ae -> {  
 if(sellerComboBoxPanel1.getSelectedIndex() == -1)  
 *showMessageDialog*(null, "Виберіть елемент для видалення");  
 else {  
 sellers.remove(sellerComboBoxPanel1.getSelectedIndex());  
 UpdateSellerPanel12();  
 }  
 });  
 sellerEditButton.addActionListener(ae -> {  
 if(sellerComboBoxPanel1.getSelectedIndex() == -1)  
 *showMessageDialog*(null, "Виберіть елемент для редагування");  
 else {  
 if (!sellerLastNameTF.getText().isEmpty()  
 && !sellerFirstNameTF.getText().isEmpty()  
 && !sellerExperienceTF.getText().isEmpty()) {  
 sellers.set(sellerComboBoxPanel1.getSelectedIndex(),  
 new Seller(sellerLastNameTF.getText(), sellerFirstNameTF.getText(),  
 Integer.*parseInt*(sellerExperienceTF.getText())));  
  
 sellerLastNameTF.setText("");  
 sellerFirstNameTF.setText("");  
 sellerExperienceTF.setText("");  
 UpdateSellerPanel12();  
 *showMessageDialog*(null, "Запис редаговано");  
 } else  
 *showMessageDialog*(null, "Заповніть всі поля вводу значеннями");  
 }  
 });  
 sellerSaveButton.addActionListener(ae -> {  
 if(sellers.size() == 0)  
 *showMessageDialog*(null, "Немає жодного запису для збереження");  
 else {  
 try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream("seller.dat"))) {  
 oos.writeObject(sellers);  
 } catch (Exception ex) {  
 System.*out*.println(ex.getMessage());  
 }  
 *showMessageDialog*(null, "Записи збережено у файл");  
 }  
 });  
 sellerLoadButton.addActionListener(ae -> {  
 ArrayList<Seller> list = new ArrayList<>();  
  
 try(ObjectInputStream ois = new ObjectInputStream(new FileInputStream("seller.dat")))  
 {  
 list =(ArrayList<Seller>) ois.readObject();  
 }  
 catch(Exception ex){ System.*out*.println(ex.getMessage()); }  
  
 if(list.size() == 0)  
 *showMessageDialog*(null, "Файл пустий або його не існує");  
 else {  
 sellers = list;  
 UpdateSellerPanel12();  
 *showMessageDialog*(null, "Записи з файла успішно завантажено");  
 }  
 });  
 *//Store* storeAddButton.addActionListener(ae -> {  
 if(!storeNTF.getText().isEmpty()  
 && !storeCityTF.getText().isEmpty()  
 && !(sellerComboBoxPanel2.getSelectedIndex() == -1)) {  
 chainStores.add(new Store(Integer.*parseInt*(storeNTF.getText()),  
 storeCityTF.getText(), sellers.get(sellerComboBoxPanel2.getSelectedIndex())));  
  
 sellers.remove(sellerComboBoxPanel2.getSelectedIndex());  
 storeNTF.setText("");  
 storeCityTF.setText("");  
 UpdateSellerPanel12();  
 UpdateStorePanel23();  
 }  
 else  
 *showMessageDialog*(null, "Заповніть всі поля вводу значеннями");  
 });  
 addSellerButton.addActionListener(ae -> {  
 if(storeComboBoxPanel2.getSelectedIndex() == -1 || sellerComboBoxPanel2.getSelectedIndex() == -1)  
 *showMessageDialog*(null, "Виберіть магазин та продавця");  
 else  
 {  
 chainStores.get(storeComboBoxPanel2.getSelectedIndex()).setSellers(sellers.get(sellerComboBoxPanel2.getSelectedIndex()));  
 sellers.remove(sellerComboBoxPanel2.getSelectedIndex());  
 UpdateSellerPanel12();  
 UpdateStorePanel23();  
 }  
 });  
 storeDelButton.addActionListener(ae -> {  
 if(storeComboBoxPanel2.getSelectedIndex() == -1)  
 *showMessageDialog*(null, "Виберіть елемент для видалення");  
 else {  
 for(Seller item : chainStores.get(storeComboBoxPanel2.getSelectedIndex()).getSellers())  
 {  
 sellers.add(item);  
 }  
 chainStores.remove(storeComboBoxPanel2.getSelectedIndex());  
 UpdateStorePanel23();  
 UpdateSellerPanel12();  
 }  
 });  
 storeEditButton.addActionListener(ae -> {  
 if(storeComboBoxPanel2.getSelectedIndex() == -1)  
 *showMessageDialog*(null, "Заповніть всі поля вводу значеннями");  
 else {  
 if (!storeNTF.getText().isEmpty() && !storeCityTF.getText().isEmpty()) {  
 chainStores.get(storeComboBoxPanel2.getSelectedIndex()).EditStore(  
 Integer.*parseInt*(storeNTF.getText()), storeCityTF.getText());  
  
 storeNTF.setText("");  
 storeCityTF.setText("");  
 UpdateStorePanel23();  
 } else  
 *showMessageDialog*(null, "Заповніть всі поля вводу значеннями");  
 }  
 });  
 storeSaveButton.addActionListener(ae -> {  
 if(chainStores.size() == 0)  
 *showMessageDialog*(null, "Немає жодного запису для збереження");  
 else {  
 try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream("store.dat"))) {  
 oos.writeObject(chainStores);  
 } catch (Exception ex) {  
 System.*out*.println(ex.getMessage());  
 }  
 *showMessageDialog*(null, "Записи збережено у файл");  
 }  
 });  
 storeLoadButton.addActionListener(ae -> {  
 ArrayList<Store> list = new ArrayList<>();  
  
 try(ObjectInputStream ois = new ObjectInputStream(new FileInputStream("store.dat")))  
 {  
 list =(ArrayList<Store>) ois.readObject();  
 }  
 catch(Exception ex){ System.*out*.println(ex.getMessage()); }  
  
 if(list.size() == 0)  
 *showMessageDialog*(null, "Файл пустий або його не існує");  
 else {  
 chainStores = list;  
 for (Store item : chainStores)  
 for (Seller seller : item.getSellers())  
 sellers.add(seller);  
  
 UpdatePurchasePanel3();  
 UpdateStorePanel23();  
 UpdateSellerPanel12();  
 *showMessageDialog*(null, "Записи з файла успішно завантажено");  
 }  
 });  
 *//Purchase* purchaseAddButton.addActionListener(ae -> {  
 if(!goodsIdTF.getText().isEmpty()  
 && !goodsTotalTF.getText().isEmpty()  
 && !(storeComboBoxPanel3.getSelectedIndex() == -1)  
 && !(sellerComboBoxPanel3.getSelectedIndex() == -1)) {  
 purchases.add(new Purchase(Integer.*parseInt*(goodsIdTF.getText()), Double.*parseDouble*(goodsTotalTF.getText()),  
 chainStores.get(storeComboBoxPanel3.getSelectedIndex()),  
 chainStores.get(storeComboBoxPanel3.getSelectedIndex()).getSellers().get(sellerComboBoxPanel3.getSelectedIndex())));  
  
 goodsIdTF.setText("");  
 goodsTotalTF.setText("");  
 UpdatePurchasePanel3();  
 UpdateTable();  
  
