Національний університет «Києво-Могилянська академія»

Факультет інформатики

Інженерія програмного забезпечення

ЗВІТ

Про виконання лабораторної роботи № 2

ПРОГРАМУВАННЯ ЦИКЛІЧНИХ ПРОЦЕСІВ В С

з дисципліни: Основи програмування

**ЗВІТ**

Про виконання лабораторної роботи №2

«Автоматизоване робоче місце» з дисципліни:

Алгоритми та структури даних

Виконали:

Гапоненко Єлизавета Артемівна,

Розумєй Максим Віталійович

**Зміст**

1. [Постановка задачі.](#ПостановкаЗадачі)
2. [Розподіл ролей в групі.](#РозподілРолей)
3. [Опис усіх реалізованих можливостей.](#ОписМожливостей)
4. [Cтруктура програми - UML-діаграма та опис усіх написаних класів і методів.](#СтруктураПрограми)
5. [Проблеми в роботі.](#Проблеми)
6. [Висновки.](#Висновки)
7. [Додаток. Код програми.](#КодПрограми)

**Постановка задачі**

**Мета:** розробити систему для автоматизації обліку невеликого підприємства з можливістю додавання, редагування, видалення товарів та груп товарів, а також виконання операцій зі списання та поставки товарів, аналізу статистичних даних та пошуку товарів.

**Опис завдання:** група студентів повинна реалізувати систему у вигляді графічного інтерфейсу користувача, яка буде дозволяти вести облік товарів на складі підприємства.

**Вимоги:**

* додавання, редагування та видалення груп товарів
* додавання, редагування та видалення товарів в групах
* інтерфейс для списання та додавання товарів на склад
* пошук товарів
* виведення статистичних даних.

**Розподіл ролей в групі**

Гапоненко Єлизавета Артемівна:

Загальна архітектура проєкту, дизайн вікон, розміщення елементів, робота з файлами, дебаг.

Розумєй Максим Віталійович:

Зведення усіх вікон до одного головного вікна, робота з інформацією усередині програми, обробка виняткових ситуацій, дебаг.

**Опис усіх реалізованих можливостей**

У інтерфейсі лабораторної роботи реалізовано:

1. Додавання, редагування, видалення груп товарів
2. Додавання, редагування, видалення товарів
3. Списання та поставка товарів
4. Пошук товару за назвою
5. Вивід статичних даних
6. Збереження груп товарів в тестовий файл
7. Збереження всіх товарів в текстовий файл
8. Перехід між всіма підпрограмами

**Cтруктура програми**

**Класи програми:**

1. **Main**

Головний клас, з якого запускається програма. Містить головний масив груп, здатен вносити зміни до нього. Завантажує інформацію з файлів при запуску програми.

1. **Interface**

Клас, який є головним вікном, де можна обирати функцію та переходити на відповідне інше вікно. Поки інше вікно не буде закрите, доступу до даного вікна не буде.

1. **Items**

Клас, в якому створено товар з усіма його характеристиками(назва, опис, виробник, кількість на складі та ціна).

1. **GroupOfItems**

Клас, в якому створено групу товарів з такими характеристиками: назва, опис. Містить масив товарів, реалізовано методи додавання, редагування, видалення, постачання, списання товару.

1. **EditProducts**

Клас з графічним інтерефейсом взаємодії з товарами або їх групами (графічний інтерфейс додавання, редагування, видалення)

1. **ProductSearch**

Класз графічним інтерефейсом пошуку товарів. Підтримує спеціальні символи для пошуку: «\*» - будь-яка кількість будь-яких символів, «?» - один будь-який символ.

1. **Statistics**

Класз графічним інтерефейсом виведення статистичної інформації, а саме: загальна вартість усіх товарів складу та усі товари з їх інформацією та загальна вартість усіх товарів групи та усі товари групи з їх інформацією.

1. **SupplyOfProducts**

Класз графічним інтерефейсом списання та постачання товарів.

**[UML-діаграма](UML%20class.pdf)**

**Проблеми в роботі**

1. При роботі у нових класах-вікнах небуло можливості отримати початковий масив з даними та працювати з ним.

Рішення: передавати об’єкт головного класу як параметр для створення нових вікон.

1. Так як у файлах використовується символ «;» як розділювач, то при додаванні будь-якої інформації, яка містить цей символ, програма не зможе прочитати файли при наступному запуску.  
     
   Рішення: замінити усі входження даного символу на порожній рядок
2. При повторному запуску одного й того ж вікна елементи дублювалися на екрані.  
     
   Рішення: додати перевірку, чи не null хоча б один з елементів вікна
3. При пошуку товарів з використанням додаткових символів треба використовувати перетворення даного рядка на регекс, і тому назви, що містять спеціальні символи, будуть заважати роботі функції пошуку товарів.  
     
   Рішення: дозволити використання лише літер та цифр для назв товарів.

**Висновки**

У цій лабораторній роботі ми реалізували графічний інтерфейс складського приміщення з усіма необхідними функціями, такими як: створення/редагування/видалення групи товарів або товарів, пошук товарів, поставка/списання та виведення інформації по складу за різними критеріями.

**Код програми**

1. **[Клас Items](#Items)**
2. [**Клас GroupOfItems**](#GroupOfItems)
3. [**Клас EditProducts**](#EditProducts)
4. [**Клас Interface**](#Interface)
5. [**Клас ProductsSearch**](#ProductsSearch)
6. [**Клас Statistics**](#Statistics)
7. [**Клас SupplyOfProducts**](#SupplyOfProducts)
8. [**Клас Main**](#Main)

public class **Items** {

/\*\*Назва товару \*/

private String name;

/\*\*Опис товару \*/

private String description;

/\*\*Виробник товару \*/

private String producer;

/\*\*Кількість товару на складі\*/

private int count;

/\*\*Ціна за одиницю \*/

private int pricePerOne;

public Items(String name, String description, String producer, int count, int pricePerOne){

this.name = name;

this.description = description;

this.producer = producer;

this.count = count;

this.pricePerOne = pricePerOne;

}

// Getter methods

public String getName() {

return name;

}

public String getDescription() {

return description;

}

public String getProducer() {

return producer;

}

public int getCount() {

return count;

}

public int getPricePerOne() {

return pricePerOne;

}

public void setPricePerOne(int pricePerOne) {

this.pricePerOne = pricePerOne;

}

public void setCount(int count) {

this.count = count;

}

public void setProducer(String producer) {

this.producer = producer;

}

public void setDescription(String description) {

this.description = description;

}

public void setName(String name) {

this.name = name;

}

public String toString(){

return "\nНазва товару: " + name +

"\nОпис товару: " + description +

"\nВиробник: " + producer +

"\nКількість на складі: " + count +

"\nЦіна за одиницю: " + pricePerOne;

}

}

import java.util.ArrayList;

public class **GroupOfItems** {

private Main main;

/\*\*Масив для зберігання товарів \*/

public ArrayList<Items> productsList = new ArrayList<>();

/\*\*Назва групи товарів \*/

private String nameOfGroup;

/\*\*Опис групи товарів \*/

private String groupDescription;

public GroupOfItems(String nameOfGroup, String descriptionOfGroup, Main main){

this.main = main;

this.nameOfGroup = nameOfGroup;

this.groupDescription = descriptionOfGroup;

}

public String getNameOfGroup() {

return nameOfGroup;

}

public String getGroupDescription() {

return groupDescription;

}

public void setNameOfGroup(String nameOfGroup) {

this.nameOfGroup = nameOfGroup;

}

public void setGroupDescription(String groupDescription) {

this.groupDescription = groupDescription;

}

public String toString(){

return "\nНазва групи товарів: " + nameOfGroup +

"\nОпис групи товарів: " + groupDescription;

}

public void addProduct(String name, String desc, String producer, int count, int pricePerOne) throws IllegalArgumentException{

if (main.productExists(name)){

throw new IllegalArgumentException("Товар з такою назвою вже існує.");

} else if (name.trim().isEmpty()){

throw new IllegalArgumentException("Назва товару не може бути порожньою.");

} else if (desc.trim().isEmpty()){

throw new IllegalArgumentException("Опис товару не може бути порожнім.");

} else if (producer.trim().isEmpty()){

throw new IllegalArgumentException("Виробник товару не може бути порожнім.");

} else if (count < 0){

throw new IllegalArgumentException("Кількість товару не може бути від'ємною.");

} else if (pricePerOne <= 0){

throw new IllegalArgumentException("Ціна за одиницю товару повинна бути більше 0.");

