Лабораторна робота №2

виконали:
Ангеліна Бабій
Дарія Колодяжна
викладачка:
Пєчкурова Олена Миколаївна

«Автоматизоване робоче місце невеликого підприємства для роботи зі складом»

NATIONAL UNIVERSITY OF
KYIV MOHYLA ACADEMY

Національний університет «Києво-Могилянська академія»

2021

Зміст

1. ***Постановка задачі.***

   Необхідно автоматизувати роботу невеликого підприємства для роботи зі складом.

   Існує декілька груп товарів. В кожній групі товарів існують конкретні товари. У кожного товару є наступні властивості: назва, опис, виробник, кількість на складі, ціна за одиницю. Група товарів містить назву та опис.
   - реалізувати графічний інтерфейс користувача;
   - збереження даних в файл/файли;
   - реалізувати додавання/редагування/видалення групи товарів - при видаленні групи товарів, видаляти і всі товари;
   - реалізувати додавання/редагування/видалення товару в групу товарів;
   - реалізувати інтерфейс додавання товару, інтерфейс списання товару;
   - пошук товару;
   - вивід статистичних даних: вивід всіх товарів з інформацією по складу, вивід усіх товарів по групі товарів з інформацією, загальна вартість товару на складі, загальна вартість товарів в групі.

2. ***Розподіл ролей в групі.***

   - *конструктор для товарів – Дарія;*
   - *конструктор для груп – Дарія;*
     - *в конструкторах деякі методи були дописані Ангеліною;*

   - *загальний інтерфейс – Ангеліна, допрацювала Дарія;*

   - *таблиця товарів – створена Ангеліною, допрацювала до кінця Дарія;*
   - *таблиця груп – Дарія;*

    - *створення кнопок та зв’язування з методами – Ангеліна, Дарія редагувала та допрацьовувала, перевіряла накладені умови.*

3. ***Опис усіх реалізованих можливостей.***

   - ***ADD ARTICLE** – додає товар до списку; необхідно вказати всі дані в поля;*
   - ***EDIT ARTICLE** – дозволяє редагувати лише товар, що вже існує;*
   - ***DELETE ARTICLE** – видаляє товар (товар має існувати).*

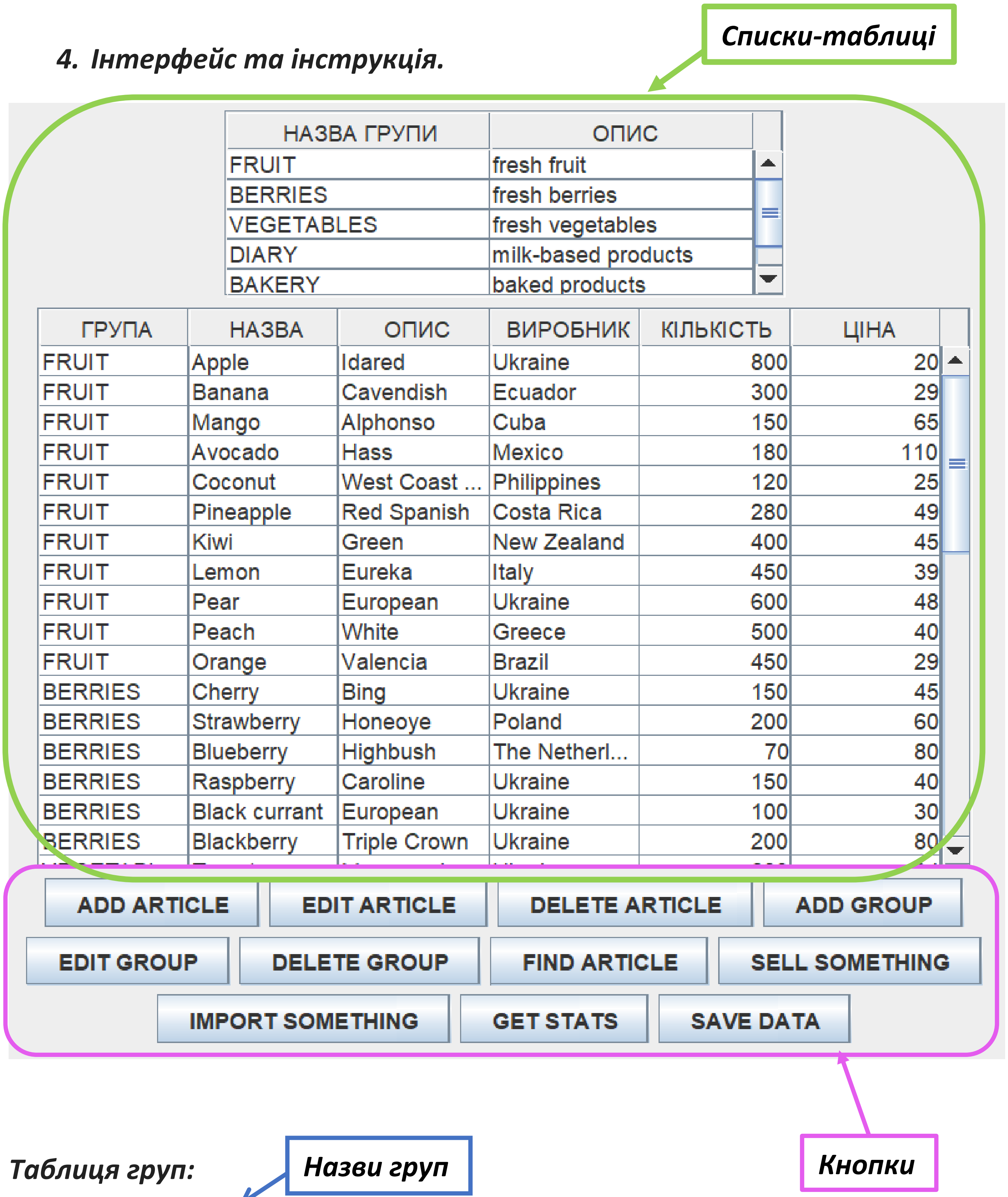
   - ***ADD GROUP** – додає групу з назвою та описом (пуста, поки не додати товарів);*
   - ***EDIT GROUP** – редагує групу, що вже існує (назву та опис);*
   - ***DELETE GROUP** – видаляє групу, а відповідно і всі товари, які у ній були.*

   - ***FIND ARTICLE** – пошук товару серед переліку.*

   - ***SELL SOMETHING** – списання товару (продаж); не можна списати 0 товарів, від’ємну кількість або ж більше, ніж є на складі;*
   - ***IMPORT SOMETHING** – імпортує (додає) товари; не можна імпортувати 0 товарів або від’ємну кількість.*

   - ***GET STATS** – виводить потрібну статистику на екран;*
   - ***SAVE DATA** – зберігає потрібну статистику у файл.*

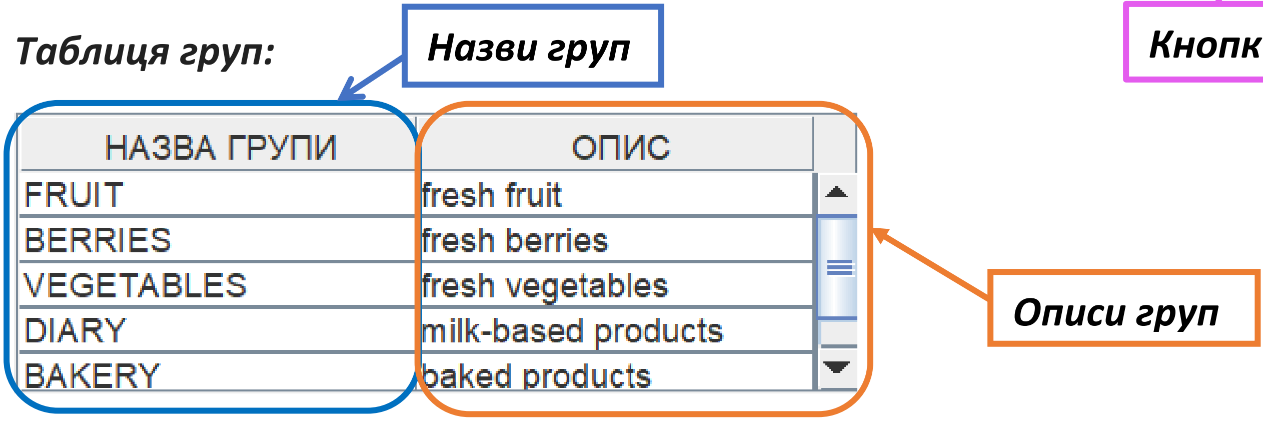
4. *Інтерфейс та інструкція.*



НАЗВА ГРУПИ	ОПИС
FRUIT	fresh fruit
BERRIES	fresh berries
VEGETABLES	fresh vegetables
DIARY	milk-based products
BAKERY	baked products