 }  
 else  
 *showMessageDialog*(null, "Заповніть всі поля вводу значеннями");  
 });  
 purchaseDelButton.addActionListener(ae -> {  
 if(purchaseComboBoxPanel3.getSelectedIndex() == -1)  
 *showMessageDialog*(null, "Виберіть елемент для видалення");  
 else {  
 purchases.remove(purchaseComboBoxPanel3.getSelectedIndex());  
 UpdatePurchasePanel3();  
 UpdateTable();  
 }  
 });  
 purchaseEditButton.addActionListener(ae -> {  
 if(sellerComboBoxPanel3.getSelectedIndex() == -1)  
 *showMessageDialog*(null, "Виберіть елемент для редагування");  
 else {  
 if (!goodsIdTF.getText().isEmpty()  
 && !goodsTotalTF.getText().isEmpty()  
 && !(purchaseComboBoxPanel3.getSelectedIndex() == -1)) {  
 purchases.set(purchaseComboBoxPanel3.getSelectedIndex(),  
 new Purchase(Integer.*parseInt*(goodsIdTF.getText()), Double.*parseDouble*(goodsTotalTF.getText()),  
 chainStores.get(storeComboBoxPanel3.getSelectedIndex()),  
 chainStores.get(storeComboBoxPanel3.getSelectedIndex()).getSellers().get(sellerComboBoxPanel3.getSelectedIndex())));  
  
 sellerLastNameTF.setText("");  
 sellerFirstNameTF.setText("");  
 sellerExperienceTF.setText("");  
 UpdatePurchasePanel3();  
 UpdateTable();  
 } else  
 *showMessageDialog*(null, "Заповніть всі поля вводу значеннями");  
 }  
 });  
 purchaseSaveButton.addActionListener(ae -> {  
 if(purchases.size() == 0)  
 *showMessageDialog*(null, "Немає жодного запису для збереження");  
 else {  
 try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream("purchase.dat"))) {  
 oos.writeObject(purchases);  
 } catch (Exception ex) {  
 System.*out*.println(ex.getMessage());  
 }  
 *showMessageDialog*(null, "Записи збережено у файл");  
 }  
 });  
 purchaseLoadButton.addActionListener(ae -> {  
 ArrayList<Purchase> list = new ArrayList<>();  
  
 try(ObjectInputStream ois = new ObjectInputStream(new FileInputStream("purchase.dat")))  
 {  
 list =(ArrayList<Purchase>) ois.readObject();  
 }  
 catch(Exception ex){ System.*out*.println(ex.getMessage()); }  
  
 if(list.size() == 0)  
 *showMessageDialog*(null, "Файл пустий або його не існує");  
 else {  
 purchases = list;  
 for (Purchase item : purchases)  
 {  
 chainStores.add(item.getStore());  
 sellers.add(item.getSeller());  
 }  
 UpdatePurchasePanel3();  
 UpdateStorePanel23();  
 UpdateSellerPanel12();  
 UpdateTable();  
 *showMessageDialog*(null, "Записи з файла успішно завантажено");  
 }  
 });  
  
 storeComboBoxPanel3.addActionListener(ae -> {  
 UpdateSellerPanel3();  
 });  
  
 sortButton.addActionListener(ae -> {  
 Collections.*sort*(purchases);  
 UpdateTable();  
 });  
 }  
  
 public void UpdateSellerPanel12() {  
 sellerComboBoxPanel1.removeAllItems();  
 sellerComboBoxPanel2.removeAllItems();  
 for(Seller item : sellers) {  
 sellerComboBoxPanel1.addItem(item.toString());  
 sellerComboBoxPanel2.addItem(item.toString());  
 }  
 sellerComboBoxPanel1.setSelectedIndex(-1);  
 sellerComboBoxPanel2.setSelectedIndex(-1);  
 }  
  
 public void UpdateStorePanel23() {  
 storeComboBoxPanel2.removeAllItems();  
 storeComboBoxPanel3.removeAllItems();  
 for(Store item : chainStores) {  
 storeComboBoxPanel2.addItem(item.toString());  
 storeComboBoxPanel3.addItem(item.toString());  
 }  
 storeComboBoxPanel2.setSelectedIndex(-1);  
 }  
  
 public void UpdatePurchasePanel3() {  
 purchaseComboBoxPanel3.removeAllItems();  
 for(Purchase item : purchases) {  
 purchaseComboBoxPanel3.addItem(item.toString());  
 }  
 purchaseComboBoxPanel3.setSelectedIndex(-1);  
 }  
  
 public void UpdateSellerPanel3() {  
 if (storeComboBoxPanel3.getSelectedIndex() == -1)  
 return;  
 else {  
 sellerComboBoxPanel3.removeAllItems();  
 for (Seller item : chainStores.get(storeComboBoxPanel3.getSelectedIndex()).getSellers()) {  
 sellerComboBoxPanel3.addItem(item.toString());  
 }  
 }  
 }  
  
 public void UpdateTable() {  
 for (int i = tableModel.getRowCount() - 1; i >= 0; i--)  
 tableModel.removeRow(i);  
  
 for (Purchase item : purchases) {  
 Object[] object = {item.getStore().getStoreN(), item.getStore().getCity(),  
 item.getSeller().getLastName(), item.getSeller().getFirstName(),  
 item.getSeller().getExperience(), item.getId(), item.getTotalCost()};  
 tableModel.addRow(object);  
 }  
 }  
  
 KeyListener intKeyListener = new KeyAdapter() {  
 public void keyTyped(KeyEvent e) {  
 char c = e.getKeyChar();  
 if (!((c >= '0') && (c <= '9') ||  
 (c == KeyEvent.*VK\_BACK\_SPACE*) ||  
 (c == KeyEvent.*VK\_DELETE*))) {  
 getToolkit().beep();  
 e.consume();  
 }  
 }  
 };  
  
 KeyListener doubleKeyListener = new KeyAdapter() {  
 public void keyTyped(KeyEvent e) {  
 char c = e.getKeyChar();  
 if (!((c >= '0') && (c <= '9') || (c == '.') ||  
 (c == KeyEvent.*VK\_BACK\_SPACE*) ||  
 (c == KeyEvent.*VK\_DELETE*))) {  
 getToolkit().beep();  
 e.consume();  
 }  
 }  
 };  
}