}

productsList.add(new Items(name, desc, producer, count, pricePerOne));

}

public void editProduct(String oldName, String newName, String newDescription, String newProducer, String newCount, String newPricePerOne) throws IllegalArgumentException{

if (!newCount.trim().isEmpty()) {

try {

Integer.parseInt(newCount);

} catch (NumberFormatException e) {

throw new IllegalArgumentException("Кількість товару повинна бути числом.");

}

}

if (!newPricePerOne.trim().isEmpty()) {

try {

Integer.parseInt(newPricePerOne);

} catch (NumberFormatException e) {

throw new IllegalArgumentException("Ціна за одиницю товару повинна бути числом.");

}

}

if (!main.productExists(oldName)){

throw new IllegalArgumentException("Товар з такою назвою не існує.");

} else if (main.productExists(newName) && newDescription.trim().isEmpty() && newProducer.trim().isEmpty() && newCount.trim().isEmpty() && newPricePerOne.trim().isEmpty()){

throw new IllegalArgumentException("Товар має таку саму назву");

} else if (newName.trim().isEmpty() && newDescription.trim().isEmpty() && newProducer.trim().isEmpty() && newCount.trim().isEmpty() && newPricePerOne.trim().isEmpty()){

throw new IllegalArgumentException("Зміни не введено.");

} else if (newName.trim().isEmpty() && !newDescription.trim().isEmpty() && getProductByName(oldName).getDescription().equals(newDescription) && newProducer.trim().isEmpty() && newCount.trim().isEmpty() && newPricePerOne.trim().isEmpty()){

throw new IllegalArgumentException("Товар має такий самий опис.");

} else if (newName.trim().isEmpty() && newDescription.trim().isEmpty() && !newProducer.trim().isEmpty() && getProductByName(oldName).getProducer().equals(newProducer) && newCount.trim().isEmpty() && newPricePerOne.trim().isEmpty()){

throw new IllegalArgumentException("Товар має такого самого виробника.");

} else if (newName.trim().isEmpty() && newDescription.trim().isEmpty() && newProducer.trim().isEmpty() && !newCount.trim().isEmpty() && getProductByName(oldName).getCount() == Integer.parseInt(newCount) && newPricePerOne.trim().isEmpty()){

throw new IllegalArgumentException("Товар має таку саму кількість.");

} else if (newName.trim().isEmpty() && newDescription.trim().isEmpty() && newProducer.trim().isEmpty() && newCount.trim().isEmpty() && !newPricePerOne.trim().isEmpty() && getProductByName(oldName).getPricePerOne() == Integer.parseInt(newPricePerOne)){

throw new IllegalArgumentException("Товар має таку саму ціну за одиницю.");

}

if (!newCount.trim().isEmpty() && Integer.parseInt(newCount) < 0) {

throw new IllegalArgumentException("Кількість товару не може бути від'ємною.");

} else if (!newPricePerOne.trim().isEmpty() && Integer.parseInt(newPricePerOne) <= 0) {

throw new IllegalArgumentException("Ціна за одиницю товару повинна бути більше 0.");

}

for (Items product : productsList) {

if (product.getName().equals(oldName)) {

if (!newName.trim().isEmpty()) {

product.setName(newName);

}

if (!newDescription.trim().isEmpty()) {

product.setDescription(newDescription);

}

if (!newProducer.trim().isEmpty()) {

product.setProducer(newProducer);

}

if (!newCount.trim().isEmpty()) {

product.setCount(Integer.parseInt(newCount));

}

if (!newPricePerOne.trim().isEmpty()) {

product.setPricePerOne(Integer.parseInt(newPricePerOne));

}

break;

}

}

}

public void removeProduct(String name) throws IllegalArgumentException{

if (!main.productExists(name)){

throw new IllegalArgumentException("Товар з такою назвою не існує.");

}

for (Items product : productsList){

if (product.getName().equals(name)){

productsList.remove(product);

break;

}

}

}

public Items getProductByName(String name){

for (Items product : productsList){

if (product.getName().equals(name)){

return product;

}

}

return null;

}

public int getTotalValue(){

int totalValue = 0;

for (Items product : productsList){

totalValue += product.getCount() \* product.getPricePerOne();

}

return totalValue;

}

public void subtractProduct(String name, String count) throws IllegalArgumentException{

if (!main.productExists(name)){

throw new IllegalArgumentException("Товар з такою назвою не існує.");

} else if (count == null || count.trim().isEmpty()){

throw new IllegalArgumentException("Кількість товару не введено.");

}

int countInt;

try {

countInt = Integer.parseInt(count);

} catch (NumberFormatException e) {

throw new IllegalArgumentException("Кількість товару повинна бути числом.");

}

if (countInt <= 0){

throw new IllegalArgumentException("Кількість товару повинна бути більше 0.");

}

Items product = getProductByName(name);

if (product.getCount() - countInt < 0){

throw new IllegalArgumentException("На складі недостатньо товару.");

}

product.setCount(product.getCount() - countInt);

}

public void supplyProduct(String name, String count) throws IllegalArgumentException{

if (!main.productExists(name)){

throw new IllegalArgumentException("Товар з такою назвою не існує.");

} else if (count == null || count.trim().isEmpty()){

throw new IllegalArgumentException("Кількість товару не введено.");

}

int countInt;

try {

countInt = Integer.parseInt(count);

} catch (NumberFormatException e) {

throw new IllegalArgumentException("Кількість товару повинна бути числом.");

}

if (countInt <= 0){

throw new IllegalArgumentException("Кількість товару повинна бути більше 0.");

}

Items product = getProductByName(name);

getProductByName(name).setCount(product.getCount() + countInt);

}

}

import javax.swing.\*;

import java.awt.\*;

import java.io.\*;

import static java.lang.Integer.parseInt;

public class **EditProducts** extends JDialog {

private final Main main;

/\*\*Вибір функції: група товарів/товар \*/

private JComboBox<String> chooseSubject;

/\*\*Вибір функції \*/

private JComboBox<String> chooseFunction;

/\*\*Ввести назву товару/ групи товарів \*/

private JTextField nameOfGroup;

/\*\*Ввести опис групи товарів \*/

private JTextField descriptionOfGroup;

/\*\*Ввести назву товару \*/

private JTextField nameOfProduct;

/\*\*Ввести опис товару \*/

private JTextField descriptionOfProduct;

/\*\*Ввести виробника товару \*/

private JTextField producerOfProduct;

/\*\*Ввести кількість товарів \*/

private JTextField countOfProduct;

/\*\*Ввести ціну товару \*/

private JTextField priceOfProduct;

/\*\*Вибір групи товарів, куди додати товар \*/

public JComboBox<String> groupNameBox;

/\*\*Вибір товару при видаленні \*/

public JComboBox<String> productNameBox;

/\*\*Кнопка "зберегти" \*/

public JButton saveButton;

/\*\*Напис назви \*/

private JLabel typeName;

/\*\*Напис опису \*/

private JLabel typeDescription;

/\*\*Напис виробника \*/

private JLabel typeProducer;

/\*\*Напис кількості \*/

private JLabel typeCount;

/\*\*Напис ціни \*/

private JLabel typePrice;

private volatile boolean isUpdating = false;

private static final int WIDTH\_OF\_FRAME = 800;

private static final int HEIGHT\_OF\_FRAME = 700;

private static final int WIDTH\_OF\_FIELD = 300;

private static final int HEIGHT\_OF\_FIELD = 70;

private static final int WIDTH\_OF\_BUTTON = 170;

private static final int HEIGHT\_OF\_BUTTON = 70;

public EditProducts(String name, Main main, JFrame parent) {

super(parent, name, true);

this.main = main;

this.setTitle(name);

this.setSize(WIDTH\_OF\_FRAME, HEIGHT\_OF\_FRAME);

this.getContentPane().setLayout(null);

this.getContentPane().setBackground(new Color(204, 255, 153, 250));

this.setLocationRelativeTo(null);

}

public void start(){

init();

updateInformation();

this.setVisible(true);

}

private void init() {

if (chooseSubject != null){

return;

}

chooseSubject = new JComboBox<>();

chooseSubject.addItem("групу товарів");

chooseSubject.addItem("товар");

DefaultListCellRenderer renderer = new DefaultListCellRenderer();

renderer.setHorizontalAlignment(SwingConstants.CENTER);

chooseSubject.setRenderer(renderer);