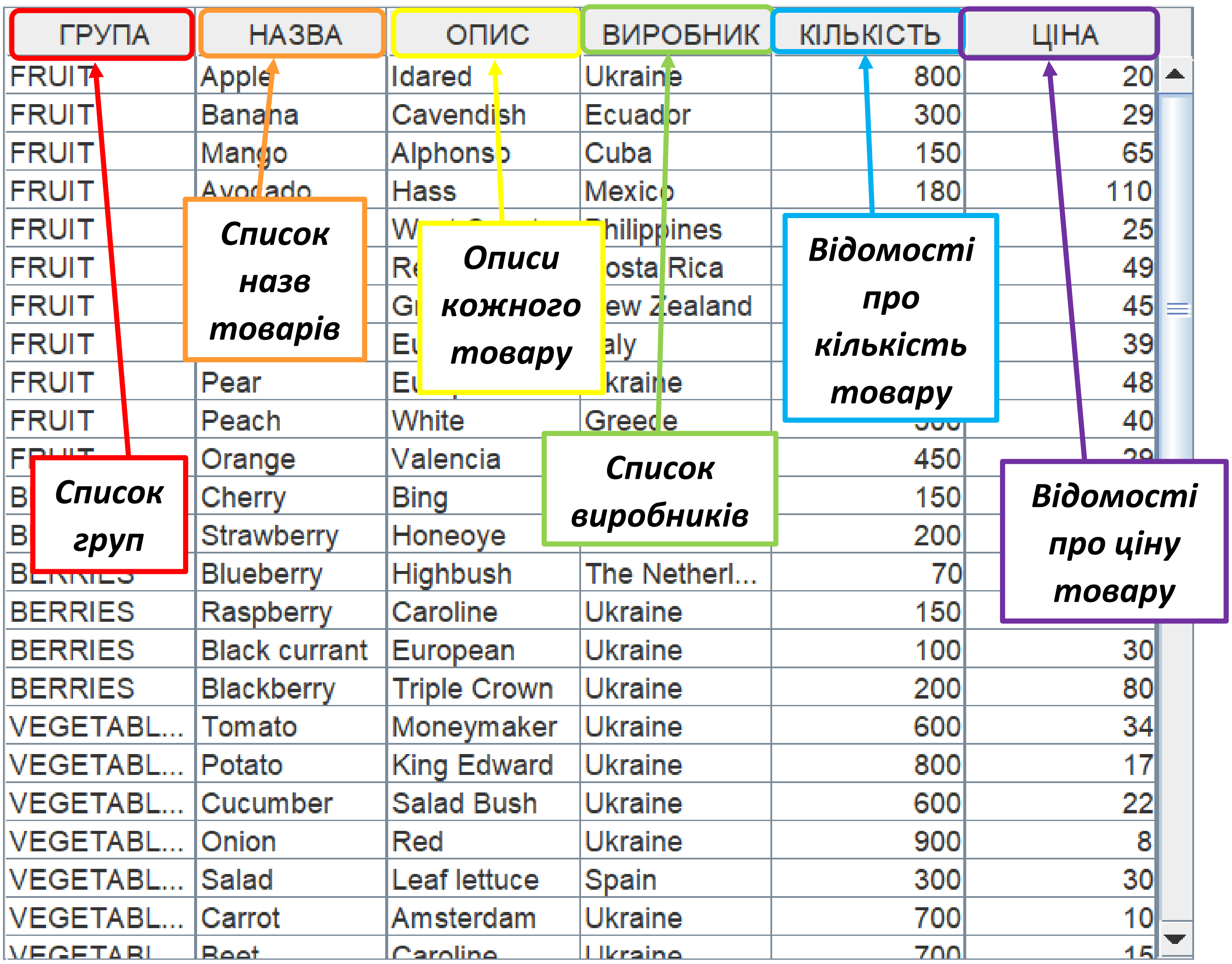
ГРУПА	НАЗВА	ОПИС	ВИРОБНИК	КІЛЬКІСТЬ	ЦІНА
FRUIT	Apple	Idared	Ukraine	800	20
FRUIT	Banana	Cavendish	Ecuador	300	29
FRUIT	Mango	Alphonso	Cuba	150	65
FRUIT	Avocado	Hass	Mexico	180	110
FRUIT	Coconut	West Coast ...	Philippines	120	25
FRUIT	Pineapple	Red Spanish	Costa Rica	280	49
FRUIT	Kiwi	Green	New Zealand	400	45
FRUIT	Lemon	Eureka	Italy	450	39
FRUIT	Pear	European	Ukraine	600	48
FRUIT	Peach	White	Greece	500	40
FRUIT	Orange	Valencia	Brazil	450	29
BERRIES	Cherry	Bing	Ukraine	150	45
BERRIES	Strawberry	Honeoye	Poland	200	60
BERRIES	Blueberry	Highbush	The Netherl...	70	80
BERRIES	Raspberry	Caroline	Ukraine	150	40
BERRIES	Black currant	European	Ukraine	100	30
BERRIES	Blackberry	Triple Crown	Ukraine	200	80

Таблиця груп:



НАЗВА ГРУПИ	ОПИС
FRUIT	fresh fruit
BERRIES	fresh berries
VEGETABLES	fresh vegetables
DIARY	milk-based products
BAKERY	baked products

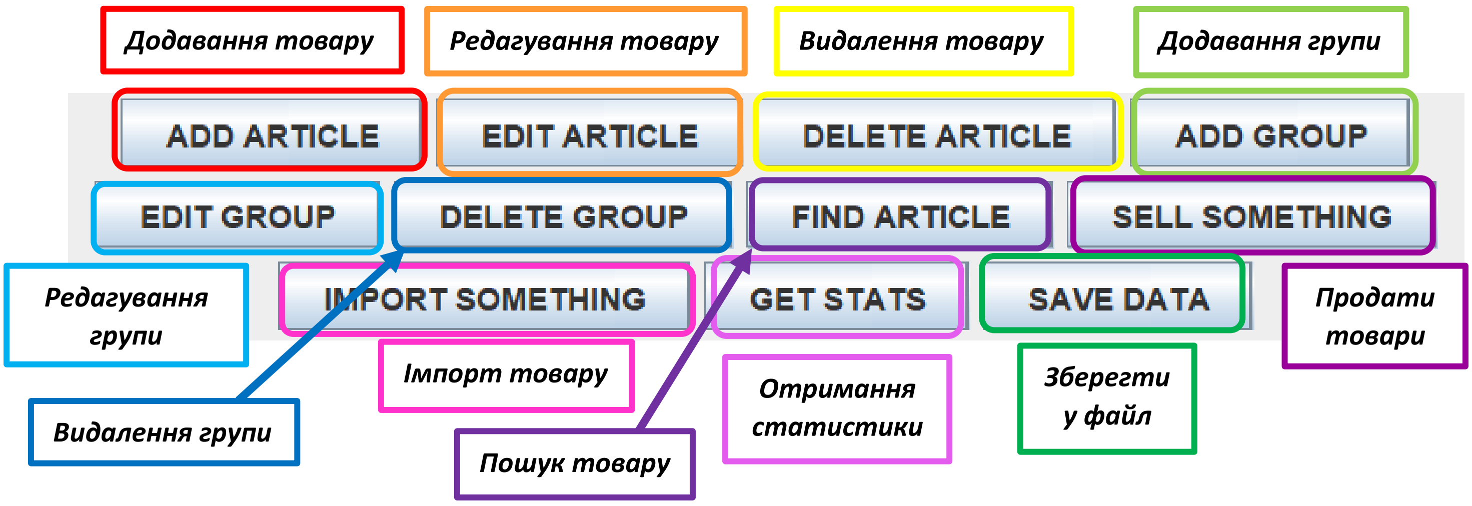
Таблиця товарів:



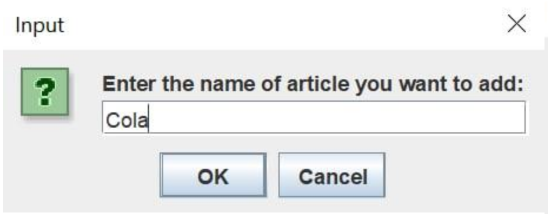
ГРУПА	НАЗВА	ОПИС	ВИРОБНИК	КІЛЬКІСТЬ	ЦІНА
FRUIT	Apple	Idared	Ukraine	800	20
FRUIT	Banana	Cavendish	Ecuador	300	29
FRUIT	Mango	Alphonso	Cuba	150	65
FRUIT	Avocado	Hass	Mexico	180	110
FRUIT	[illegible]	W	hilippines	[illegible]	25
FRUIT	[illegible]	Re	osta Rica	[illegible]	49
FRUIT	[illegible]	Gr	ew Zealand	[illegible]	45
FRUIT	[illegible]	Eu	aly	[illegible]	39
FRUIT	Pear	Eu	kraine	[illegible]	48
FRUIT	Peach	White	Greece	[illegible]	40
FRUIT	Orange	Valencia	[illegible]	450	[illegible]
B	Cherry	Bing	[illegible]	150	[illegible]
B	Strawberry	Honeoye	[illegible]	200	[illegible]
BERRIES	Blueberry	Highbush	The Netherl...	70	[illegible]
BERRIES	Raspberry	Caroline	Ukraine	150	[illegible]
BERRIES	Black currant	European	Ukraine	100	30
BERRIES	Blackberry	Triple Crown	Ukraine	200	80
VEGETABL...	Tomato	Moneymaker	Ukraine	600	34
VEGETABL...	Potato	King Edward	Ukraine	800	17
VEGETABL...	Cucumber	Salad Bush	Ukraine	600	22
VEGETABL...	Onion	Red	Ukraine	900	8
VEGETABL...	Salad	Leaf lettuce	Spain	300	30
VEGETABL...	Carrot	Amsterdam	Ukraine	700	10
VEGETABL	Beet	Caroline	Ukraine	700	15

Інструкція за кнопками:

У програмі представлено 11 кнопок:

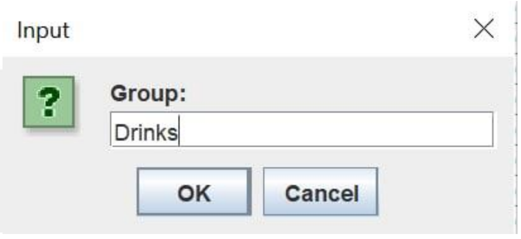


- ADD ARTICLE


Input

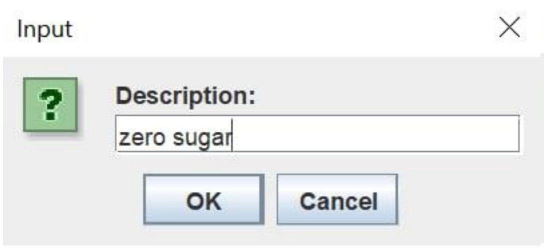
Enter the name of article you want to add:

Cola


Input

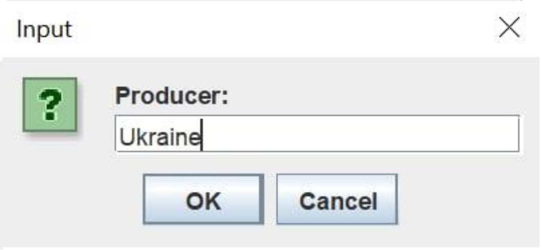
Group:

Drinks


Input

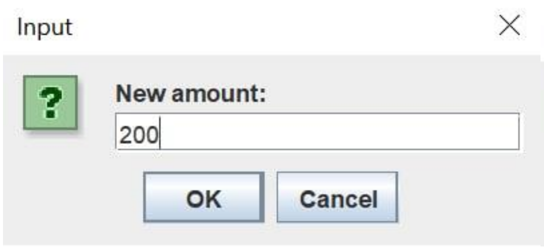
Description:

zero sugar


Input

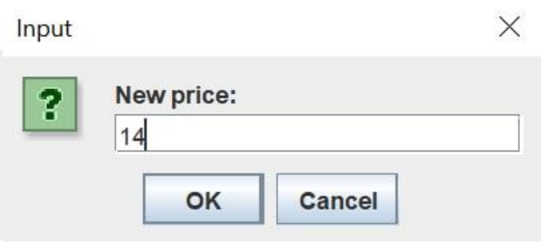
Producer:

Ukraine


Input

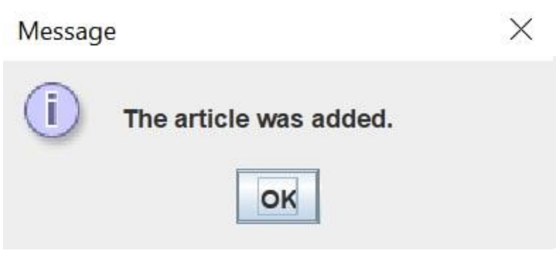
New amount:

200

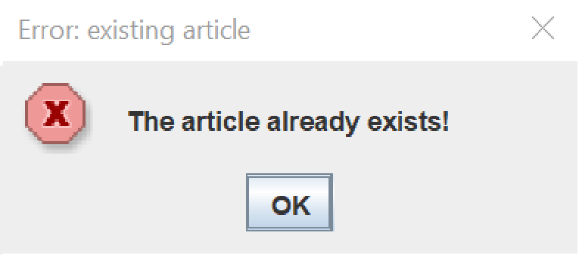

Input

New price:

14

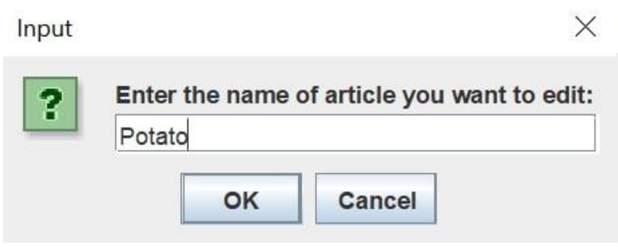

Message

The article was added.


Error: existing article

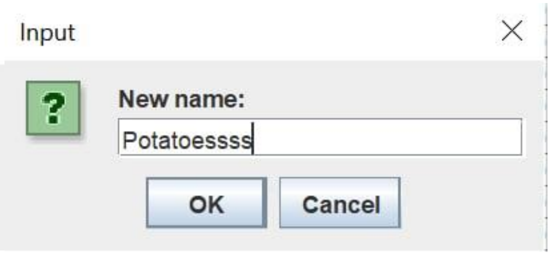
The article already exists!

- EDIT ARTICLE


Input

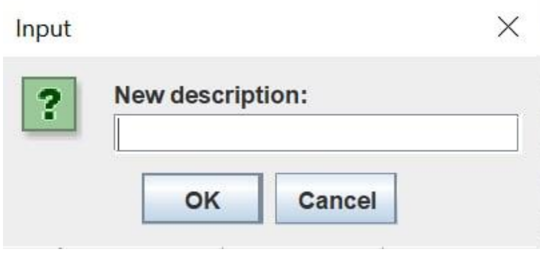
Enter the name of article you want to edit:

Potato

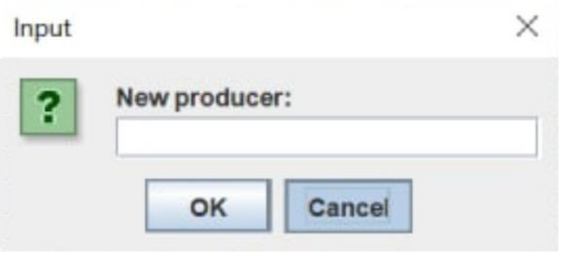

Input

New name:

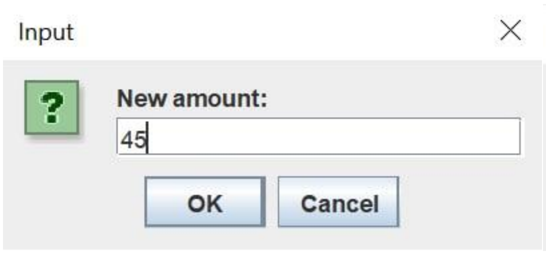
Potatoessss


Input

New description:

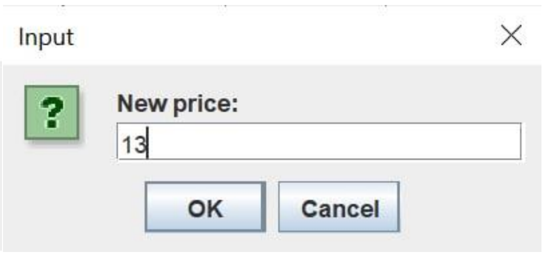

Input

New producer:


Input

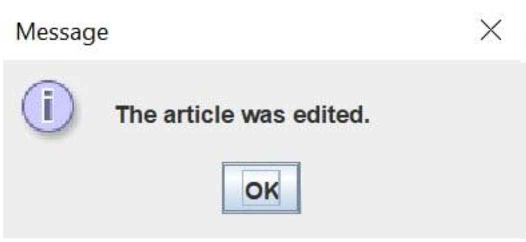
New amount:

45

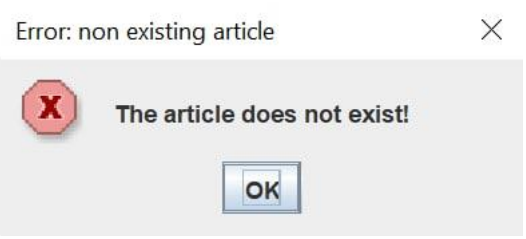

Input

New price:

13

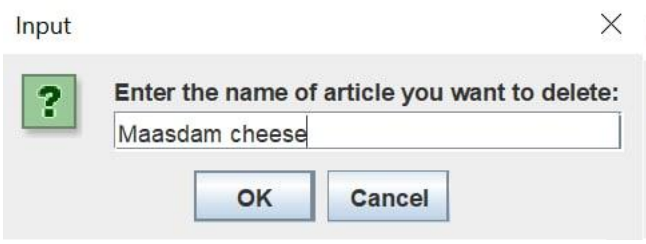

Message

The article was edited.


Error: non existing article

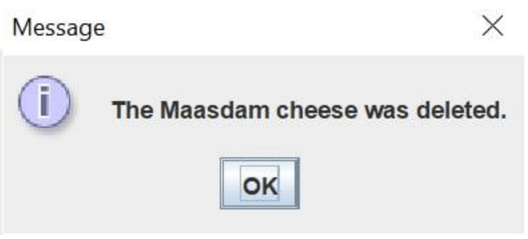
The article does not exist!

- DELETE ARTICLE

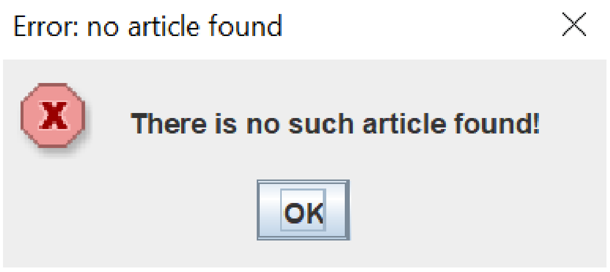

Input

Enter the name of article you want to delete:

Maasdam cheese

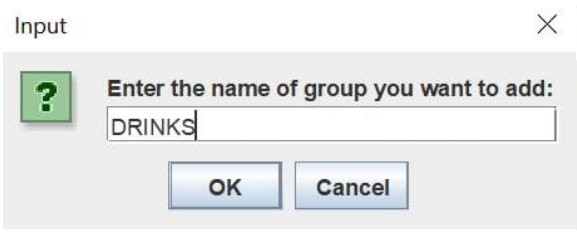

Message

The Maasdam cheese was deleted.