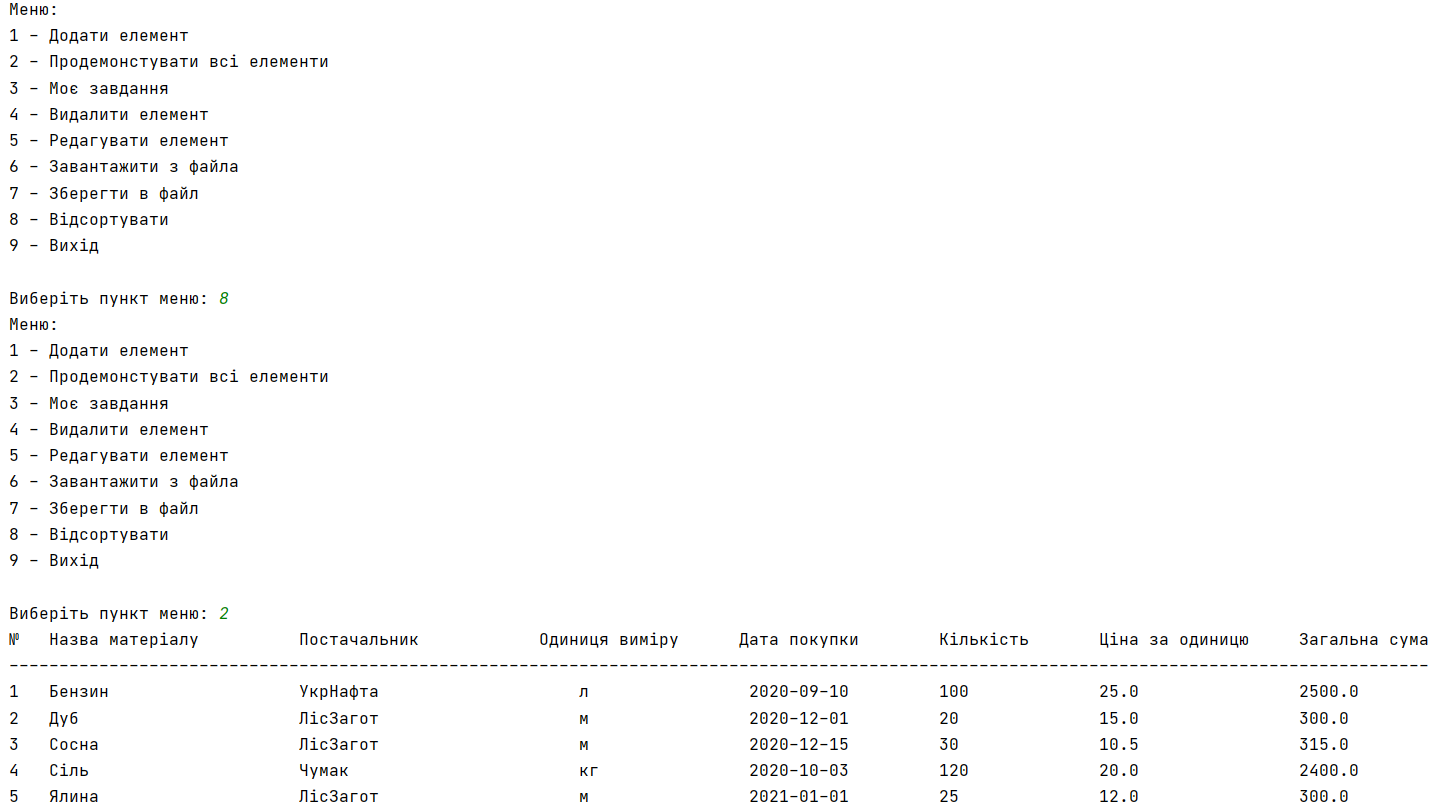
Materia.java

package com.company;  
  
import java.time.LocalDate;  
import java.util.Scanner;  
import java.io.Serializable;  
  
public class Material implements Serializable, Comparable<Material> {  
  
 private String name;  
 private String provider;  
 private String unitOfMeasurement;  
 private LocalDate buyDate;  
 private int year, month, day;  
 private int amount;  
 private double priceOfOne;  
 private double cost;  
  
 public Material()  
 {  
 name = "";  
 provider = "";  
 unitOfMeasurement = "";  
 buyDate = null;  
 year = 0;  
 month = 0;  
 day = 0;  
 amount = 0;  
 priceOfOne = 0;  
 cost = 0;  
 }  
  
 public String getName() { return name; }  
 public void setName(String name) { this.name = name; }  
  
 public String getProvider() { return provider; }  
 public void setProvider(String provider) { this.provider = provider; }  
  
 public String getUnitOfMeasurement() { return unitOfMeasurement; }  
 public void setUnitOfMeasurement(String unitOfMeasurement) { this.unitOfMeasurement = unitOfMeasurement; }  
  
 public LocalDate getBuyDate() { return buyDate; }  
 public void setBuyDate(LocalDate buyDate) { this.buyDate = buyDate; }  
  
 public int getYear() { return year; }  
 public void setYear(int year) { this.year = year; }  
  
 public int getMonth() { return month; }  
 public void setMonth(int month) { this.month = month; }  
  
 public int getDay() { return day; }  
 public void setDay(int day) { this.day = day; }  
  
 public int getAmount() { return amount; }  
 public void setAmount(int amount) { this.amount = amount; }  
  
 public double getPriceOfOne() { return priceOfOne; }  
 public void setPriceOfOne(double priceOfOne) { this.priceOfOne = priceOfOne; }  
  
 public double getCost() { return cost; }  
  
 public void Input()  
 {  
 Scanner sc = new Scanner(System.*in*);  
  
 System.*out*.print("Назва метеріалу: ");  
 name = sc.nextLine();  
  
 System.*out*.print("Постачальник: ");  
 provider = sc.nextLine();  
  
 System.*out*.print("Одиниція виміру: ");  
 unitOfMeasurement = sc.nextLine();  
  
 System.*out*.println("Дата покупки:");  
 System.*out*.print("День: ");  
 day = sc.nextInt();  
 System.*out*.print("Місяць: ");  
 month = sc.nextInt();  
 System.*out*.print("Рік: ");  
 year = sc.nextInt();  
 buyDate = LocalDate.*of*(year, month, day);  
  
 System.*out*.print("Кількість: ");  
 amount = sc.nextInt();  
  
 System.*out*.print("Ціна за одиницю: ");  
 priceOfOne = sc.nextDouble();  
  
 cost = amount \* priceOfOne;  
 }  
  
 public void Show()  
 {  
 System.*out*.printf("%-20s\t %-25s\t %-16s %-15s\t %-15s %-15s\t %-20s%n",  
 name, provider, unitOfMeasurement, buyDate, amount, priceOfOne, cost);  
 }  
  
 @Override  
 public int compareTo(Material o) {  
 return this.name.compareTo(o.name);  
 }  
}

Приклад роботи програми:





  
Рисунок 7 – Вивід програми

Завдання 7:

Код програми:

Main.java

package com.company;  
  
public class Main {  
 public Main() {  
 }  
  
 public static void main(String[] args) {  
 Window window = new Window();  
 window.setVisible(true);  
 }  
}

Window.java

package com.company;  
  
import javax.swing.\*;  
import javax.swing.table.DefaultTableModel;  
import java.awt.\*;  
import java.awt.event.KeyAdapter;  
import java.awt.event.KeyEvent;  
import java.awt.event.KeyListener;  
import java.io.FileInputStream;  
import java.io.FileOutputStream;  
import java.io.ObjectInputStream;  
import java.io.ObjectOutputStream;  
import java.util.ArrayList;  
import java.util.Collections;  
  
import static javax.swing.JOptionPane.*showMessageDialog*;  
  
public class Window extends JFrame {  
  
 JTabbedPane tabbedPane = new JTabbedPane();  
 JPanel panel1 = new JPanel();  
 JPanel panel1Lb = new JPanel();  
 JPanel panel1Btn = new JPanel();  
 JPanel panel2 = new JPanel();  
 JPanel panel2Lb = new JPanel();  
 JPanel panel2Btn = new JPanel();  
 JPanel panel3 = new JPanel();  
 JPanel panel3Lb = new JPanel();  
 JPanel panel3Btn = new JPanel();  
 JPanel panel4 = new JPanel();  
 JPanel panel4Btn = new JPanel();  
 JPanel panel4Tab = new JPanel();  
  