Font font = chooseSubject.getFont();

chooseSubject.setFont(font.deriveFont(Font.BOLD, 16f));

chooseSubject.setBounds(WIDTH\_OF\_FRAME /2 + 10, HEIGHT\_OF\_FRAME /2 - 300 , WIDTH\_OF\_FIELD, HEIGHT\_OF\_FIELD);

this.getContentPane().add(chooseSubject);

chooseSubject.addActionListener(e -> {

functionActionPerformed();

});

chooseFunction = new JComboBox<>();

chooseFunction.addItem("Додати");

chooseFunction.addItem("Редагувати");

chooseFunction.addItem("Видалити");

chooseFunction.setRenderer(renderer);

chooseFunction.setFont(font.deriveFont(Font.BOLD, 16f));

chooseFunction.setBounds(WIDTH\_OF\_FRAME /2 - 10 - WIDTH\_OF\_FIELD, HEIGHT\_OF\_FRAME /2 - 300 , WIDTH\_OF\_FIELD, HEIGHT\_OF\_FIELD);

add(chooseFunction);

chooseFunction.addActionListener(e -> {

functionActionPerformed();

});

typeName = new JLabel("Назва групи товарів: ");

typeName.setBounds(WIDTH\_OF\_FRAME /2 - WIDTH\_OF\_FIELD /2, HEIGHT\_OF\_FRAME /2 - HEIGHT\_OF\_FIELD \*2 , WIDTH\_OF\_FIELD, HEIGHT\_OF\_FIELD);

typeName.setFont(font.deriveFont(Font.BOLD, 16f));

this.getContentPane().add(typeName);

typeDescription = new JLabel("Опис групи товарів: ");

typeDescription.setBounds(WIDTH\_OF\_FRAME /2 - WIDTH\_OF\_FIELD /2, HEIGHT\_OF\_FRAME /2 , WIDTH\_OF\_FIELD, HEIGHT\_OF\_FIELD);

typeDescription.setFont(font.deriveFont(Font.BOLD, 16f));

this.getContentPane().add(typeDescription);

typeProducer = new JLabel("Виробник: ");

typeProducer.setBounds(WIDTH\_OF\_FRAME /2 - 270, HEIGHT\_OF\_FRAME /2 - 40 , WIDTH\_OF\_FIELD, HEIGHT\_OF\_FIELD);

typeProducer.setFont(font.deriveFont(Font.BOLD, 16f));

this.getContentPane().add(typeProducer);

typeProducer.setVisible(false);

typeCount = new JLabel("Кількість на складі: ");

typeCount.setBounds(WIDTH\_OF\_FRAME /2 + WIDTH\_OF\_FIELD /2, HEIGHT\_OF\_FRAME /2 - 40 , WIDTH\_OF\_FIELD, HEIGHT\_OF\_FIELD);

typeCount.setFont(font.deriveFont(Font.BOLD, 16f));

this.getContentPane().add(typeCount);

typeCount.setVisible(false);

typePrice = new JLabel("Ціна за одиницю: ");

typePrice.setBounds(WIDTH\_OF\_FRAME /2 - 55, HEIGHT\_OF\_FRAME /2 + 60, WIDTH\_OF\_FIELD, HEIGHT\_OF\_FIELD);

typePrice.setFont(font.deriveFont(Font.BOLD, 16f));

this.getContentPane().add(typePrice);

typePrice.setVisible(false);

nameOfGroup = new JTextField();

nameOfGroup.setHorizontalAlignment(JTextField.CENTER);

nameOfGroup.setBounds(WIDTH\_OF\_FRAME /2 - WIDTH\_OF\_FIELD /2, HEIGHT\_OF\_FRAME /2 - HEIGHT\_OF\_FIELD, WIDTH\_OF\_FIELD, HEIGHT\_OF\_FIELD);

nameOfGroup.setFont(font.deriveFont(Font.BOLD, 16f));

this.getContentPane().add(nameOfGroup);

descriptionOfGroup = new JTextField();

descriptionOfGroup.setHorizontalAlignment(JTextField.CENTER);

descriptionOfGroup.setBounds(WIDTH\_OF\_FRAME /2 - WIDTH\_OF\_FIELD /2, HEIGHT\_OF\_FRAME /2 + HEIGHT\_OF\_FIELD, WIDTH\_OF\_FIELD, HEIGHT\_OF\_FIELD);

descriptionOfGroup.setFont(font.deriveFont(Font.BOLD, 16f));

this.getContentPane().add(descriptionOfGroup);

nameOfProduct = new JTextField();

nameOfProduct.setHorizontalAlignment(JTextField.CENTER);

nameOfProduct.setBounds(WIDTH\_OF\_FRAME /2 - 350, HEIGHT\_OF\_FRAME /2 - 90 , WIDTH\_OF\_FIELD, HEIGHT\_OF\_FIELD);

nameOfProduct.setFont(font.deriveFont(Font.BOLD, 16f));

this.getContentPane().add(nameOfProduct);

nameOfProduct.setVisible(false);

descriptionOfProduct = new JTextField();

descriptionOfProduct.setHorizontalAlignment(JTextField.CENTER);

descriptionOfProduct.setBounds(WIDTH\_OF\_FRAME /2 + 50, HEIGHT\_OF\_FRAME /2 - 90, WIDTH\_OF\_FIELD, HEIGHT\_OF\_FIELD);

descriptionOfProduct.setFont(font.deriveFont(Font.BOLD, 16f));

this.getContentPane().add(descriptionOfProduct);

descriptionOfProduct.setVisible(false);

producerOfProduct = new JTextField();

producerOfProduct.setHorizontalAlignment(JTextField.CENTER);

producerOfProduct.setFont(font.deriveFont(Font.BOLD, 16f));

producerOfProduct.setBounds(WIDTH\_OF\_FRAME /2 - 350, 365 , WIDTH\_OF\_FIELD, HEIGHT\_OF\_FIELD);

this.getContentPane().add(producerOfProduct);

producerOfProduct.setVisible(false);

countOfProduct = new JTextField();

countOfProduct.setHorizontalAlignment(JTextField.CENTER);

countOfProduct.setFont(font.deriveFont(Font.BOLD, 16f));

countOfProduct.setBounds(WIDTH\_OF\_FRAME /2 + 50, 365, WIDTH\_OF\_FIELD, HEIGHT\_OF\_FIELD);

this.getContentPane().add(countOfProduct);

countOfProduct.setVisible(false);

priceOfProduct = new JTextField();

priceOfProduct.setHorizontalAlignment(JTextField.CENTER);

priceOfProduct.setFont(font.deriveFont(Font.BOLD, 16f));

priceOfProduct.setBounds(WIDTH\_OF\_FRAME /2 - WIDTH\_OF\_FIELD /2, 460, WIDTH\_OF\_FIELD, HEIGHT\_OF\_FIELD);

this.getContentPane().add(priceOfProduct);

priceOfProduct.setVisible(false);

groupNameBox = new JComboBox<>();

groupNameBox.setBounds(WIDTH\_OF\_FRAME /2 - WIDTH\_OF\_FIELD /2, 130 , WIDTH\_OF\_FIELD, HEIGHT\_OF\_FIELD);

groupNameBox.setRenderer(renderer);

groupNameBox.setFont(font.deriveFont(Font.BOLD, 16f));

groupNameBox.setVisible(false);

this.getContentPane().add(groupNameBox);

groupNameBox.addActionListener(e -> {

updateProductNameBox();

});

productNameBox = new JComboBox<>();

productNameBox.setBounds(WIDTH\_OF\_FRAME /2 - WIDTH\_OF\_FIELD /2, 140 + HEIGHT\_OF\_FIELD, WIDTH\_OF\_FIELD, HEIGHT\_OF\_FIELD);

productNameBox.setRenderer(renderer);

productNameBox.setFont(font.deriveFont(Font.BOLD, 16f));

productNameBox.setVisible(false);

this.getContentPane().add(productNameBox);

saveButton = new JButton("OK");

saveButton.setBounds(WIDTH\_OF\_FRAME /2 - WIDTH\_OF\_BUTTON /2, HEIGHT\_OF\_FRAME - 150, WIDTH\_OF\_BUTTON, HEIGHT\_OF\_BUTTON);

saveButton.setBackground(new Color(102, 153, 102, 250));

saveButton.setForeground(Color.WHITE);

Font buttonFont = saveButton.getFont();

saveButton.setFont(buttonFont.deriveFont(Font.BOLD, 16f));

this.getContentPane().add(saveButton);

saveButton.addActionListener(e -> {

saveButtonActionPerformed();

});

}

private void updateProductNameBox() {

productNameBox.removeAllItems();