Error: no article found

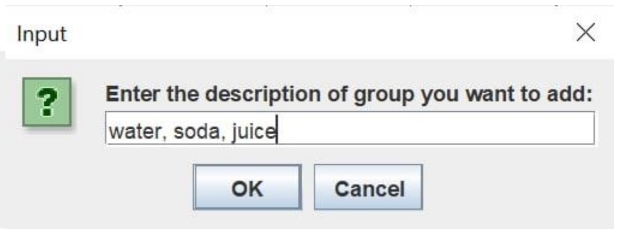
There is no such article found!

- ADD GROUP


Input

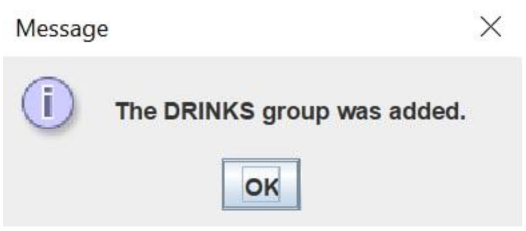
Enter the name of group you want to add:

DRINKS

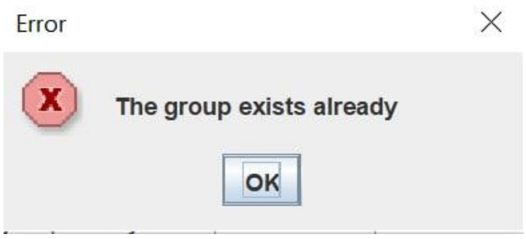

Input

Enter the description of group you want to add:

water, soda, juice

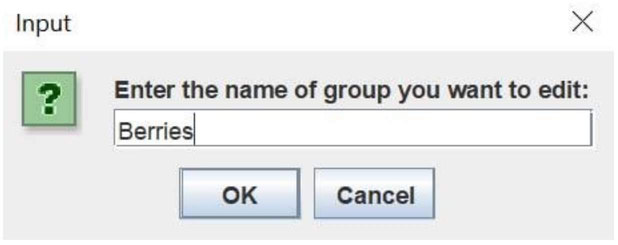

Message

The DRINKS group was added.


Error

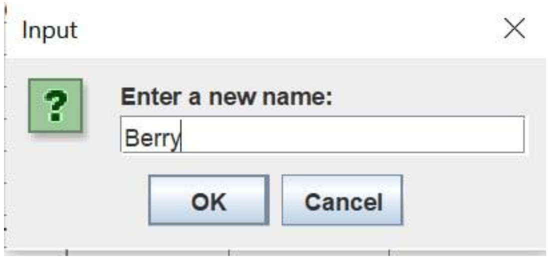
The group exists already

- EDIT GROUP


Input

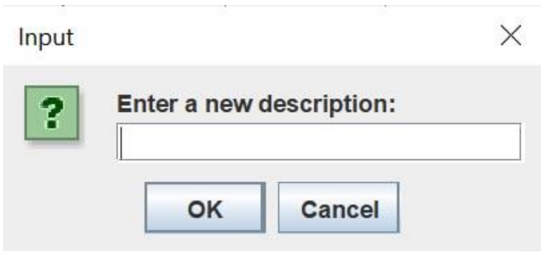
Enter the name of group you want to edit:

Berries

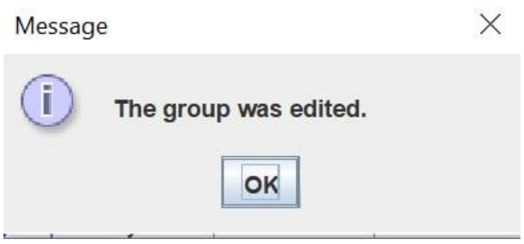

Input

Enter a new name:

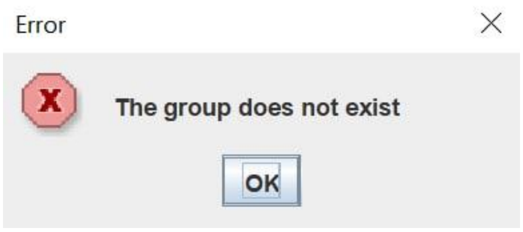
Berry


Input

Enter a new description:

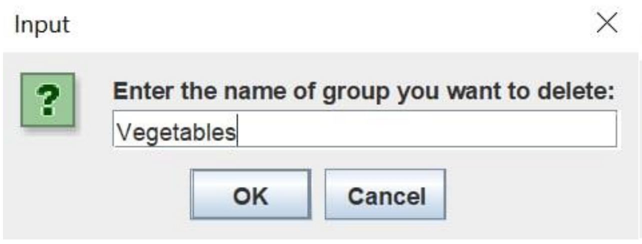

Message

The group was edited.


Error

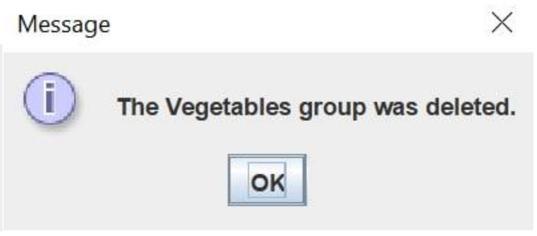
The group does not exist

- DELETE GROUP

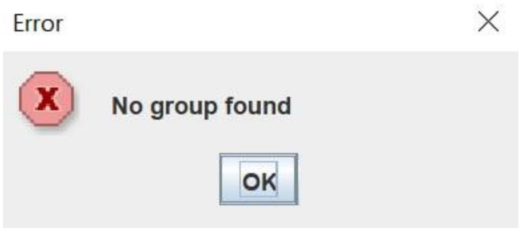

Input

Enter the name of group you want to delete:

Vegetables

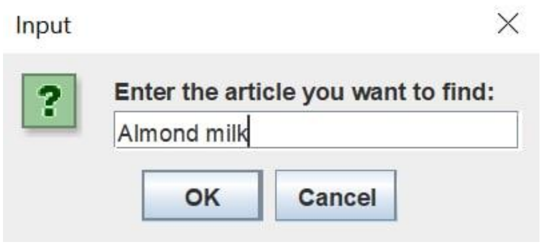

Message

The Vegetables group was deleted.


Error

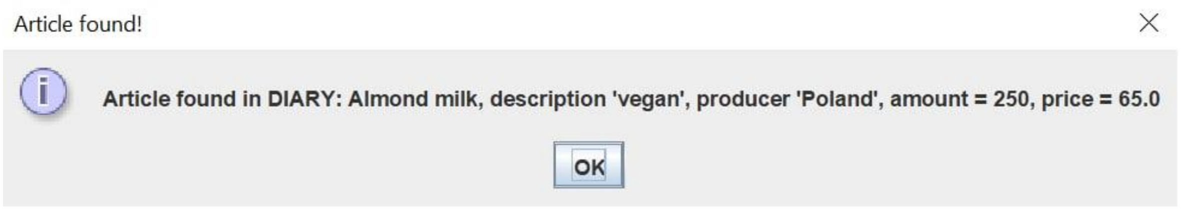
No group found

- FIND ARTICLE

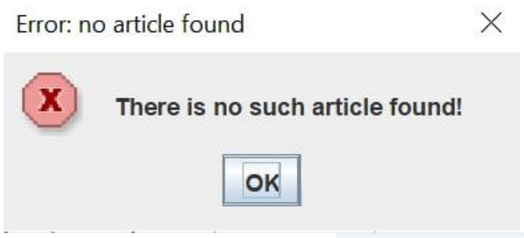

Input

Enter the article you want to find:

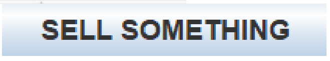
Almond milk

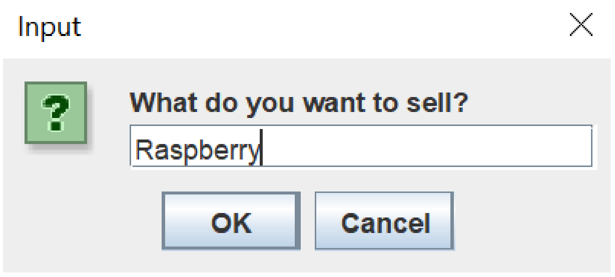

Article found!

Article found in DIARY: Almond milk, description 'vegan', producer 'Poland', amount = 250, price = 65.0


Error: no article found

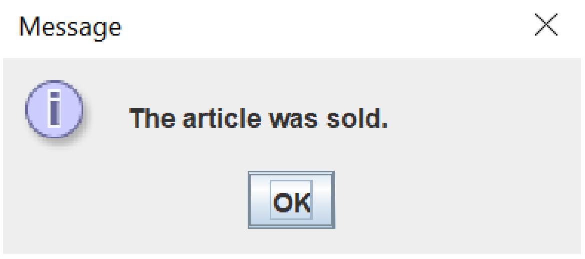
There is no such article found!

- SELL SOMETHING 

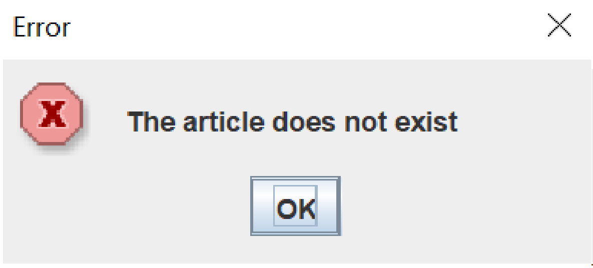

Input

What do you want to sell?