 *//Tab 1 textField* JTextField storeNTF = new JTextField();  
 JTextField storeCityTF = new JTextField();  
 *//Tab 2 textField* JTextField sellerLastNameTF = new JTextField();  
 JTextField sellerFirstNameTF = new JTextField();  
 JTextField sellerExperienceTF = new JTextField();  
 *//Tab 3 textField* JTextField goodsIdTF = new JTextField();  
 JTextField goodsTotalTF = new JTextField();  
  
 *//Tab 1 comboBox* JComboBox sellerComboBoxPanel1 = new JComboBox();  
 *//Tab2 comboBox* JComboBox storeComboBoxPanel2 = new JComboBox();  
 JComboBox sellerComboBoxPanel2 = new JComboBox();  
 *//Tab3 comboBox* JComboBox storeComboBoxPanel3 = new JComboBox();  
 JComboBox sellerComboBoxPanel3 = new JComboBox();  
 JComboBox purchaseComboBoxPanel3 = new JComboBox();  
  
 *//Tab 1 buttons* JButton sellerAddButton = new JButton("Додати");  
 JButton sellerDelButton = new JButton("Видалити");  
 JButton sellerEditButton = new JButton("Редагувати");  
 JButton sellerLoadButton = new JButton("Завантажити з файла");  
 JButton sellerSaveButton = new JButton("Зберегти у файл");  
 *//Tab 2 buttons* JButton storeAddButton = new JButton("Додати");  
 JButton storeDelButton = new JButton("Видалити");  
 JButton storeEditButton = new JButton("Редагувати");  
 JButton storeLoadButton = new JButton("Завантажити з файла");  
 JButton storeSaveButton = new JButton("Зберегти у файл");  
 JButton addSellerButton = new JButton("Додати продавця");  
 *//Tab 3 buttons* JButton purchaseAddButton = new JButton("Додати");  
 JButton purchaseDelButton = new JButton("Видалити");  
 JButton purchaseEditButton = new JButton("Редагувати");  
 JButton purchaseLoadButton = new JButton("Завантажити з файла");  
 JButton purchaseSaveButton = new JButton("Зберегти у файл");  
 *//Tab 4 buttons* JButton sortButton = new JButton("Сортувати");  
  
 ArrayList<Store> chainStores = new ArrayList<>();  
 ArrayList<Seller> sellers = new ArrayList<>();  
 ArrayList<Purchase> purchases = new ArrayList<>();  
  
 String[] col = {"№ магазину", "Місто", "Прізвище продавця", "Ім'я продавця", "Стаж", "ID покупки", "Сума покупки"};  
 DefaultTableModel tableModel = new DefaultTableModel(col, 0);  
  
 JTable table = new JTable(tableModel);  
  
 public Window() {  
 super("Мережа магазинів");  
 this.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
 this.setBounds(400, 100, 850, 400);  
 this.add(tabbedPane);  
  
 tabbedPane.addTab("Продавець", panel1);  
 tabbedPane.addTab("Магазин", panel2);  
 tabbedPane.addTab("Покупка", panel3);  
 tabbedPane.addTab("Таблиця", panel4);  
  
 *//Tab 1 layouts* panel1.setLayout(new GridLayout(2, 1, 1, 5));  
 panel1Lb.setLayout(new GridLayout(4, 2, 1, 5));  
 panel1Btn.setLayout(new GridLayout(2, 3, 1, 5));  
 *//Tab 2 layouts* panel2.setLayout(new GridLayout(2, 1, 1, 5));  
 panel2Lb.setLayout(new GridLayout(4, 2, 1, 5));  
 panel2Btn.setLayout(new GridLayout(2, 3, 1, 5));  
 *//Tab 3 layout* panel3.setLayout(new GridLayout(2, 1, 1, 5));  
 panel3Lb.setLayout(new GridLayout(5, 2, 1, 5));  
 panel3Btn.setLayout(new GridLayout(2, 3, 1, 5));  
 *//Tab 4* panel4.setLayout(new GridLayout(2, 1, 1, 5));  
 panel4Tab.setLayout(new GridLayout(1, 1, 1,5));  
 panel4Btn.setLayout(new GridLayout(3, 1, 1,5));  
  
 *//Tab 1 add* panel1Lb.add(new JLabel("Прізвище продавця", SwingConstants.*CENTER*));  
 panel1Lb.add(sellerLastNameTF);  
 panel1Lb.add(new JLabel("Ім'я продавця", SwingConstants.*CENTER*));  
 panel1Lb.add(sellerFirstNameTF);  
 panel1Lb.add(new JLabel("Стаж продавця", SwingConstants.*CENTER*));  
 panel1Lb.add(sellerExperienceTF);  
 panel1Lb.add(new JLabel("Список продавців", SwingConstants.*CENTER*));  
 panel1Lb.add(sellerComboBoxPanel1);  
 panel1.add(panel1Lb);  
 panel1.add(panel1Btn);  
 panel1Btn.add(sellerAddButton);  
 panel1Btn.add(sellerDelButton);  
 panel1Btn.add(sellerEditButton);  
 panel1Btn.add(sellerSaveButton);  
 panel1Btn.add(sellerLoadButton);  
 *//Tab 2 add* panel2Lb.add(new JLabel("Список продавців", SwingConstants.*CENTER*));  
 panel2Lb.add(sellerComboBoxPanel2);  
 panel2Lb.add(new JLabel("Номер магазину", SwingConstants.*CENTER*));  
 panel2Lb.add(storeNTF);  
 panel2Lb.add(new JLabel("Місто", SwingConstants.*CENTER*));  
 panel2Lb.add(storeCityTF);  
 panel2Lb.add(new JLabel("Список магазинів", SwingConstants.*CENTER*));  
 panel2Lb.add(storeComboBoxPanel2);  
 panel2.add(panel2Lb);  
 panel2.add(panel2Btn);  
 panel2Btn.add(storeAddButton);  
 panel2Btn.add(storeDelButton);  
 panel2Btn.add(storeEditButton);  
 panel2Btn.add(storeSaveButton);  
 panel2Btn.add(storeLoadButton);  
 panel2Btn.add(addSellerButton);  
 *//Tab 3 add* panel3Lb.add(new JLabel("Список магазинів", SwingConstants.*CENTER*));  
 panel3Lb.add(storeComboBoxPanel3);  
 panel3Lb.add(new JLabel("Список продавців", SwingConstants.*CENTER*));  
 panel3Lb.add(sellerComboBoxPanel3);  
 panel3Lb.add(new JLabel("ID покупки", SwingConstants.*CENTER*));  
 panel3Lb.add(goodsIdTF);  
 panel3Lb.add(new JLabel("Загальна сума покупки", SwingConstants.*CENTER*));  
 panel3Lb.add(goodsTotalTF);  
 panel3Lb.add(new JLabel("Список покупок", SwingConstants.*CENTER*));  
 panel3Lb.add(purchaseComboBoxPanel3);  
 panel3.add(panel3Lb);  
 panel3.add(panel3Btn);  
 panel3Btn.add(purchaseAddButton);  
 panel3Btn.add(purchaseDelButton);  
 panel3Btn.add(purchaseEditButton);  
 panel3Btn.add(purchaseSaveButton);  
 panel3Btn.add(purchaseLoadButton);  
  