GroupOfItems selectedGroup;

if (groupNameBox.getSelectedItem() != null) {

selectedGroup = main.getGroupByName(groupNameBox.getSelectedItem().toString());

for (Items product : selectedGroup.productsList) {

productNameBox.addItem(product.getName());

}

}

}

private void functionActionPerformed() {

if (chooseFunction.getSelectedIndex() != 2) {

if (chooseSubject.getSelectedIndex() == 0) {

productNameBox.setBounds(WIDTH\_OF\_FRAME /2 - WIDTH\_OF\_FIELD /2, 140 + HEIGHT\_OF\_FIELD, WIDTH\_OF\_FIELD, HEIGHT\_OF\_FIELD);

groupNameBox.setBounds(WIDTH\_OF\_FRAME /2 - WIDTH\_OF\_FIELD /2, 130 , WIDTH\_OF\_FIELD, HEIGHT\_OF\_FIELD);

productNameBox.setVisible(false);

if (chooseFunction.getSelectedIndex() == 0){

groupNameBox.setVisible(false);

} else {

groupNameBox.setVisible(true);

}

nameOfProduct.setVisible(false);

descriptionOfProduct.setVisible(false);

producerOfProduct.setVisible(false);

countOfProduct.setVisible(false);

priceOfProduct.setVisible(false);

typeProducer.setVisible(false);

typeCount.setVisible(false);

typePrice.setVisible(false);

typeDescription.setBounds(WIDTH\_OF\_FRAME /2 - WIDTH\_OF\_FIELD /2, HEIGHT\_OF\_FRAME /2 , WIDTH\_OF\_FIELD, HEIGHT\_OF\_FIELD);

typeName.setText("Назва групи товарів:");

typeName.setBounds(WIDTH\_OF\_FRAME /2 - WIDTH\_OF\_FIELD /2, HEIGHT\_OF\_FRAME /2 - HEIGHT\_OF\_FIELD \*2 , WIDTH\_OF\_FIELD, HEIGHT\_OF\_FIELD);

descriptionOfGroup.setVisible(true);

typeDescription.setVisible(true);

nameOfGroup.setVisible(true);

typeName.setVisible(true);

} else {

nameOfGroup.setVisible(false);

descriptionOfGroup.setVisible(false);

if(chooseFunction.getSelectedIndex() == 1){

groupNameBox.setBounds(WIDTH\_OF\_FRAME /2 - 10 - WIDTH\_OF\_FIELD, 130 , WIDTH\_OF\_FIELD, HEIGHT\_OF\_FIELD);

productNameBox.setBounds(WIDTH\_OF\_FRAME /2 + 10, 130 , WIDTH\_OF\_FIELD, HEIGHT\_OF\_FIELD);

productNameBox.setVisible(true);

} else {

productNameBox.setVisible(false);

productNameBox.setBounds(WIDTH\_OF\_FRAME /2 - WIDTH\_OF\_FIELD /2, 140 + HEIGHT\_OF\_FIELD, WIDTH\_OF\_FIELD, HEIGHT\_OF\_FIELD);

groupNameBox.setBounds(WIDTH\_OF\_FRAME /2 - WIDTH\_OF\_FIELD /2, 130 , WIDTH\_OF\_FIELD, HEIGHT\_OF\_FIELD);

}

typeName.setText("Назва товару:");

typeName.setBounds(WIDTH\_OF\_FRAME /2 - 270, HEIGHT\_OF\_FRAME /2 - HEIGHT\_OF\_FIELD \*2 , WIDTH\_OF\_FIELD, HEIGHT\_OF\_FIELD);

typeDescription.setVisible(true);

typeDescription.setText("Опис товару:");

typeDescription.setBounds(WIDTH\_OF\_FRAME /2 + WIDTH\_OF\_FIELD /2, HEIGHT\_OF\_FRAME /2 - HEIGHT\_OF\_FIELD \*2 , WIDTH\_OF\_FIELD, HEIGHT\_OF\_FIELD);

groupNameBox.setVisible(true);

nameOfProduct.setVisible(true);

descriptionOfProduct.setVisible(true);

producerOfProduct.setVisible(true);

countOfProduct.setVisible(true);

priceOfProduct.setVisible(true);

typeProducer.setVisible(true);

typeCount.setVisible(true);

typePrice.setVisible(true);

typeName.setVisible(true);

}

} else {

productNameBox.setBounds(WIDTH\_OF\_FRAME /2 - WIDTH\_OF\_FIELD /2, 140 + HEIGHT\_OF\_FIELD, WIDTH\_OF\_FIELD, HEIGHT\_OF\_FIELD);

groupNameBox.setBounds(WIDTH\_OF\_FRAME /2 - WIDTH\_OF\_FIELD /2, 130 , WIDTH\_OF\_FIELD, HEIGHT\_OF\_FIELD);

if (chooseSubject.getSelectedIndex() == 0) {

productNameBox.setVisible(false);

nameOfProduct.setVisible(false);

descriptionOfProduct.setVisible(false);

producerOfProduct.setVisible(false);

countOfProduct.setVisible(false);

priceOfProduct.setVisible(false);

descriptionOfGroup.setVisible(false);

nameOfGroup.setVisible(false);

typeProducer.setVisible(false);

typeCount.setVisible(false);

typePrice.setVisible(false);

typeDescription.setVisible(false);

typeName.setVisible(false);

groupNameBox.setVisible(true);

} else {

nameOfProduct.setVisible(false);

descriptionOfProduct.setVisible(false);

producerOfProduct.setVisible(false);

countOfProduct.setVisible(false);

priceOfProduct.setVisible(false);

descriptionOfGroup.setVisible(false);

typeProducer.setVisible(false);

typeCount.setVisible(false);

typePrice.setVisible(false);

typeDescription.setVisible(false);

typeName.setVisible(false);

groupNameBox.setVisible(true);

productNameBox.setVisible(true);

}

}

}

private void saveButtonActionPerformed() {

if (chooseFunction.getSelectedIndex() == 0){

if(chooseSubject.getSelectedIndex() == 0){

addGroup();

} else {

addProduct();

}

} else if (chooseFunction.getSelectedIndex() == 1){

if(chooseSubject.getSelectedIndex() == 0){

editGroup();

} else {

editProduct();

}

} else {

if(chooseSubject.getSelectedIndex() == 0){

deleteGroup();

} else {

deleteProduct();

}

}

nameOfGroup.setText("");

descriptionOfGroup.setText("");

nameOfProduct.setText("");

descriptionOfProduct.setText("");

producerOfProduct.setText("");

countOfProduct.setText("");

priceOfProduct.setText("");

}

private void deleteProduct() {

String groupName;

if(groupNameBox.getSelectedItem() != null){

groupName = groupNameBox.getSelectedItem().toString();

} else {

JOptionPane.showMessageDialog(this, "Групу товарів не обрано!", "Error", JOptionPane.ERROR\_MESSAGE);

return;

}

String productName;

if(productNameBox.getSelectedItem() != null){

productName = productNameBox.getSelectedItem().toString();

} else {

JOptionPane.showMessageDialog(this, "Товар не обрано!", "Error", JOptionPane.ERROR\_MESSAGE);

return;

}

try {

main.getGroupByName(groupName).removeProduct(productName);

main.updateFiles();

} catch (IllegalArgumentException e) {

e.printStackTrace();

JOptionPane.showMessageDialog(this, e.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);

return;

}

updateProductNameBox();

JOptionPane.showMessageDialog(this, "Товар \"" + productName + "\" успішно видалено з групи \"" + groupName + "\"!");

}

private void deleteGroup() {

String groupName;

if(groupNameBox.getSelectedItem() != null){

groupName = groupNameBox.getSelectedItem().toString();

} else {

JOptionPane.showMessageDialog(this, "Групу товарів не обрано!", "Error", JOptionPane.ERROR\_MESSAGE);

return;

}

try {

main.removeGroup(groupName);

main.updateFiles();

} catch (IllegalArgumentException e) {

e.printStackTrace();

JOptionPane.showMessageDialog(this, e.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);

return;

}

updateInformation();

JOptionPane.showMessageDialog(this, "Групу товарів \"" + groupName + "\" успішно видалено!");

}

private void editProduct() {

String groupName;

if(groupNameBox.getSelectedItem() != null){

groupName = groupNameBox.getSelectedItem().toString();

} else {

JOptionPane.showMessageDialog(this, "Групу товарів не обрано!", "Error", JOptionPane.ERROR\_MESSAGE);

return;

}

String productName;

if(productNameBox.getSelectedItem() != null){

productName = productNameBox.getSelectedItem().toString();