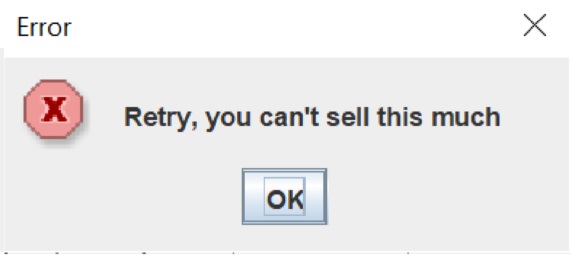
Raspberry


Message

The article was sold.

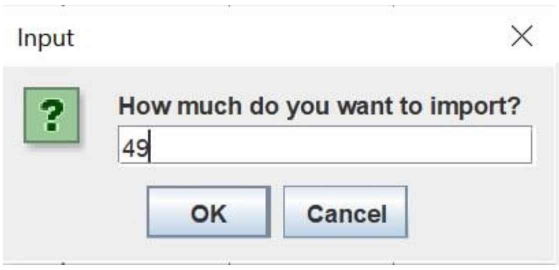

Error

The article does not exist


Error

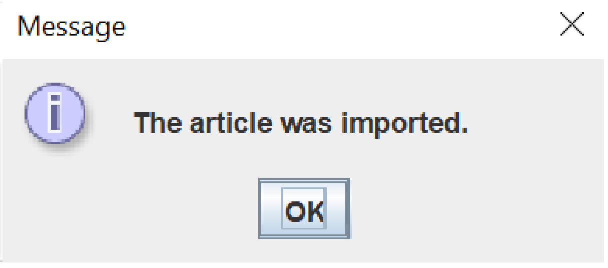
Retry, you can't sell this much

- IMPORT SOMETHING

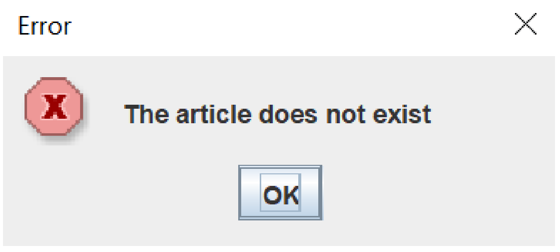

Input

How much do you want to import?

49

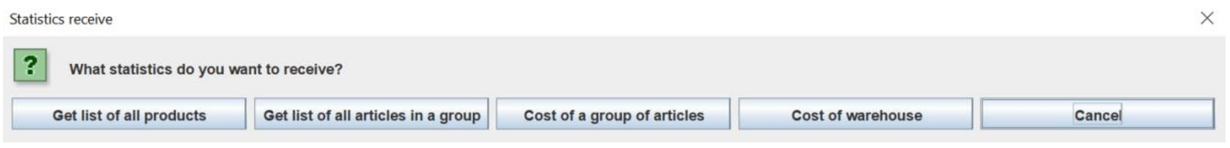

Message

The article was imported.

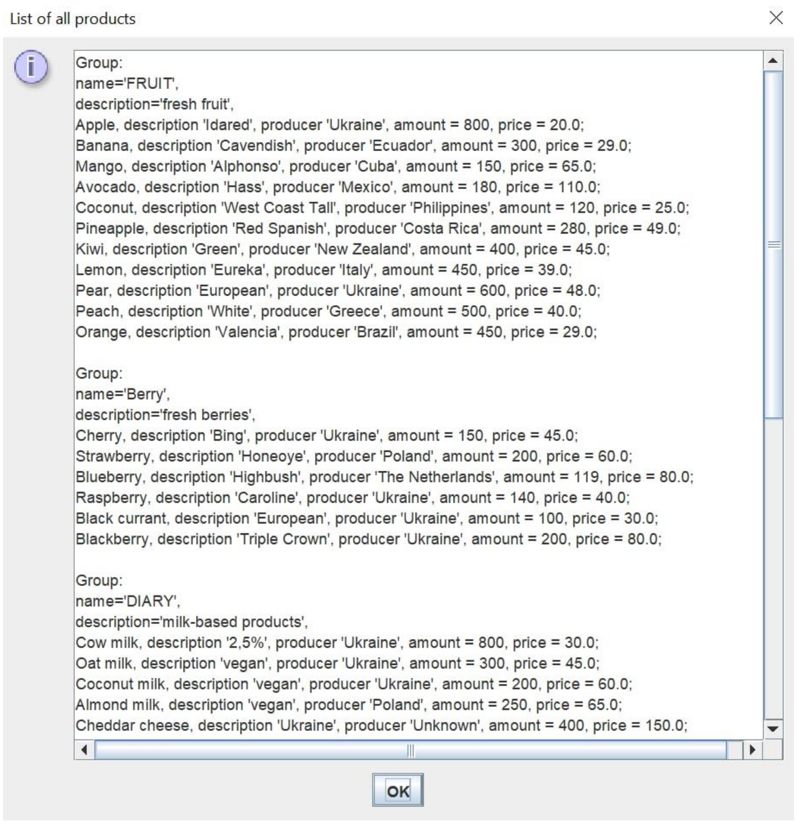


The article does not exist

- GET STATS

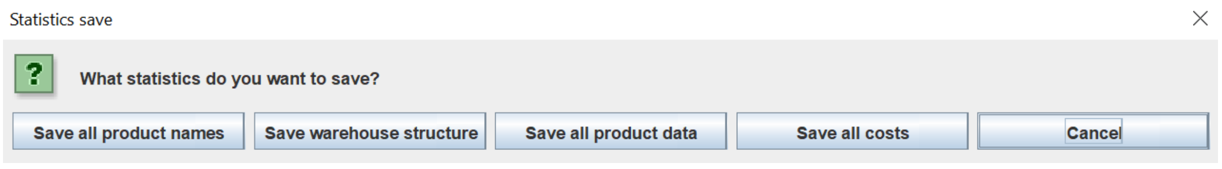


What statistics do you want to receive?

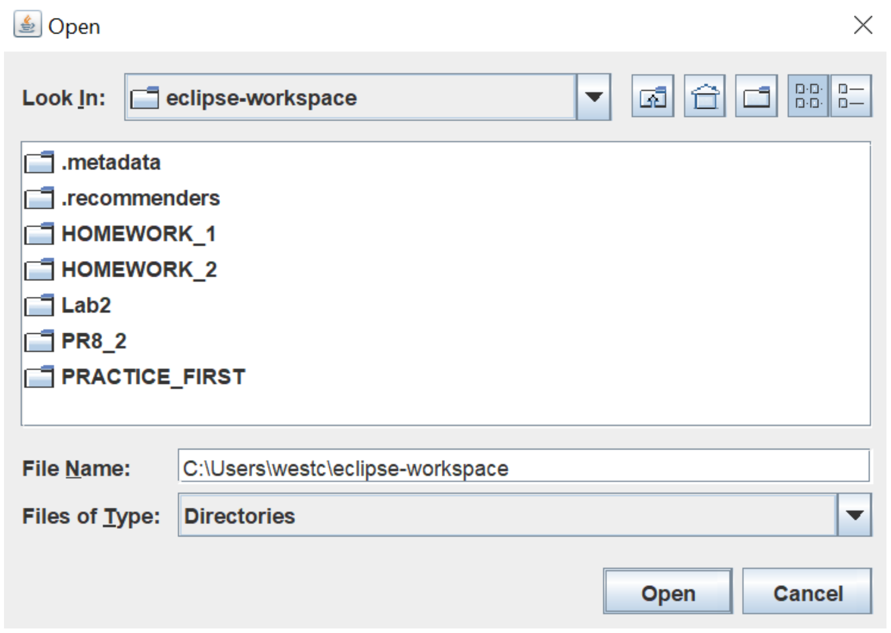


```
Group:
name='FRUIT',
description='fresh fruit',
Apple, description 'Idared', producer 'Ukraine', amount = 800, price = 20.0;
Banana, description 'Cavendish', producer 'Ecuador', amount = 300, price = 29.0;
Mango, description 'Alphonso', producer 'Cuba', amount = 150, price = 65.0;
Avocado, description 'Hass', producer 'Mexico', amount = 180, price = 110.0;
Coconut, description 'West Coast Tall', producer 'Philippines', amount = 120, price = 25.0;
Pineapple, description 'Red Spanish', producer 'Costa Rica', amount = 280, price = 49.0;
Kiwi, description 'Green', producer 'New Zealand', amount = 400, price = 45.0;
Lemon, description 'Eureka', producer 'Italy', amount = 450, price = 39.0;
Pear, description 'European', producer 'Ukraine', amount = 600, price = 48.0;
Peach, description 'White', producer 'Greece', amount = 500, price = 40.0;
Orange, description 'Valencia', producer 'Brazil', amount = 450, price = 29.0;

Group:
name='Berry',
description='fresh berries',
Cherry, description 'Bing', producer 'Ukraine', amount = 150, price = 45.0;
Strawberry, description 'Honeoye', producer 'Poland', amount = 200, price = 60.0;
Blueberry, description 'Highbush', producer 'The Netherlands', amount = 119, price = 80.0;
Raspberry, description 'Caroline', producer 'Ukraine', amount = 140, price = 40.0;
Black currant, description 'European', producer 'Ukraine', amount = 100, price = 30.0;
Blackberry, description 'Triple Crown', producer 'Ukraine', amount = 200, price = 80.0;

Group:
name='DIARY',
description='milk-based products',
Cow milk, description '2,5%', producer 'Ukraine', amount = 800, price = 30.0;
Oat milk, description 'vegan', producer 'Ukraine', amount = 300, price = 45.0;
Coconut milk, description 'vegan', producer 'Ukraine', amount = 200, price = 60.0;
Almond milk, description 'vegan', producer 'Poland', amount = 250, price = 65.0;
Cheddar cheese, description 'Ukraine', producer 'Unknown', amount = 400, price = 150.0;
```