 storeNTF.addKeyListener(intKeyListener);  
 sellerExperienceTF.addKeyListener(intKeyListener);  
 goodsIdTF.addKeyListener(intKeyListener);  
 goodsTotalTF.addKeyListener(doubleKeyListener);  
  
 storeNTF.setHorizontalAlignment(JTextField.*CENTER*);  
 storeCityTF.setHorizontalAlignment(JTextField.*CENTER*);  
 sellerLastNameTF.setHorizontalAlignment(JTextField.*CENTER*);  
 sellerFirstNameTF.setHorizontalAlignment(JTextField.*CENTER*);  
 sellerExperienceTF.setHorizontalAlignment(JTextField.*CENTER*);  
 goodsIdTF.setHorizontalAlignment(JTextField.*CENTER*);  
 goodsTotalTF.setHorizontalAlignment(JTextField.*CENTER*);  
  
 table.setRowHeight(24);  
 panel4Tab.add(table);  
 panel4Tab.add(new JScrollPane(table));  
 panel4Tab.add(table.getTableHeader());  
 table.setDefaultEditor(Object.class, null);  
 panel4Btn.add(sortButton);  
 panel4.add(panel4Tab);  
 panel4.add(panel4Btn);  
  
  
 *//Seller* sellerAddButton.addActionListener(ae -> {  
 if(!sellerLastNameTF.getText().isEmpty()  
 && !sellerFirstNameTF.getText().isEmpty()  
 && !sellerExperienceTF.getText().isEmpty()) {  
 sellers.add(new Seller(sellerLastNameTF.getText(), sellerFirstNameTF.getText(),  
 Integer.*parseInt*(sellerExperienceTF.getText())));  
  
 sellerLastNameTF.setText("");  
 sellerFirstNameTF.setText("");  
 sellerExperienceTF.setText("");  
 UpdateSellerPanel12();  
 }  
 else  
 *showMessageDialog*(null, "Заповніть всі поля вводу значеннями");  
 });  
 sellerDelButton.addActionListener(ae -> {  
 if(sellerComboBoxPanel1.getSelectedIndex() == -1)  
 *showMessageDialog*(null, "Виберіть елемент для видалення");  
 else {  
 sellers.remove(sellerComboBoxPanel1.getSelectedIndex());  
 UpdateSellerPanel12();  
 }  
 });  
 sellerEditButton.addActionListener(ae -> {  
 if(sellerComboBoxPanel1.getSelectedIndex() == -1)  
 *showMessageDialog*(null, "Виберіть елемент для редагування");  
 else {  
 if (!sellerLastNameTF.getText().isEmpty()  
 && !sellerFirstNameTF.getText().isEmpty()  
 && !sellerExperienceTF.getText().isEmpty()) {  
 sellers.set(sellerComboBoxPanel1.getSelectedIndex(),  
 new Seller(sellerLastNameTF.getText(), sellerFirstNameTF.getText(),  
 Integer.*parseInt*(sellerExperienceTF.getText())));  
  
 sellerLastNameTF.setText("");  
 sellerFirstNameTF.setText("");  
 sellerExperienceTF.setText("");  
 UpdateSellerPanel12();  
 *showMessageDialog*(null, "Запис редаговано");  
 } else  
 *showMessageDialog*(null, "Заповніть всі поля вводу значеннями");  
 }  
 });  
 sellerSaveButton.addActionListener(ae -> {  
 if(sellers.size() == 0)  
 *showMessageDialog*(null, "Немає жодного запису для збереження");  
 else {  
 try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream("seller.dat"))) {  
 oos.writeObject(sellers);  
 } catch (Exception ex) {  
 System.*out*.println(ex.getMessage());  
 }  
 *showMessageDialog*(null, "Записи збережено у файл");  
 }  
 });  
 sellerLoadButton.addActionListener(ae -> {  
 ArrayList<Seller> list = new ArrayList<>();  
  
 try(ObjectInputStream ois = new ObjectInputStream(new FileInputStream("seller.dat")))  
 {  
 list =(ArrayList<Seller>) ois.readObject();  
 }  
 catch(Exception ex){ System.*out*.println(ex.getMessage()); }  
  
 if(list.size() == 0)  
 *showMessageDialog*(null, "Файл пустий або його не існує");  
 else {  
 sellers = list;  
 UpdateSellerPanel12();  
 *showMessageDialog*(null, "Записи з файла успішно завантажено");  
 }  
 });  
 *//Store* storeAddButton.addActionListener(ae -> {  
 if(!storeNTF.getText().isEmpty()  
 && !storeCityTF.getText().isEmpty()  
 && !(sellerComboBoxPanel2.getSelectedIndex() == -1)) {  
 chainStores.add(new Store(Integer.*parseInt*(storeNTF.getText()),  
 storeCityTF.getText(), sellers.get(sellerComboBoxPanel2.getSelectedIndex())));  
  
 sellers.remove(sellerComboBoxPanel2.getSelectedIndex());  
 storeNTF.setText("");  
 storeCityTF.setText("");  
 UpdateSellerPanel12();  
 UpdateStorePanel23();  
 }  
 else  
 *showMessageDialog*(null, "Заповніть всі поля вводу значеннями");  
 });  
 addSellerButton.addActionListener(ae -> {  
 if(storeComboBoxPanel2.getSelectedIndex() == -1 || sellerComboBoxPanel2.getSelectedIndex() == -1)  
 *showMessageDialog*(null, "Виберіть магазин та продавця");  
 else  
 {  
 chainStores.get(storeComboBoxPanel2.getSelectedIndex()).setSellers(sellers.get(sellerComboBoxPanel2.getSelectedIndex()));  
 sellers.remove(sellerComboBoxPanel2.getSelectedIndex());  
 UpdateSellerPanel12();  
 UpdateStorePanel23();  
 }  
 });  
 storeDelButton.addActionListener(ae -> {  
 if(storeComboBoxPanel2.getSelectedIndex() == -1)  
 *showMessageDialog*(null, "Виберіть елемент для видалення");  
 else {  
 for(Seller item : chainStores.get(storeComboBoxPanel2.getSelectedIndex()).getSellers())  
 {  
 sellers.add(item);  
 }  
 chainStores.remove(storeComboBoxPanel2.getSelectedIndex());  
 UpdateStorePanel23();  
 UpdateSellerPanel12();  
 }  
 });  
 storeEditButton.addActionListener(ae -> {  
 if(storeComboBoxPanel2.getSelectedIndex() == -1)  
 *showMessageDialog*(null, "Заповніть всі поля вводу значеннями");  
 else {  
 if (!storeNTF.getText().isEmpty() && !storeCityTF.getText().isEmpty()) {  
 chainStores.get(storeComboBoxPanel2.getSelectedIndex()).EditStore(  
 Integer.*parseInt*(storeNTF.getText()), storeCityTF.getText());  
  