} else {

JOptionPane.showMessageDialog(this, "Товар не обрано!", "Error", JOptionPane.ERROR\_MESSAGE);

return;

}

String newName = nameOfProduct.getText();

newName = newName.replaceAll(";", "");

String newDescription = descriptionOfProduct.getText();

newDescription = newDescription.replaceAll(";", "");

String newProducer = producerOfProduct.getText();

newProducer = newProducer.replaceAll(";", "");

String newCount = countOfProduct.getText();

String newPrice = priceOfProduct.getText();

try {

main.getGroupByName(groupName).editProduct(productName, newName, newDescription, newProducer, newCount, newPrice);

main.updateFiles();

} catch (IllegalArgumentException e) {

e.printStackTrace();

JOptionPane.showMessageDialog(this, e.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);

return;

}

updateProductNameBox();

JOptionPane.showMessageDialog(this, "Товар \"" + productName + "\" успішно відредаговано!");

}

private void editGroup() {

String oldName;

if(groupNameBox.getSelectedItem() != null){

oldName = groupNameBox.getSelectedItem().toString();

} else {

JOptionPane.showMessageDialog(this, "Групу товарів не обрано!", "Error", JOptionPane.ERROR\_MESSAGE);

return;

}

String newName = nameOfGroup.getText();

newName = newName.replaceAll(";", "");

String newDescription = descriptionOfGroup.getText();

newDescription = newDescription.replaceAll(";", "");

try {

main.editGroup(oldName, newName, newDescription);

main.updateFiles();

} catch (IllegalArgumentException e) {

e.printStackTrace();

JOptionPane.showMessageDialog(this, e.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);

return;

}

updateInformation();

JOptionPane.showMessageDialog(this, "Групу товарів \"" + oldName + "\" успішно відредаговано!");

}

private void addProduct() {

String groupName;

if(groupNameBox.getSelectedItem() != null){

groupName = groupNameBox.getSelectedItem().toString();

} else {

JOptionPane.showMessageDialog(this, "Групу товарів не обрано!", "Error", JOptionPane.ERROR\_MESSAGE);

return;

}

String name = nameOfProduct.getText();

name = name.replaceAll(";", "");

String description = descriptionOfProduct.getText();

description = description.replaceAll(";", "");

String producer = producerOfProduct.getText();

producer = producer.replaceAll(";", "");

int count = 0;

int price = 0;

try {

count = parseInt(countOfProduct.getText());

price = parseInt(priceOfProduct.getText());

} catch (NumberFormatException e) {

JOptionPane.showMessageDialog(this, "Кількість та ціна повинні бути числами!", "Error", JOptionPane.ERROR\_MESSAGE);

return;

}

try {

main.getGroupByName(groupName).addProduct(name, description, producer, count, price);

main.updateFiles();

} catch (IllegalArgumentException e) {

e.printStackTrace();

JOptionPane.showMessageDialog(this, e.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);

return;

}

updateInformation();

JOptionPane.showMessageDialog(this, "Товар \"" + name + "\" успішно додано в групу \"" + groupName + "\"!");

}

private void addGroup() {

String groupName = nameOfGroup.getText();

groupName = groupName.replaceAll(";", "");

String groupDescription = descriptionOfGroup.getText();

groupName = groupName.replaceAll(";", "");

try (BufferedWriter writer = new BufferedWriter(new FileWriter(main.groupsFileName, true))) {

main.addGroup(new GroupOfItems(groupName, groupDescription, main));

writer.write(groupName + ";" + groupDescription + "\n");

} catch (IllegalArgumentException e) {

e.printStackTrace();

JOptionPane.showMessageDialog(this, e.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);

return;

} catch (IOException e) {

e.printStackTrace();

JOptionPane.showMessageDialog(this, "Помилка при додаванні групи товарів у файл.", "Error", JOptionPane.ERROR\_MESSAGE);

return;

}

updateInformation();

JOptionPane.showMessageDialog(this, "Групу товарів \"" + groupName + "\" успішно додано!");

}

private void addProductToGroup(String groupName, String name, String description, String producer, int count, int pricePerOne) {

}

private void updateInformation() {

if (isUpdating) {

return;

}

isUpdating = true;

System.out.println("Updating information");

groupNameBox.removeAllItems();

for (GroupOfItems group : main.groupsList) {

groupNameBox.addItem(group.getNameOfGroup());

}

isUpdating = false;

}

}

import javax.swing.\*;

import java.awt.\*;

/\*\*

\* Вікно головного меню

\*/

public class **Interface** extends JFrame{

private final Main main;

private JComboBox<String> choose;

private JButton okButton;

private static final int WIDTH\_OF\_FRAME = 700;

private static final int HEIGHT\_OF\_FRAME = 500;

private static final int WIDTH\_OF\_FIELD = 450;

private static final int HEIGHT\_OF\_FIELD = 70;

private static final int WIDTH\_OF\_BUTTON = 170;

private static final int HEIGHT\_OF\_BUTTON = 70;

public Interface(String name, Main main) {

this.main = main;

this.setTitle(name);

this.setSize(WIDTH\_OF\_FRAME, HEIGHT\_OF\_FRAME);

this.getContentPane().setLayout(null);

this.getContentPane().setBackground(new Color(204, 255, 153, 250));

this.setLocationRelativeTo(null);

this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

}

public void start(){

init();

this.setVisible(true);

}

private void init() {

if (choose == null){

JLabel label = new JLabel("Автоматизоване робоче місце");

label.setBounds(WIDTH\_OF\_FRAME / 2 - WIDTH\_OF\_FIELD /2, 35, 480, HEIGHT\_OF\_FIELD);

Font font = label.getFont();

label.setFont(font.deriveFont(Font.BOLD, 30f));

add(label);

choose = new JComboBox<>();

choose.addItem("Додати/Редагувати/Видалити товар/групу товарів");

choose.addItem("Постачання/списання товару");

choose.addItem("Пошук товару");

choose.addItem("Статистика");

DefaultListCellRenderer renderer = new DefaultListCellRenderer();

renderer.setHorizontalAlignment(SwingConstants.CENTER);

choose.setRenderer(renderer);

choose.setFont(font.deriveFont(Font.BOLD, 16f));

choose.setBounds(WIDTH\_OF\_FRAME /2 - WIDTH\_OF\_FIELD /2, HEIGHT\_OF\_FRAME /2 - HEIGHT\_OF\_FIELD\*2, WIDTH\_OF\_FIELD, HEIGHT\_OF\_FIELD);

add(choose);

okButton = new JButton("OK");

okButton.setBounds(WIDTH\_OF\_FRAME /2 - WIDTH\_OF\_BUTTON /2, (int) (HEIGHT\_OF\_FRAME\*0.8 - HEIGHT\_OF\_BUTTON), WIDTH\_OF\_BUTTON, HEIGHT\_OF\_BUTTON);

okButton.setBackground(new Color(102, 153, 102, 250));

okButton.setForeground(Color.WHITE);

Font buttonFont = okButton.getFont();

okButton.setFont(buttonFont.deriveFont(Font.BOLD, 16f));

add(okButton);

okButton.addActionListener(e -> {

okButtonActionPerformed();

});

}

}

private void okButtonActionPerformed() {

switch (choose.getSelectedIndex()){

case 0:

main.editProducts.start();

break;

case 1:

main.supplyOfProducts.start();

break;

case 2:

main.productSearch.start();

break;

case 3:

main.statistics.start();

break;

}

}

}

ЗВІТ

Про виконання лабораторної роботи № 2

ПРОГРАМУВАННЯ ЦИКЛІЧНИХ ПРОЦЕСІВ В С

з дисципліни: Основи програмува

import javax.swing.\*;

import java.awt.\*;

public class **ProductSearch** extends JDialog {

private Main main;

/\*\* Ввести назву товару \*/

private JTextField nameOfProduct;

/\*\* Кнопка пошуку \*/

private JButton searchButton;

/\*\* Вивести інформацію про товар \*/

private JTextArea information;

private static final int WIDTH\_OF\_FRAME = 500;

private static final int HEIGHT\_OF\_FRAME = 700;

private static final int WIDTH\_OF\_FIELD = 300;

private static final int HEIGHT\_OF\_FIELD = 70;

private static final int WIDTH\_OF\_BUTTON = 170;

private static final int HEIGHT\_OF\_BUTTON = 70;

public ProductSearch(String name, Main main, JFrame parent) {

super(parent, name, true);

this.main = main;

this.setTitle(name);

this.setSize(WIDTH\_OF\_FRAME, HEIGHT\_OF\_FRAME);

this.getContentPane().setLayout(null);

this.getContentPane().setBackground(new Color(204, 255, 153, 250));

this.setLocationRelativeTo(null);