- SAVE DATA

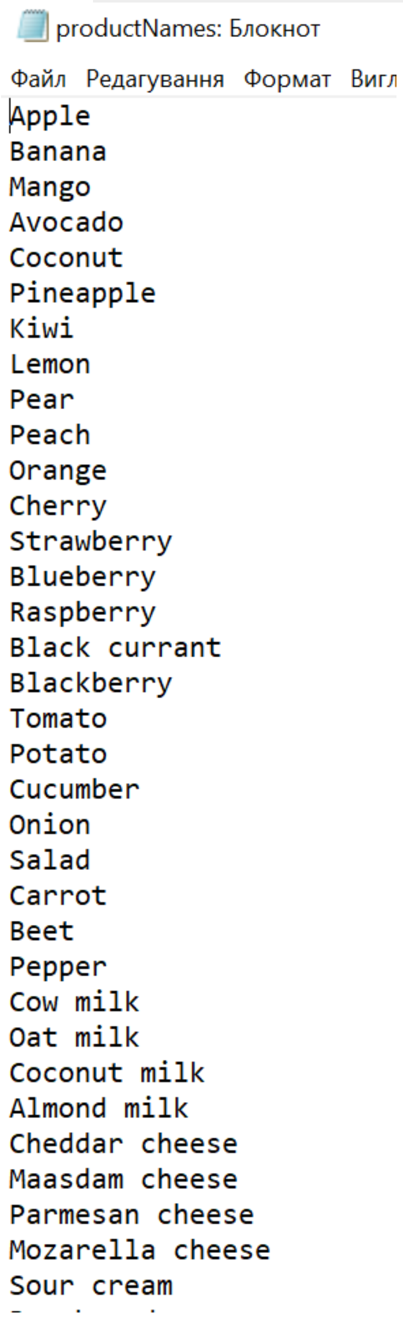


What statistics do you want to save?

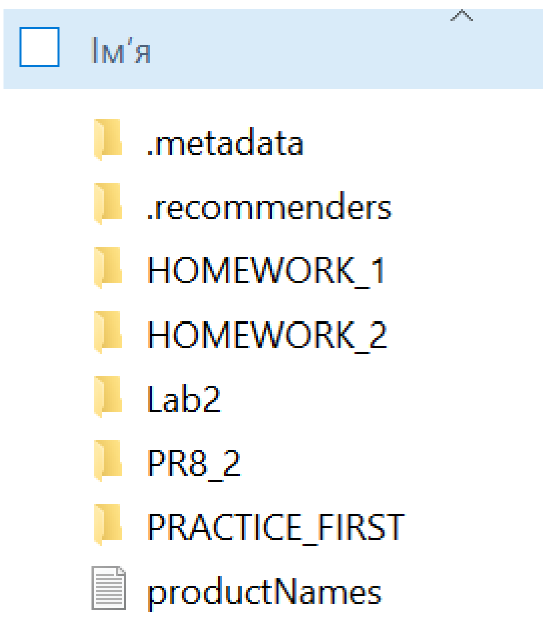


.metadata
.recommenders
HOMEWORK_1
HOMEWORK_2
Lab2
PR8_2
PRACTICE_FIRST

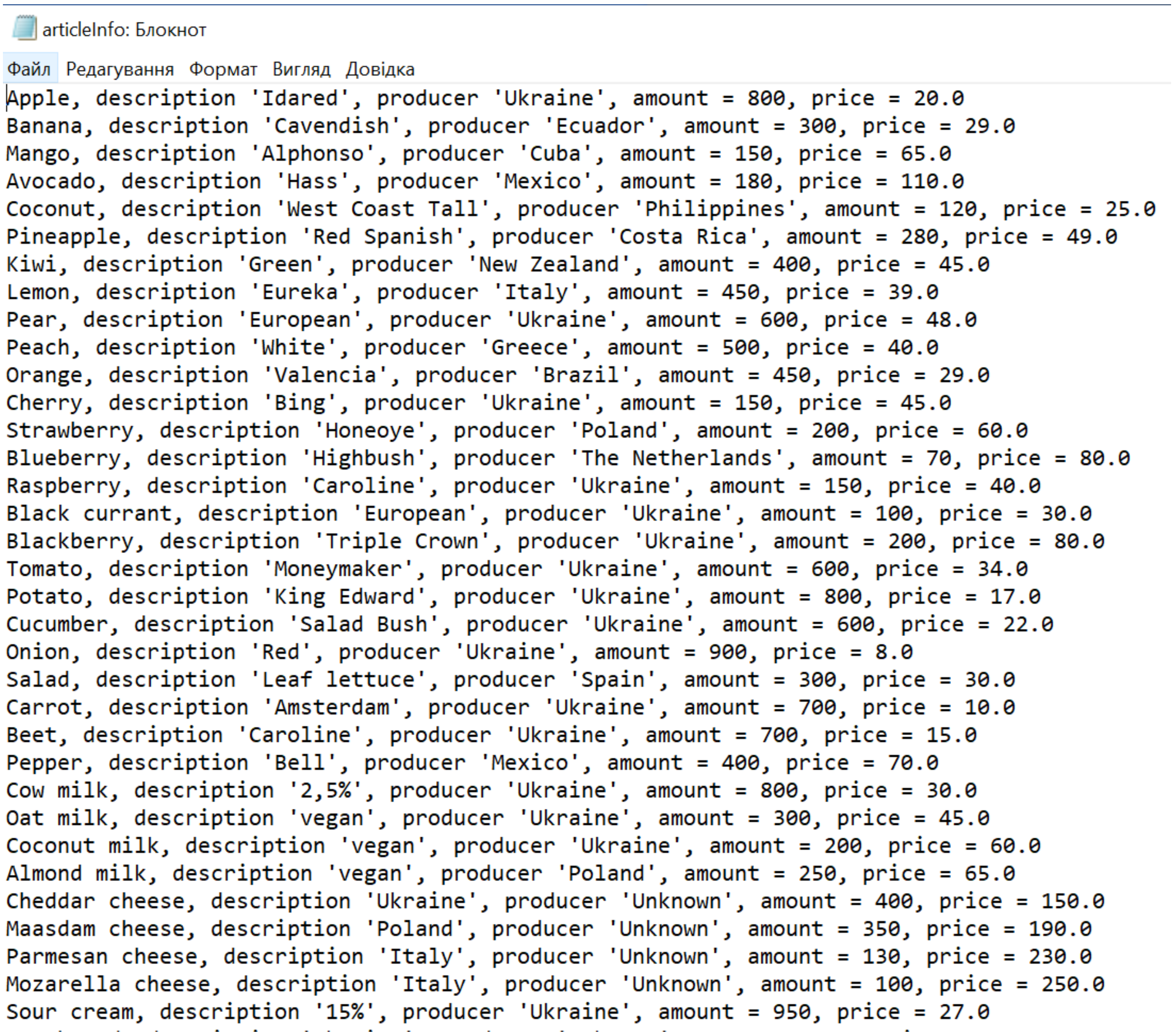
File Name: C:\Users\westc\eclipse-workspace
Files of Type: Directories



```
Apple
Banana
Mango
Avocado
Coconut
Pineapple
Kiwi
Lemon
Pear
Peach
Orange
Cherry
Strawberry
Blueberry
Raspberry
Black currant
Blackberry
Tomato
Potato
Cucumber
Onion
Salad
Carrot
Beet
Pepper
Cow milk
Oat milk
Coconut milk
Almond milk
Cheddar cheese
Maasdam cheese
Parmesan cheese
Mozarella cheese
Sour cream
```