 storeNTF.setText("");  
 storeCityTF.setText("");  
 UpdateStorePanel23();  
 } else  
 *showMessageDialog*(null, "Заповніть всі поля вводу значеннями");  
 }  
 });  
 storeSaveButton.addActionListener(ae -> {  
 if(chainStores.size() == 0)  
 *showMessageDialog*(null, "Немає жодного запису для збереження");  
 else {  
 try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream("store.dat"))) {  
 oos.writeObject(chainStores);  
 } catch (Exception ex) {  
 System.*out*.println(ex.getMessage());  
 }  
 *showMessageDialog*(null, "Записи збережено у файл");  
 }  
 });  
 storeLoadButton.addActionListener(ae -> {  
 ArrayList<Store> list = new ArrayList<>();  
  
 try(ObjectInputStream ois = new ObjectInputStream(new FileInputStream("store.dat")))  
 {  
 list =(ArrayList<Store>) ois.readObject();  
 }  
 catch(Exception ex){ System.*out*.println(ex.getMessage()); }  
  
 if(list.size() == 0)  
 *showMessageDialog*(null, "Файл пустий або його не існує");  
 else {  
 chainStores = list;  
 for (Store item : chainStores)  
 for (Seller seller : item.getSellers())  
 sellers.add(seller);  
  
 UpdatePurchasePanel3();  
 UpdateStorePanel23();  
 UpdateSellerPanel12();  
 *showMessageDialog*(null, "Записи з файла успішно завантажено");  
 }  
 });  
 *//Purchase* purchaseAddButton.addActionListener(ae -> {  
 if(!goodsIdTF.getText().isEmpty()  
 && !goodsTotalTF.getText().isEmpty()  
 && !(storeComboBoxPanel3.getSelectedIndex() == -1)  
 && !(sellerComboBoxPanel3.getSelectedIndex() == -1)) {  
 purchases.add(new Purchase(Integer.*parseInt*(goodsIdTF.getText()), Double.*parseDouble*(goodsTotalTF.getText()),  
 chainStores.get(storeComboBoxPanel3.getSelectedIndex()),  
 chainStores.get(storeComboBoxPanel3.getSelectedIndex()).getSellers().get(sellerComboBoxPanel3.getSelectedIndex())));  
  
 goodsIdTF.setText("");  
 goodsTotalTF.setText("");  
 UpdatePurchasePanel3();  
 UpdateTable();  
  
 }  
 else  
 *showMessageDialog*(null, "Заповніть всі поля вводу значеннями");  
 });  
 purchaseDelButton.addActionListener(ae -> {  
 if(purchaseComboBoxPanel3.getSelectedIndex() == -1)  
 *showMessageDialog*(null, "Виберіть елемент для видалення");  
 else {  
 purchases.remove(purchaseComboBoxPanel3.getSelectedIndex());  
 UpdatePurchasePanel3();  
 UpdateTable();  
 }  
 });  
 purchaseEditButton.addActionListener(ae -> {  
 if(sellerComboBoxPanel3.getSelectedIndex() == -1)  
 *showMessageDialog*(null, "Виберіть елемент для редагування");  
 else {  
 if (!goodsIdTF.getText().isEmpty()  
 && !goodsTotalTF.getText().isEmpty()  
 && !(purchaseComboBoxPanel3.getSelectedIndex() == -1)) {  
 purchases.set(purchaseComboBoxPanel3.getSelectedIndex(),  
 new Purchase(Integer.*parseInt*(goodsIdTF.getText()), Double.*parseDouble*(goodsTotalTF.getText()),  
 chainStores.get(storeComboBoxPanel3.getSelectedIndex()),  
 chainStores.get(storeComboBoxPanel3.getSelectedIndex()).getSellers().get(sellerComboBoxPanel3.getSelectedIndex())));  
  
 sellerLastNameTF.setText("");  
 sellerFirstNameTF.setText("");  
 sellerExperienceTF.setText("");  
 UpdatePurchasePanel3();  
 UpdateTable();  
 } else  
 *showMessageDialog*(null, "Заповніть всі поля вводу значеннями");  
 }  
 });  
 purchaseSaveButton.addActionListener(ae -> {  
 if(purchases.size() == 0)  
 *showMessageDialog*(null, "Немає жодного запису для збереження");  
 else {  
 try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream("purchase.dat"))) {  
 oos.writeObject(purchases);  
 } catch (Exception ex) {  
 System.*out*.println(ex.getMessage());  
 }  
 *showMessageDialog*(null, "Записи збережено у файл");  
 }  
 });  
 purchaseLoadButton.addActionListener(ae -> {  
 ArrayList<Purchase> list = new ArrayList<>();  
  
 try(ObjectInputStream ois = new ObjectInputStream(new FileInputStream("purchase.dat")))  
 {  
 list =(ArrayList<Purchase>) ois.readObject();  
 }  
 catch(Exception ex){ System.*out*.println(ex.getMessage()); }  
  
 if(list.size() == 0)  
 *showMessageDialog*(null, "Файл пустий або його не існує");  
 else {  
 purchases = list;  
 for (Purchase item : purchases)  
 {  
 chainStores.add(item.getStore());  
 sellers.add(item.getSeller());  
 }  
 UpdatePurchasePanel3();  
 UpdateStorePanel23();  
 UpdateSellerPanel12();  
 UpdateTable();  
 *showMessageDialog*(null, "Записи з файла успішно завантажено");  
 }  
 });  
  
 storeComboBoxPanel3.addActionListener(ae -> {  
 UpdateSellerPanel3();  
 });  
  
 sortButton.addActionListener(ae -> {  
 Collections.*sort*(purchases);  
 UpdateTable();  
 });  
 }  
  
 public void UpdateSellerPanel12() {  
 sellerComboBoxPanel1.removeAllItems();  
 sellerComboBoxPanel2.removeAllItems();  
 for(Seller item : sellers) {  
 sellerComboBoxPanel1.addItem(item.toString());  
 sellerComboBoxPanel2.addItem(item.toString());  
 }  
 sellerComboBoxPanel1.setSelectedIndex(-1);  
 sellerComboBoxPanel2.setSelectedIndex(-1);  
 }  
  
 public void UpdateStorePanel23() {  
 storeComboBoxPanel2.removeAllItems();  
 storeComboBoxPanel3.removeAllItems();  
 for(Store item : chainStores) {  
 storeComboBoxPanel2.addItem(item.toString());  
 storeComboBoxPanel3.addItem(item.toString());  
 }  
 storeComboBoxPanel2.setSelectedIndex(-1);  
 }  
  
 public void UpdatePurchasePanel3() {  
 purchaseComboBoxPanel3.removeAllItems();  
 for(Purchase item : purchases) {  
 purchaseComboBoxPanel3.addItem(item.toString());  
 }  
 purchaseComboBoxPanel3.setSelectedIndex(-1);  
 }  
  
 public void UpdateSellerPanel3() {  
 if (storeComboBoxPanel3.getSelectedIndex() == -1)  
 return;  
 else {  
 sellerComboBoxPanel3.removeAllItems();  
 for (Seller item : chainStores.get(storeComboBoxPanel3.getSelectedIndex()).getSellers()) {  
 sellerComboBoxPanel3.addItem(item.toString());  
 }  
 }  
 }  
  
 public void UpdateTable() {  
 for (int i = tableModel.getRowCount() - 1; i >= 0; i--)  
 tableModel.removeRow(i);  
  
 for (Purchase item : purchases) {  
 Object[] object = {item.getStore().getStoreN(), item.getStore().getCity(),  
 item.getSeller().getLastName(), item.getSeller().getFirstName(),  
 item.getSeller().getExperience(), item.getId(), item.getTotalCost()};  
 tableModel.addRow(object);  
 }  
 }  
  