}

public void start(){

init();

this.setVisible(true);

}

private void init() {

if (nameOfProduct != null){

return;

}

nameOfProduct = new JTextField();

nameOfProduct.setHorizontalAlignment(JTextField.CENTER);

Font font = nameOfProduct.getFont();

nameOfProduct.setFont(font.deriveFont(Font.BOLD, 16f));

nameOfProduct.setBounds(WIDTH\_OF\_FRAME / 2 - WIDTH\_OF\_FIELD / 2, 100, WIDTH\_OF\_FIELD, HEIGHT\_OF\_FIELD);

add(nameOfProduct);

JLabel label1 = new JLabel("Пошук товару");

label1.setBounds(WIDTH\_OF\_FRAME / 2 - 50, 10, WIDTH\_OF\_FIELD, HEIGHT\_OF\_FIELD);

label1.setFont(font.deriveFont(Font.BOLD, 16f));

add(label1);

JLabel label2 = new JLabel("\* - будь-яка к-ть будь-яких символів");

label2.setBounds(WIDTH\_OF\_FRAME / 2 - WIDTH\_OF\_FIELD / 2, 30, WIDTH\_OF\_FIELD, HEIGHT\_OF\_FIELD);

label2.setFont(font.deriveFont(Font.BOLD, 16f));

add(label2);

JLabel label3 = new JLabel("? - один будь-який символ");

label3.setBounds(WIDTH\_OF\_FRAME / 2 - WIDTH\_OF\_FIELD / 2, 50, WIDTH\_OF\_FIELD, HEIGHT\_OF\_FIELD);

label3.setFont(font.deriveFont(Font.BOLD, 16f));

add(label3);

searchButton = new JButton("Знайти товар");

searchButton.setBounds(WIDTH\_OF\_FRAME / 2 - WIDTH\_OF\_BUTTON / 2, 200, WIDTH\_OF\_BUTTON, HEIGHT\_OF\_BUTTON);

searchButton.setBackground(new Color(102, 153, 102, 250));

searchButton.setForeground(Color.WHITE);

Font buttonFont = searchButton.getFont();

searchButton.setFont(buttonFont.deriveFont(Font.BOLD, 16f));

add(searchButton);

searchButton.addActionListener(e -> searchProduct());

information = new JTextArea(10, 30);

information.setEditable(false);

information.setFont(new Font("Arial", Font.PLAIN, 19));

JScrollPane scrollPane = new JScrollPane(information);

scrollPane.setBounds(WIDTH\_OF\_FRAME / 2 - 200, 300, 400, 300);

add(scrollPane);

}

private void searchProduct() {

information.setText("");

String searchRegex = nameOfProduct.getText();

if (searchRegex.isEmpty()){

JOptionPane.showMessageDialog(this, "Введіть вираз для пошуку.", "Помилка", JOptionPane.ERROR\_MESSAGE);

return;

}

searchRegex = searchRegex.replaceAll("[\*]+", ".\*");

searchRegex = searchRegex.replaceAll("[?]", ".");

boolean isFound = false;

for (GroupOfItems group : main.getGroupsList()){

for (Items product : group.productsList){

if (product.getName().matches(searchRegex)){

information.append(product.getName() + "\n");

information.append(" Група товару: " + group.getNameOfGroup() + "\n");

information.append(" Ціна товару: " + product.getPricePerOne() + "\n");

information.append(" Кількість на складі: " + product.getCount() + "\n");

information.append(" Виробник: " + product.getProducer() + "\n");

information.append(" Опис: " + product.getDescription() + "\n");

information.append("--------------------------------------------------------------------------------\n");

isFound = true;

}

}

}

if (!isFound){

JOptionPane.showMessageDialog(this, "Товар не знайдено.");

}

}

}

import javax.swing.\*;

import java.awt.\*;

public class **Statistics** extends JDialog {

private final Main main;

private JLabel label;

private JComboBox<String> statFuncBox;

private JComboBox<String> groupNameBox;

private JTextArea statisticsArea;

private JScrollPane scrollPane;

private static final int WIDTH\_OF\_FRAME = 800;

private static final int HEIGHT\_OF\_FRAME = 700;

private static final int WIDTH\_OF\_FIELD = 300;

private static final int HEIGHT\_OF\_FIELD = 70;

private static final int WIDTH\_OF\_AREA = 610;

private static final int HEIGHT\_OF\_AREA = 400;

private volatile boolean isPrinting = false;

public Statistics(String name, Main main, JFrame parent) {

super(parent, name, true);

this.main = main;

this.setTitle(name);

this.setSize(WIDTH\_OF\_FRAME, HEIGHT\_OF\_FRAME);

this.getContentPane().setLayout(null);

this.getContentPane().setBackground(new Color(204, 255, 153, 250));

this.setLocationRelativeTo(null);

}

public void start(){

init();

this.setVisible(true);

}

private void init() {

if (label != null){

return;

}

label = new JLabel("Статистика");

label.setBounds(WIDTH\_OF\_FRAME / 2 - 50, 40, WIDTH\_OF\_FIELD, HEIGHT\_OF\_FIELD);

Font font = label.getFont();

label.setFont(font.deriveFont(Font.BOLD, 16f));

add(label);

statisticsArea = new JTextArea();

statisticsArea.setFont(font.deriveFont(Font.BOLD, 16f));

statisticsArea.setEditable(false);

scrollPane = new JScrollPane(statisticsArea);

scrollPane.setBounds(WIDTH\_OF\_FRAME / 2 - WIDTH\_OF\_AREA / 2, 200, WIDTH\_OF\_AREA, HEIGHT\_OF\_AREA);

add(scrollPane);

statFuncBox = new JComboBox<>();

statFuncBox.setBounds(WIDTH\_OF\_FRAME /2 - WIDTH\_OF\_FIELD /2, 120 , WIDTH\_OF\_FIELD, HEIGHT\_OF\_FIELD);

DefaultListCellRenderer renderer = new DefaultListCellRenderer();

renderer.setHorizontalAlignment(SwingConstants.CENTER);

statFuncBox.setRenderer(renderer);

statFuncBox.setFont(font.deriveFont(Font.BOLD, 16f));

statFuncBox.addItem("Загальна статистика");

statFuncBox.addItem("Статистика за групою товарів");

add(statFuncBox);

statFuncBox.addActionListener(e -> {

updateGroupList();

printStatistics();

});

groupNameBox = new JComboBox<>();

groupNameBox.setBounds(WIDTH\_OF\_FRAME /2 + 5, 120, WIDTH\_OF\_FIELD, HEIGHT\_OF\_FIELD);

groupNameBox.setRenderer(renderer);

groupNameBox.setFont(font.deriveFont(Font.BOLD, 16f));

groupNameBox.setVisible(false);

this.getContentPane().add(groupNameBox);

groupNameBox.addActionListener(e -> {

printStatistics();

});

}

private void updateGroupList() {

groupNameBox.removeAllItems();

for (GroupOfItems group : main.getGroupsList()){

groupNameBox.addItem(group.getNameOfGroup());

}

}

private void printStatistics() {

if (isPrinting){

return;

}

isPrinting = true;

if(statFuncBox.getSelectedIndex() == 1){

statFuncBox.setBounds(WIDTH\_OF\_FRAME /2 - 5 - WIDTH\_OF\_FIELD, 120, WIDTH\_OF\_FIELD, HEIGHT\_OF\_FIELD);

groupNameBox.setVisible(true);

} else {

statFuncBox.setBounds(WIDTH\_OF\_FRAME /2 - WIDTH\_OF\_FIELD /2, 120, WIDTH\_OF\_FIELD, HEIGHT\_OF\_FIELD);

groupNameBox.setVisible(false);

}

statisticsArea.setText("");

if (statFuncBox.getSelectedIndex() == 0){

statisticsArea.append("Загальна вартість товарів на складі: " + main.getTotalStockValue() + "\n");

statisticsArea.append("Ім'я товару - Кількість - Ціна за одиницю - Виробник - Опис\n");

statisticsArea.append("----------------------------------------------------------------------------------------------------------------\n");

for (GroupOfItems group : main.getGroupsList()){

for (Items product : group.productsList){

statisticsArea.append(product.getName() + " - " + product.getCount() + " шт. - " + product.getPricePerOne() + " грн. - " + product.getProducer() + " - " + product.getDescription() + "\n");