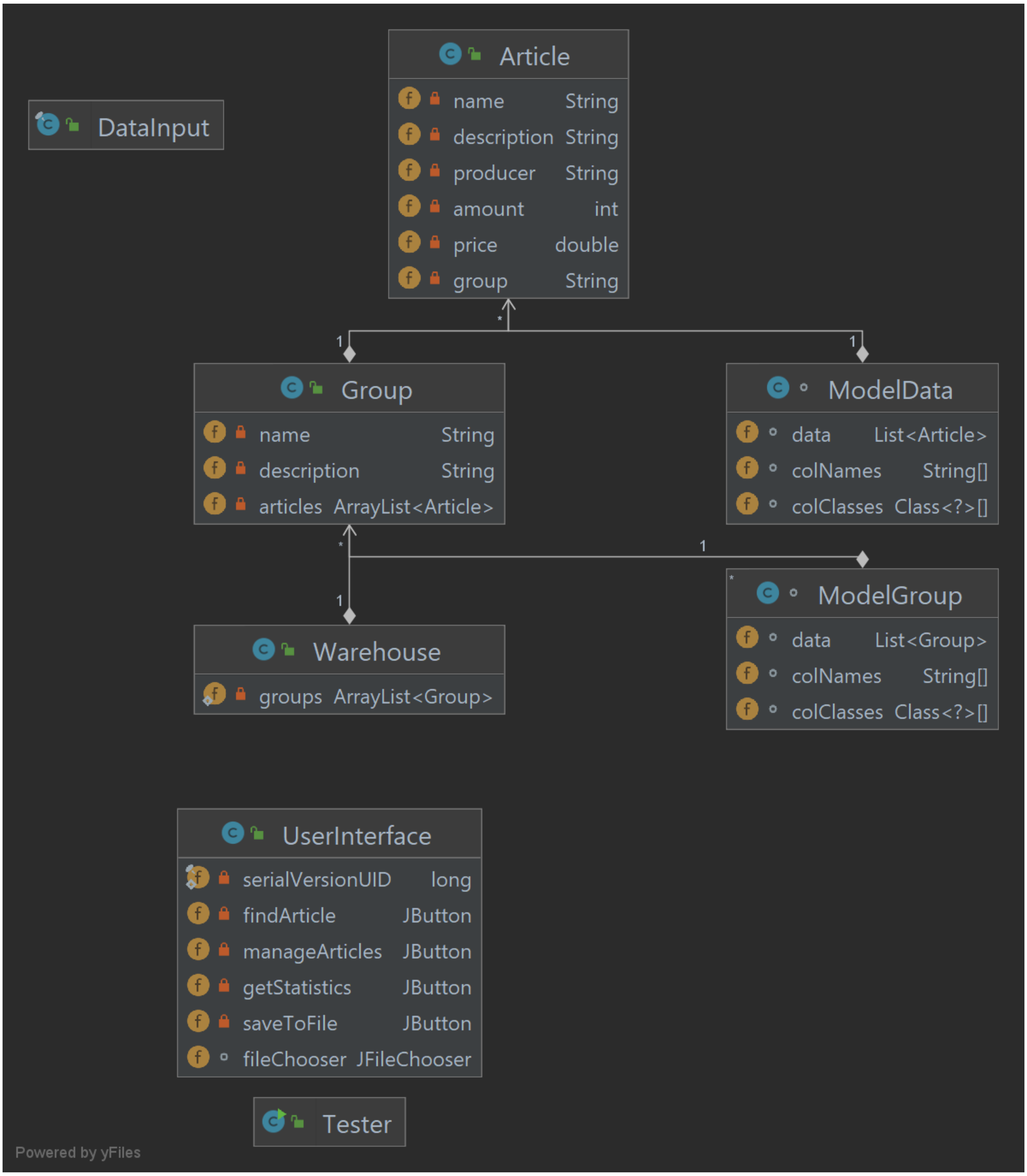


.metadata
.recommenders
HOMEWORK_1
HOMEWORK_2
Lab2
PR8_2
PRACTICE_FIRST
productNames



```
Apple, description 'Idared', producer 'Ukraine', amount = 800, price = 20.0
Banana, description 'Cavendish', producer 'Ecuador', amount = 300, price = 29.0
Mango, description 'Alphonso', producer 'Cuba', amount = 150, price = 65.0
Avocado, description 'Hass', producer 'Mexico', amount = 180, price = 110.0
Coconut, description 'West Coast Tall', producer 'Philippines', amount = 120, price = 25.0
Pineapple, description 'Red Spanish', producer 'Costa Rica', amount = 280, price = 49.0
Kiwi, description 'Green', producer 'New Zealand', amount = 400, price = 45.0
Lemon, description 'Eureka', producer 'Italy', amount = 450, price = 39.0
Pear, description 'European', producer 'Ukraine', amount = 600, price = 48.0
Peach, description 'White', producer 'Greece', amount = 500, price = 40.0
Orange, description 'Valencia', producer 'Brazil', amount = 450, price = 29.0
Cherry, description 'Bing', producer 'Ukraine', amount = 150, price = 45.0
Strawberry, description 'Honeoye', producer 'Poland', amount = 200, price = 60.0
Blueberry, description 'Highbush', producer 'The Netherlands', amount = 70, price = 80.0
Raspberry, description 'Caroline', producer 'Ukraine', amount = 150, price = 40.0
Black currant, description 'European', producer 'Ukraine', amount = 100, price = 30.0
Blackberry, description 'Triple Crown', producer 'Ukraine', amount = 200, price = 80.0
Tomato, description 'Moneymaker', producer 'Ukraine', amount = 600, price = 34.0
Potato, description 'King Edward', producer 'Ukraine', amount = 800, price = 17.0
Cucumber, description 'Salad Bush', producer 'Ukraine', amount = 600, price = 22.0
Onion, description 'Red', producer 'Ukraine', amount = 900, price = 8.0
Salad, description 'Leaf lettuce', producer 'Spain', amount = 300, price = 30.0
Carrot, description 'Amsterdam', producer 'Ukraine', amount = 700, price = 10.0
Beet, description 'Caroline', producer 'Ukraine', amount = 700, price = 15.0
Pepper, description 'Bell', producer 'Mexico', amount = 400, price = 70.0
Cow milk, description '2,5%', producer 'Ukraine', amount = 800, price = 30.0
Oat milk, description 'vegan', producer 'Ukraine', amount = 300, price = 45.0
Coconut milk, description 'vegan', producer 'Ukraine', amount = 200, price = 60.0
Almond milk, description 'vegan', producer 'Poland', amount = 250, price = 65.0
Cheddar cheese, description 'Ukraine', producer 'Unknown', amount = 400, price = 150.0
Maasdam cheese, description 'Poland', producer 'Unknown', amount = 350, price = 190.0
Parmesan cheese, description 'Italy', producer 'Unknown', amount = 130, price = 230.0
Mozarella cheese, description 'Italy', producer 'Unknown', amount = 100, price = 250.0
Sour cream, description '15%', producer 'Ukraine', amount = 950, price = 27.0
```

5. Структура даних.



6. ***Проблеми в роботі.***

   Довго сиділи з таблицями та їхнім заповненням з ArrayList. Не обійшлося і без помилок через неуважність. Також багато часу було витрачено на обдумування і прописування інтерфейсу. Труднощі додавалися й через те, що коди писалися паралельно і потрібно було слідкувати за тим, щоб всі версії коду були оновлені відповідно до обох студенток.

7. ***Висновки.***

   Загалом робота пройшла плідно. Звісно, працювати онлайн на відстані складно, бо кожний пише своє і потрібно все вчасно оновлювати. Але робота в парі пішла на користь, оскільки ми виправляли помилки одна одної та допомагали з питаннями, які виникали в ході виконання лабораторної.