 KeyListener intKeyListener = new KeyAdapter() {  
 public void keyTyped(KeyEvent e) {  
 char c = e.getKeyChar();  
 if (!((c >= '0') && (c <= '9') ||  
 (c == KeyEvent.*VK\_BACK\_SPACE*) ||  
 (c == KeyEvent.*VK\_DELETE*))) {  
 getToolkit().beep();  
 e.consume();  
 }  
 }  
 };  
  
 KeyListener doubleKeyListener = new KeyAdapter() {  
 public void keyTyped(KeyEvent e) {  
 char c = e.getKeyChar();  
 if (!((c >= '0') && (c <= '9') || (c == '.') ||  
 (c == KeyEvent.*VK\_BACK\_SPACE*) ||  
 (c == KeyEvent.*VK\_DELETE*))) {  
 getToolkit().beep();  
 e.consume();  
 }  
 }  
 };  
}

People.java

package com.company;  
  
import java.io.Serializable;  
  
public class People implements Serializable {  
 private String lastName;  
 private String firstName;  
  
 public People(String lastName, String firstName) {  
 this.lastName = lastName;  
 this.firstName = firstName;  
 }  
  
 public String getLastName() {  
 return lastName;  
 }  
 public void setLastName(String lastName) {  
 this.lastName = lastName;  
 }  
  
 public String getFirstName() {  
 return firstName;  
 }  
 public void setFirstName(String firstName) {  
 this.firstName = firstName;  
 }  
  
 @Override  
 public String toString() {  
 return "Прізвище: '" + this.lastName + "', Ім'я: '" + this.firstName + "'";  
 }  
}

Seller.java

package com.company;  
  
import java.io.Serializable;  
  
public class Seller extends People implements Serializable {  
 private int experience;  
  
 public Seller(String lastName, String firstName, int experience) {  
 super(lastName, firstName);  
 this.experience = experience;  
 }  
  
 public int getExperience() {  
 return experience;  
 }  
 public void setExperience(int experience) {  
 this.experience = experience;  
 }  
  
 @Override  
 public String toString() {  
 return "Прізвище: '" + this.getLastName() + "', Ім'я: '" + this.getFirstName() + "', Стаж: '" + this.experience + "'";  
 }  
}

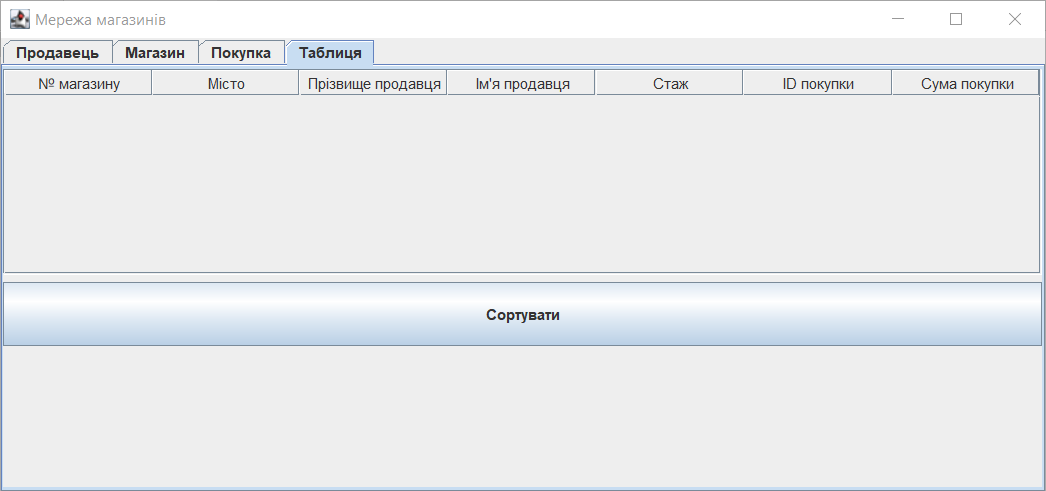
Store.java

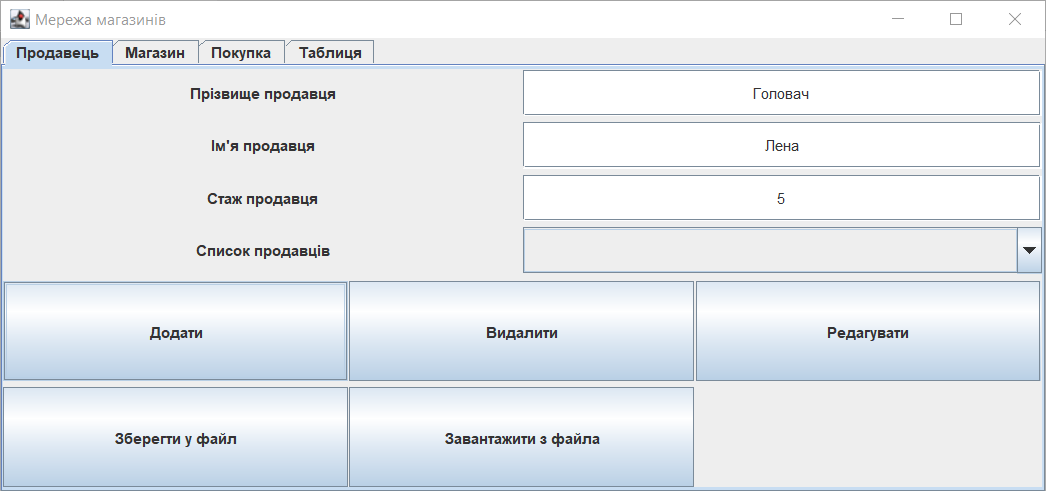
package com.company;  
  
import java.io.Serializable;  
import java.util.ArrayList;  
  
public class Store implements Serializable{  
 private int storeN;  
 private String city;  
 private ArrayList<Seller> sellers = new ArrayList<>();  
  
 public Store(int storeN, String city, Seller seller) {  
 this.storeN = storeN;  
 this.city = city;  
 this.sellers.add(seller);  
 }  
  
 public int getStoreN() {  
 return storeN;  
 }  
 public void setStoreN(int storeN) {  
 this.storeN = storeN;  
 }  
  
 public String getCity() {  
 return city;  
 }  
 public void setCity(String city) {  
 this.city = city;  
 }  
  
 public ArrayList<Seller> getSellers() {  
 return sellers;  
 }  
 public void setSellers(Seller sellers) {  
 this.sellers.add(sellers);  
 }  
  
 public void EditStore(int storeN, String city) {  
 this.storeN = storeN;  
 this.city = city;  
 }  
  