}

}

} else {

if(groupNameBox.getSelectedItem() != null){

GroupOfItems group = main.getGroupByName(groupNameBox.getSelectedItem().toString());

statisticsArea.append("Загальна вартість товарів групи " + group.getNameOfGroup() + " на складі: " + group.getTotalValue() + "\n");

statisticsArea.append("Ім'я товару - Кількість - Ціна за одиницю - Виробник - Опис\n");

statisticsArea.append("----------------------------------------------------------------------------------------------------------------\n");

for (Items product : group.productsList){

statisticsArea.append(product.getName() + " - " + product.getCount() + " шт. - " + product.getPricePerOne() + " грн. - " + product.getProducer() + " - " + product.getDescription() + "\n");

}

}

}

isPrinting = false;

}

}

import javax.swing.\*;

import java.awt.\*;

public class **SupplyOfProducts** extends JDialog {

private final Main main;

/\*\* Вибір функції: списати товар/поставка товарів \*/

private JComboBox<String> chooseFunction;

/\*\* Вибір товару \*/

private JComboBox<String> chooseProduct;

/\*\* Вибір групи товарів \*/

private JComboBox<String> chooseGroup;

/\*\* Скільки поставили/списали \*/

private JTextField supplyCount;

/\*\* Напис "Скільки списати"/"Скільки поставили" \*/

private JLabel supplyCountLabel;

/\*\* Напис "Поточна кількість товару" \*/

private JLabel currentQuantityLabel;

private static final int WIDTH\_OF\_FRAME = 500;

private static final int HEIGHT\_OF\_FRAME = 700;

private static final int WIDTH\_OF\_FIELD = 300;

private static final int HEIGHT\_OF\_FIELD = 70;

private static final int WIDTH\_OF\_BUTTON = 170;

private static final int HEIGHT\_OF\_BUTTON = 70;

private volatile boolean isUpdating = false;

public SupplyOfProducts(String name, Main main, JFrame parent) {

super(parent, name, true);

this.main = main;

this.setTitle(name);

this.setSize(WIDTH\_OF\_FRAME, HEIGHT\_OF\_FRAME);

this.getContentPane().setLayout(null);

this.getContentPane().setBackground(new Color(204, 255, 153, 250));

this.setLocationRelativeTo(null);

}

public void start(){

init();

updateInformation();

this.setVisible(true);

}

private void init() {

if (chooseFunction != null){

return;

}

chooseFunction = new JComboBox<>();

chooseFunction.addItem("Списати товар");

chooseFunction.addItem("Доставити товар");

DefaultListCellRenderer renderer = new DefaultListCellRenderer();

renderer.setHorizontalAlignment(SwingConstants.CENTER);

chooseFunction.setRenderer(renderer);

Font font = chooseFunction.getFont();

chooseFunction.setFont(font.deriveFont(Font.BOLD, 16f));

chooseFunction.setBounds(WIDTH\_OF\_FRAME / 2 - WIDTH\_OF\_FIELD / 2, HEIGHT\_OF\_FRAME / 2 - 300, WIDTH\_OF\_FIELD, HEIGHT\_OF\_FIELD);

add(chooseFunction);

chooseGroup = new JComboBox<>();

chooseGroup.setBounds(WIDTH\_OF\_FRAME / 2 - WIDTH\_OF\_FIELD / 2, HEIGHT\_OF\_FRAME / 2 - 200, WIDTH\_OF\_FIELD, HEIGHT\_OF\_FIELD);

chooseGroup.setRenderer(renderer);

chooseGroup.setFont(font.deriveFont(Font.BOLD, 16f));

add(chooseGroup);

chooseProduct = new JComboBox<>();

chooseProduct.setBounds(WIDTH\_OF\_FRAME / 2 - WIDTH\_OF\_FIELD / 2, HEIGHT\_OF\_FRAME / 2 - 100, WIDTH\_OF\_FIELD, HEIGHT\_OF\_FIELD);

chooseProduct.setRenderer(renderer);

chooseProduct.setFont(font.deriveFont(Font.BOLD, 16f));

add(chooseProduct);

supplyCountLabel = new JLabel("Скільки списати:");

supplyCountLabel.setBounds(WIDTH\_OF\_FRAME / 2 - WIDTH\_OF\_FIELD / 2, HEIGHT\_OF\_FRAME - 320, WIDTH\_OF\_FIELD, HEIGHT\_OF\_FIELD);

supplyCountLabel.setFont(font.deriveFont(Font.BOLD, 16f));

add(supplyCountLabel);

currentQuantityLabel = new JLabel("Поточна кількість товару: ");

currentQuantityLabel.setBounds(WIDTH\_OF\_FRAME / 2 - WIDTH\_OF\_FIELD / 2, HEIGHT\_OF\_FRAME - 360, WIDTH\_OF\_FIELD, HEIGHT\_OF\_FIELD);

currentQuantityLabel.setFont(font.deriveFont(Font.BOLD, 16f));

add(currentQuantityLabel);

supplyCount = new JTextField();

supplyCount.setHorizontalAlignment(JTextField.CENTER);

supplyCount.setFont(font.deriveFont(Font.BOLD, 16f));

supplyCount.setBounds(WIDTH\_OF\_FRAME / 2 - WIDTH\_OF\_FIELD / 2, 450, WIDTH\_OF\_FIELD, HEIGHT\_OF\_FIELD);

add(supplyCount);

JButton saveButton = new JButton("OK");

saveButton.setBounds(WIDTH\_OF\_FRAME / 2 - WIDTH\_OF\_BUTTON / 2, HEIGHT\_OF\_FRAME - 150, WIDTH\_OF\_BUTTON, HEIGHT\_OF\_BUTTON);

saveButton.setBackground(new Color(102, 153, 102, 250));

saveButton.setForeground(Color.WHITE);

Font buttonFont = saveButton.getFont();

saveButton.setFont(buttonFont.deriveFont(Font.BOLD, 16f));

add(saveButton);

saveButton.addActionListener(e -> {

saveButtonActionPerformed();

});

chooseFunction.addActionListener(e -> {

if (chooseFunction.getSelectedIndex() == 0) {

supplyCountLabel.setText("Скільки списати:");

} else {

supplyCountLabel.setText("Скільки доставити:");

}

});

chooseGroup.addActionListener(e -> {

updateProductNameBox();

});

chooseProduct.addActionListener(e -> {

showQuantityOfProduct();

});

}

private void showQuantityOfProduct() {

String selectedGroup;

String selectedProduct;

if (chooseGroup.getSelectedItem() != null && chooseProduct.getSelectedItem() != null) {

selectedGroup = chooseGroup.getSelectedItem().toString();

selectedProduct = chooseProduct.getSelectedItem().toString();

} else {

return;

}

currentQuantityLabel.setText("Поточна кількість товару: " + main.getGroupByName(selectedGroup).getProductByName(selectedProduct).getCount());

System.out.println("showQuantityOfProduct");

}

private void saveButtonActionPerformed() {

String selectedGroup;

String selectedProduct;

if (chooseGroup.getSelectedItem() != null) {

selectedGroup = chooseGroup.getSelectedItem().toString();

} else {

JOptionPane.showMessageDialog(this, "Виберіть групу товару.", "Помилка", JOptionPane.ERROR\_MESSAGE);

return;

}

if (chooseProduct.getSelectedItem() != null) {

selectedProduct = chooseProduct.getSelectedItem().toString();

} else {

JOptionPane.showMessageDialog(this, "Виберіть товар.", "Помилка", JOptionPane.ERROR\_MESSAGE);

return;

}

try {

if (chooseFunction.getSelectedIndex() == 0) {

main.getGroupByName(selectedGroup).subtractProduct(selectedProduct, supplyCount.getText());

main.updateFiles();

JOptionPane.showMessageDialog(this, "Товар успішно списано.");

showQuantityOfProduct();

} else {

main.getGroupByName(selectedGroup).supplyProduct(selectedProduct, supplyCount.getText());

main.updateFiles();

JOptionPane.showMessageDialog(this, "Товар успішно доставлено.");

showQuantityOfProduct();

}

} catch (IllegalArgumentException e) {

e.printStackTrace();

JOptionPane.showMessageDialog(this, e.getMessage(), "Помилка", JOptionPane.ERROR\_MESSAGE);

return;

} finally {

supplyCount.setText("");

}

}

private void updateInformation() {

if (isUpdating) {

return;

}

isUpdating = true;

System.out.println("Updating information");

chooseGroup.removeAllItems();

for (GroupOfItems group : main.groupsList) {

chooseGroup.addItem(group.getNameOfGroup());