 @Override  
 public String toString() {  
 return "Магазин №: '" + this.storeN + "', Місто: '" + this.city +  
 "', Кількість продавців: '" + this.sellers.size() + " '";  
 }  
}

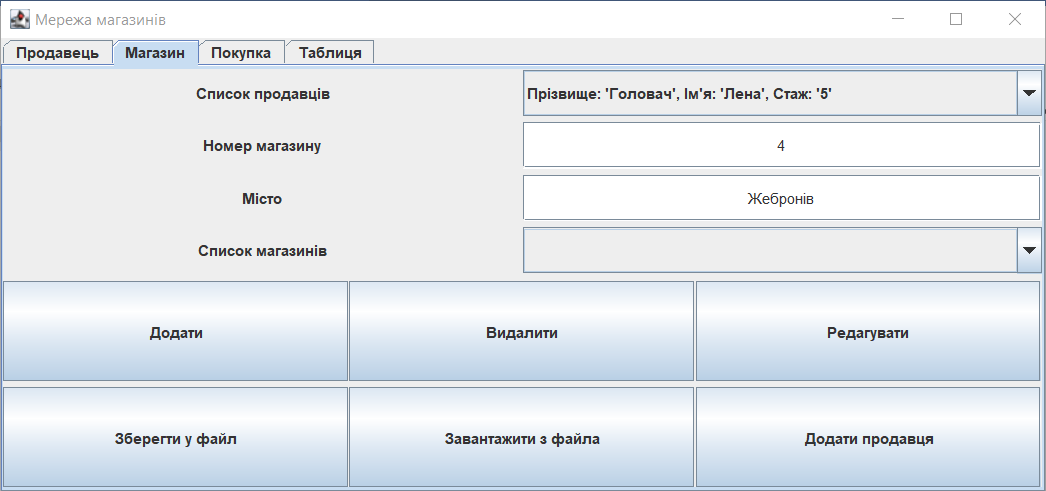
Purchase.java

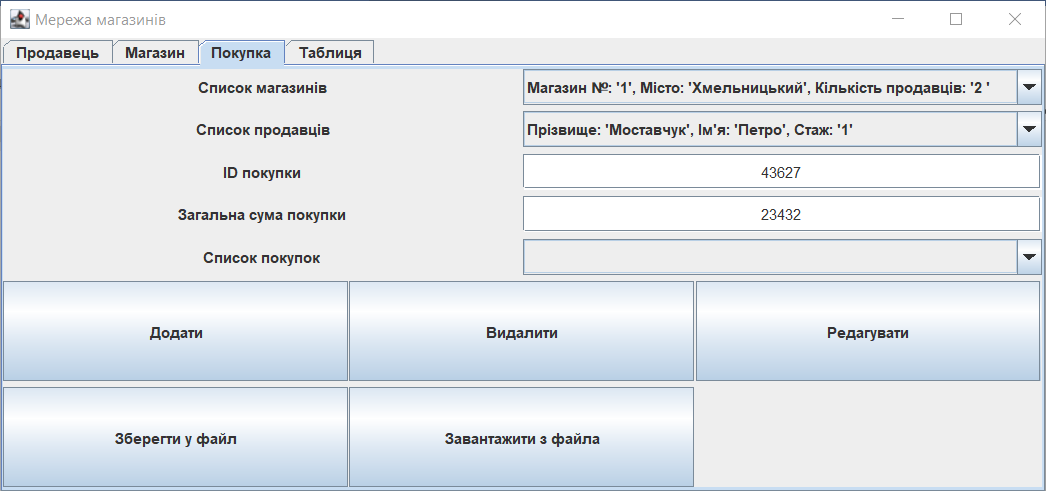
package com.company;  
  
import java.io.Serializable;  
  
public class Purchase implements Serializable, Comparable<Purchase> {  
 private int id;  
 private double totalCost;  
 private Store store;  
 private Seller seller;  
  
 public Purchase(int id, double totalCost, Store store, Seller seller) {  
 this.id = id;  
 this.totalCost = totalCost;  
 this.store = store;  
 this.seller = seller;  
 }  
  
 public int getId() {  
 return id;  
 }  
  
 public void setId(int id) {  
 this.id = id;  
 }  
  
 public double getTotalCost() {  
 return totalCost;  
 }  
  
 public void setTotalCost(double totalCost) {  
 this.totalCost = totalCost;  
 }  
  
 public Store getStore() {  
 return store;  
 }  
  
 public void setStore(Store store) {  
 this.store = store;  
 }  
  
 public Seller getSeller() {  
 return seller;  
 }  
  
 public void setSeller(Seller seller) {  
 this.seller = seller;  
 }  
  
 @Override  
 public int compareTo(Purchase o) {  
 return Double.*compare*(this.totalCost, o.totalCost);  
 }  
  
 @Override  
 public String toString() {  
 return "ID: '" + this.id + "', Сума покупки: '" + this.totalCost + "'";  
 }  
}

Приклад роботи програми:

  
Рисунок 8 – Початкове вікно програми

  
Рисунок 9 – Вкладка для редагування та видалення продавців

  
Рисунок 10 – Вкладка для редагування та видалення магазинів

  
Рисунок 11 – Вкладка для редагування та видалення покупок

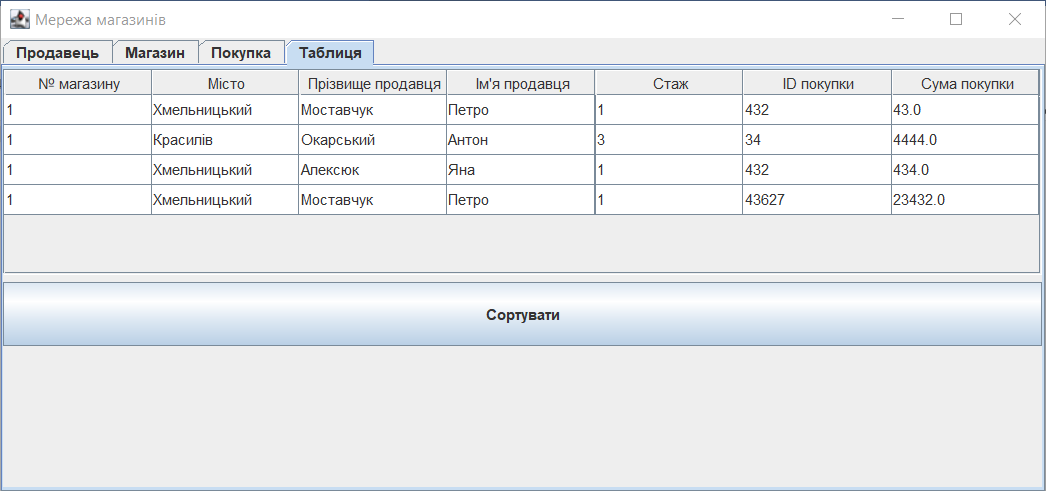


Рисунок 12 – Вивід даних

**Висновки:** Я набув навиків розробки об'єктно-орієнтованих програм використовуючи мову програмування Java, покращив своє розуміння концепції об'єктно-орієнтованого програмування.