}

isUpdating = false;

}

private void updateProductNameBox() {

chooseProduct.removeAllItems();

GroupOfItems selectedGroup;

if (chooseGroup.getSelectedItem() != null) {

selectedGroup = main.getGroupByName(chooseGroup.getSelectedItem().toString());

for (Items product : selectedGroup.productsList) {

chooseProduct.addItem(product.getName());

}

}

}

}

import javax.swing.\*;

import java.io.\*;

import java.util.ArrayList;

/\*\*

\* Лабораторна робота №2

\* Автоматизоване робоче місце

\*

\* @author Гапоненко Єлизавета Артемівна

\* @author Розумєй Максим Віталійович

\*/

public class **Main** {

/\*\*Масив для зберігання груп товарів \*/

public ArrayList<GroupOfItems> groupsList = new ArrayList<>();

//вікна програми

Interface mainInterface;

ProductSearch productSearch;

EditProducts editProducts;

SupplyOfProducts supplyOfProducts;

Statistics statistics;

public final String groupsFileName; //файл для збереження груп товарів

public final String allProductsFileName; //файл для збереження груп товарів

/\*\*

\* Точка входу в програму

\* @param args

\*/

public static void main(String[] args) {

Main main = new Main("GroupsOfProducts.txt", "AllProducts.txt");

main.start();

}

/\*\*

\* Конструктор класу Main

\* @param groupsFileName - назва файлу для збереження груп товарів

\* @param allProductsFileName - назва файлу для збереження усіх товарів

\*/

public Main(String groupsFileName, String allProductsFileName) {

this.groupsFileName = groupsFileName;

this.allProductsFileName = allProductsFileName;

}

/\*\*

\* Метод для запуску програми

\*/

private void start() {

loadInformationFromFiles();

for (GroupOfItems group : groupsList){

System.out.println(group.getNameOfGroup()+"-----------");

for (Items product : group.productsList){

System.out.println(product.getName());

}

}

mainInterface = new Interface("Stock", this);

editProducts = new EditProducts("Додавання товару", this, mainInterface);

productSearch = new ProductSearch("Пошук товару", this, mainInterface);

supplyOfProducts = new SupplyOfProducts("Поставка товару", this, mainInterface);

statistics = new Statistics("Статистика", this, mainInterface);

mainInterface.start();

}

/\*\*

\* Метод для завантаження інформації з файлів

\*/

private void loadInformationFromFiles() {

try(BufferedReader reader = new BufferedReader(new FileReader(groupsFileName))){

String line;

while ((line = reader.readLine()) != null){

groupsList.add(new GroupOfItems(line.split(";")[0], line.split(";")[1], this));

}

} catch (Exception e) {

e.printStackTrace();

JOptionPane.showMessageDialog(null, "Помилка при зчитуванні файлу груп товарів.", "Помилка", JOptionPane.ERROR\_MESSAGE);

System.exit(1);

}

try(BufferedReader reader = new BufferedReader(new FileReader(allProductsFileName))){

String line;

while ((line = reader.readLine()) != null){

String[] productData = line.split(";");

for (GroupOfItems group : groupsList){

if (group.getNameOfGroup().equals(productData[0])){

group.addProduct(productData[1], productData[2],productData[3], Integer.parseInt(productData[4]), Integer.parseInt(productData[5]));

break;

}

}

}

} catch (Exception e) {

e.printStackTrace();

JOptionPane.showMessageDialog(null, "Помилка при зчитуванні файлу усіх товарів.", "Помилка", JOptionPane.ERROR\_MESSAGE);

System.exit(1);

}

}

/\*\*

\* Метод для отримання списку груп товарів

\* @return - список груп товарів

\*/

public ArrayList<GroupOfItems> getGroupsList() {

return groupsList;

}

/\*\*

\* Метод для отримання групи товарів за назвою

\* @param name - назва групи товарів

\* @return - група товарів

\*/

public GroupOfItems getGroupByName(String name){

for (GroupOfItems group : groupsList){

if (group.getNameOfGroup().equals(name)){

return group;

}

}

return null;

}

/\*\*

\* Метод для перевірки існування групи товарів за назвою

\* @param name - назва групи товарів

\* @return - true, якщо група існує, false - якщо група не існує

\*/

public boolean groupExists(String name){

for (GroupOfItems group : groupsList){

if (group.getNameOfGroup().equals(name)){

return true;

}

}

return false;

}

/\*\*

\* Метод для перевірки існування товару за назвою

\* @param name - назва товару

\* @return - true, якщо товар існує, false - якщо товар не існує

\*/

public boolean productExists(String name){

for (GroupOfItems group : groupsList){

for (Items product : group.productsList){

if (product.getName().equals(name)){

return true;

}

}

}

return false;

}

/\*\*

\* Метод для додавання товару

\* @param group - група товару

\* @throws IllegalArgumentException неправильні аргументи

\*/

public void addGroup(GroupOfItems group) throws IllegalArgumentException{

if (groupExists(group.getNameOfGroup())){

throw new IllegalArgumentException("Група з такою назвою вже існує.");

} else if (group.getNameOfGroup().trim().isEmpty()){

throw new IllegalArgumentException("Назва групи не може бути порожньою.");

} else if (group.getGroupDescription().trim().isEmpty()){

throw new IllegalArgumentException("Опис групи не може бути порожнім.");

}

groupsList.add(group);

}

/\*\*

\* Метод для редагування групи товарів

\* @param oldName - стара назва групи

\* @param newName - нова назва групи

\* @param newDescription - новий опис групи

\* @throws IllegalArgumentException

\*/

public void editGroup(String oldName, String newName, String newDescription) throws IllegalArgumentException{

if (!groupExists(oldName)){

throw new IllegalArgumentException("Групи з такою назвою не існує.");

} else if (groupExists(newName) && newDescription.trim().isEmpty()){

throw new IllegalArgumentException("Група має таку саму назву");

} else if (newName.trim().isEmpty() && newDescription.trim().isEmpty()){

throw new IllegalArgumentException("Нова назва або опис повинні бути заповнені.");

} else if (newName.equals(oldName) && newDescription.equals(getGroupByName(oldName).getGroupDescription())){

throw new IllegalArgumentException("Нова назва або опис повинні бути відмінні від старих.");

} else if (newName.trim().isEmpty() && getGroupByName(oldName).getGroupDescription().equals(newDescription)){

throw new IllegalArgumentException("Група має такий самий опис.");

}

for (GroupOfItems group : groupsList) {

if (group.getNameOfGroup().equals(oldName)) {

if (!newName.trim().isEmpty()) {

group.setNameOfGroup(newName);

}

if (!newDescription.trim().isEmpty()) {

group.setGroupDescription(newDescription);

}

break;

}

}

}

/\*\*

\* Метод для видалення групи товарів

\* @param name - назва групи

\* @throws IllegalArgumentException

\*/

public void removeGroup(String name) throws IllegalArgumentException{

if (!groupExists(name)){

throw new IllegalArgumentException("Групи з такою назвою не існує.");

}

for (GroupOfItems group : groupsList){

if (group.getNameOfGroup().equals(name)){

groupsList.remove(group);

break;

}

}

}

/\*\*

\* Метод для збереження інформації у файли

\*/

public void updateFiles(){

try {

BufferedWriter writer1 = new BufferedWriter(new FileWriter(groupsFileName));

for (GroupOfItems group : groupsList){

writer1.write(group.getNameOfGroup() + ";" + group.getGroupDescription());

writer1.newLine();

}

writer1.flush();

writer1.close();

BufferedWriter writer2 = new BufferedWriter(new FileWriter(allProductsFileName));

for (GroupOfItems group : groupsList){

for (Items product : group.productsList){

writer2.write(group.getNameOfGroup() + ";" + product.getName() + ";" + product.getDescription() + ";" + product.getProducer() + ";" + product.getCount() + ";" + product.getPricePerOne());

writer2.newLine();

}

}

writer2.flush();

writer2.close();

} catch (IOException e) {

e.printStackTrace();

JOptionPane.showMessageDialog(null, "Помилка при записі у файли.", "Помилка", JOptionPane.ERROR\_MESSAGE);

System.exit(1);

}

}

/\*\*

\* Метод для отримання загальної вартості товарів на складі

\* @return - загальна вартість товарів на складі

\*/

public int getTotalStockValue(){

int totalValue = 0;

for (GroupOfItems group : groupsList){

for (Items product : group.productsList){

totalValue += product.getCount() \* product.getPricePerOne();

}

}

return totalValue;

}

}