8. Додаток. Код програми.

```
public class Article {


  private String name;
  private String description;
  private String producer;
  private int amount;
  private double price;
  private String group;

  Article(String name, String description) {
         this.name = name;
         this.description = description;
  }

  Article( String name, String description, String producer, int amount, double price) {

         this.name = name;
         this.description = description;
         this.producer = producer;
         this.amount = amount;
         this.price = price;
         if(name == null || name.isEmpty()) {
         this.name = "Unknown";
         }
         if(description == null ||description.isEmpty()) {
         this.description = "Unknown";
         }
         if(producer == null || producer.isEmpty()) {
         this.producer = "Unknown";
         }
  }

  Article(String name, String description, int amount, double price) {
         this.name = name;
         this.description = description;
         this.amount = amount;
         this.price = price;
         this.producer = "Unknown";
  }

  /**
   * Gets price
   * @return double
   */
  public double getPrice() {
         return price;
  }

  /**
   * Gets amount
   * @return int
   */
  public int getAmount() {
         return amount;
  }

  /**
   * Gets description
   * @return String
	 */
	public String getDescription() {
		return description;
	}

	/**
	 * Gets name
	 * @return String
	 */
	public String getName() {
		return name;
	}

	/**
	 * Gets producer
	 * @return String
	 */
	public String getProducer() {
		return producer;
	}

	public void setGroup(String group){
		this.group=group;
	}

	public String getGroup() {
		return group;
	}

	/**
	 * Sets amount
	 * @param amount
	 */
	public void setAmount(int amount) {
		this.amount = amount;
	}

	/**
	 * Sets description
	 * @param description
	 */
	public void setDescription(String description) {
		this.description = description;
	}

	/**
	 * Sets name
	 * @param name
	 */
	public void setName(String name) {
		this.name = name;
	}

	/**
	 * Sets price
	 * @param price
	 */
	public void setPrice(double price) {
		this.price = price;
	}

	/**
	 * Sets producer
	 * @param producer
	 */
	public void setProducer(String producer) {
		this.producer = producer;
	}

	@Override
	public String toString() {
		return name + ", description '" + description + '\'' + ", producer '" + producer +
'\'' + ", amount = " + amount
				+ ", price = " + price;
	}
}
```

```java
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public final class DataInput {
    public static Long getLong() throws IOException {
        String s = getString();
        Long value = Long.valueOf(s);
        return value;
    }

    public static char getChar() throws IOException {
        String s = getString();
        return s.charAt(0);
    }

    public static Integer getInt() {
        String s = "";
        try {
            s = getString();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        Integer value = Integer.valueOf(s);
        return value;

    }

    public static String getString() throws IOException {
        InputStreamReader isr = new InputStreamReader(System.in);
        BufferedReader br = new BufferedReader(isr);
        String s = br.readLine();
        return s;
    }

}
```

```java
import java.util.ArrayList;
import java.util.Locale;

public class Group {
    private String name;
    private String description;
    private  ArrayList<Article> articles;

    public Group() {
    }

    public Group(String name, String description, ArrayList<Article> articles) {
        this.name = name;
        this.description = description;
        if(name == null) {
             this.name = "Unknown";
        }
        if(description == null) {
             this.description = "Unknown";
        }
        this.articles = new ArrayList<>();
        this.articles.addAll(articles);
    }


    public Group(String name, String description) {
        this.name = name;
        this.description = description;
        if(name == null) {
             this.name = "Unknown";
        }
        if(description == null) {
             this.description = "Unknown";
        }
    }

    /**
     * Adds article to arraylist
     * @param name
     * @param description
     * @param producer
     * @param amount
     * @param price
     */
    public void addArticle(String name, String description,
                           String producer, int amount, double price) {
        if (articles == null) articles = new ArrayList<>();
        articles.add(new Article(name, description, producer, amount, price));
    }


    /**
     * Removes article from arraylist
     * @param name
     */
    public void removeArticle(String name) {
        for (int i = 0; i < articles.size(); i++) {
            if (articles.get(i).getName().equals(name)) articles.remove(articles.get(i));
        }
    }

    /**
     * Gets name
     * @return String
     */
    public String getName() {
        return name;
    }


    /**
     * Gets description
     * @return String
     */
    public String getDescription() {
        return description;
    }

    /**
     * Gets articles arraylist
     * @return ArrayList<Article>
     */
    public  ArrayList<Article> getArticles() {
        return articles;
    }

    /**
     * Gets article by name
     * @param name
     * @return Article
     */
    public Article getArticle(String name) {
      if(articles == null) {
            return null;
      }
        for (Article a : articles) {
            if (a.getName().toLowerCase(Locale.ROOT).equals(name.toLowerCase(Locale.ROOT)))
return a;
        }
        return null;
    }

    /**
     * Sets name
     * @param name
     */
    public void setName(String name) {
        this.name = name;
    }

    /**
     * Sets description
     * @param description
     */
    public void setDescription(String description) {
        this.description = description;
    }

    /**
     * Sets articles
     * @param articles
     */
    public void setArticles(ArrayList<Article> articles) {
        this.articles = articles;
    }
```

```java
    /**
     * Gets total price of a group
     * @return double
     */
    public double getTotalPrice() {
        double result = 0;
        if(articles == null) {
             return 0;
        }
        for (Article a : articles) {
            result +=  a.getPrice() *((double) a.getAmount());
        }
        return result;
    }

    @Override
    public String toString() {
        String artString = "";
        if(articles == null) {
             return "Group:\n" +
                    "name='" + name + "'" +
                    ", \ndescription='" + description + "'" +
                    ", \nThere are no articles in this group";
        }
        for (Article article : articles) {
            artString += article.toString() + "; \n";
        }
        return "Group:\n" +
                "name='" + name + "'" +
                ", \ndescription='" + description + "'" +
                ", \n" + artString;
    }
}
```

```java
import javax.swing.table.AbstractTableModel;
import java.util.List;

class ModelData extends AbstractTableModel {
    List<Article> data = Warehouse.mergeAll();

    String colNames[] = {"ГРУПА", "НАЗВА", "ОПИС", "ВИРОБНИК", "КІЛЬКІСТЬ", "ЦІНА"};
    Class<?> colClasses[] = {String.class, String.class, String.class, String.class,
Double.class, Double.class};

    ModelData() {
        super();

    }

    public int getRowCount() {
        return data.size();
    }

    public int getColumnCount() {
        return colNames.length;
    }

    public Object getValueAt(int rowIndex, int columnIndex) {
        if (columnIndex == 0) {
            String nameA = (String) getValueAt(rowIndex, 1);
            String nameG=Warehouse.getGroupByArticle(nameA).getName();
            return nameG;
        }
        if (columnIndex == 1) {
            return data.get(rowIndex).getName();
        }
        if (columnIndex == 2) {
            return data.get(rowIndex).getDescription();
        }
        if (columnIndex == 3) {
            return data.get(rowIndex).getProducer();
        }
        if (columnIndex == 4) {
            return data.get(rowIndex).getAmount();
        }
        if (columnIndex == 5) {
            return data.get(rowIndex).getPrice();
        }
        return null;
    }

    public String getColumnName(int columnIndex) {
        return colNames[columnIndex];
    }

    public Class<?> getColumnClass(int columnIndex) {
        return colClasses[columnIndex];
    }

    public boolean isCellEditable(int rowIndex, int columnIndex) {
        return false;
    }

    public void setValueAt(Object aValue, int rowIndex, int columnIndex) {
        if (columnIndex == 0) {

            data.get(rowIndex).setGroup((String) aValue);
        }
        if (columnIndex == 1) {
            data.get(rowIndex).setName((String) aValue);
        }
        if (columnIndex == 2) {
            data.get(rowIndex).setDescription((String) aValue);
        }
        if (columnIndex == 3) {
            data.get(rowIndex).setProducer((String) aValue);
        }
        if (columnIndex == 4) {
            data.get(rowIndex).setAmount((Integer) aValue);
        }
        if (columnIndex == 5) {
            data.get(rowIndex).setPrice((Double) aValue);
        }
        fireTableCellUpdated(rowIndex, columnIndex);
    }
    public void fireTableStructureChanged(){};
}
```

```java
import javax.swing.table.AbstractTableModel;
import java.util.List;

class ModelGroup extends AbstractTableModel {
    List<Group> data = Warehouse.allGroups();

    String colNames[] = {"НАЗВА ГРУПИ",  "ОПИС"};
    Class<?> colClasses[] = {String.class, String.class};

    ModelGroup() {
        super();

    }

    public int getRowCount() {
        return data.size();
    }

    public int getColumnCount() {
        return colNames.length;
    }

    public Object getValueAt(int rowIndex, int columnIndex) {

        if (columnIndex == 0) {
            return data.get(rowIndex).getName();
        }
        if (columnIndex == 1) {
            return data.get(rowIndex).getDescription();
        }

        return null;
    }

    public String getColumnName(int columnIndex) {
        return colNames[columnIndex];
    }

    public Class<?> getColumnClass(int columnIndex) {
        return colClasses[columnIndex];
    }

    public boolean isCellEditable(int rowIndex, int columnIndex) {
        return false;
    }

    public void setValueAt(Object aValue, int rowIndex, int columnIndex) {
        if (columnIndex == 0) {

            data.get(rowIndex).setName((String) aValue);
        }
        if (columnIndex == 1) {
            data.get(rowIndex).setDescription((String) aValue);
        }

    }

}
```

```
import javax.swing.SwingUtilities;

public class Tester {
    public static void main(String[] args) {

      Runnable FileChooserThread = new Runnable() {
                   public void run() {
                         new UserInterface();
                   }
             };

             SwingUtilities.invokeLater(FileChooserThread);

    }
}
```

```java
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Collections;

import javax.swing.*;
import javax.swing.filechooser.FileFilter;
import javax.swing.tree.DefaultMutableTreeNode;
import javax.swing.tree.DefaultTreeModel;

public class UserInterface extends JFrame {
	private static final long serialVersionUID = 2768304222530285489L;

	private JButton findArticle;
	private JButton manageArticles;
	private JButton getStatistics;
	private JButton saveToFile;

	JFileChooser fileChooser;

	public UserInterface() {
		super("FileChooserApp");
		initialize();
		this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		this.setLocation(400, 200);
	}

	public void initialize() {
		JFrame f = new JFrame("Warehouse");

		f.setLayout(new FlowLayout());
		f.setBounds(350,20,550, 650);

		setDefaultCloseOperation(EXIT_ON_CLOSE);

		fileChooser = new JFileChooser();
		fileChooser.setFileSelectionMode(JFileChooser.FILES_AND_DIRECTORIES);
		FileFilter filter = new FileFilter() {

			@Override
			public boolean accept(File f) {
				if (f.isDirectory()) {
					return true;
				} else
					return false;
			}

			@Override
			public String getDescription() {
				return "Directories";
			}
		};
		fileChooser.setFileFilter(filter);

		Warehouse warehouse = new Warehouse();
		// FRUIT
		warehouse.addGroup("FRUIT", "fresh fruit");
		ArrayList<Article> articles1 = new ArrayList<>();
		Collections.addAll(articles1,
                new Article("Apple", "Idared", "Ukraine", 800, 20),
                new Article("Banana", "Cavendish", "Ecuador", 300, 29),
                new Article("Mango", "Alphonso", "Cuba", 150, 65),
                new Article("Avocado", "Hass", "Mexico", 180, 110),
                new Article("Coconut", "West Coast Tall", "Philippines", 120, 25),
                new Article("Pineapple", "Red Spanish", "Costa Rica", 280, 49),
                new Article("Kiwi", "Green", "New Zealand", 400, 45),
                new Article("Lemon", "Eureka", "Italy", 450, 39),
                new Article("Pear", "European", "Ukraine", 600, 48),
                new Article("Peach", "White", "Greece", 500, 40),
                new Article("Orange", "Valencia", "Brazil", 450, 29));
        warehouse.getGroup("FRUIT").setArticles(articles1);

        // BERRIES
        warehouse.addGroup("BERRIES", "fresh berries");
        ArrayList<Article> articles2 = new ArrayList<>();
        Collections.addAll(articles2,
                new Article("Cherry", "Bing", "Ukraine", 150, 45),
                new Article("Strawberry", "Honeoye", "Poland", 200, 60),
                new Article("Blueberry", "Highbush", "The Netherlands", 70, 80),
                new Article("Raspberry", "Caroline", "Ukraine", 150, 40),
                new Article("Black currant", "European", "Ukraine", 100, 30),
                new Article("Blackberry", "Triple Crown", "Ukraine", 200, 80));
        warehouse.getGroup("BERRIES").setArticles(articles2);

        // VEGETABLES
        warehouse.addGroup("VEGETABLES", "fresh vegetables");
        ArrayList<Article> articles3 = new ArrayList<>();
        Collections.addAll(articles3,
                new Article("Tomato", "Moneymaker", "Ukraine", 600, 34),
                new Article("Potato", "King Edward", "Ukraine", 800, 17),
                new Article("Cucumber", "Salad Bush", "Ukraine", 600, 22),
                new Article("Onion", "Red", "Ukraine", 900, 8),
                new Article("Salad", "Leaf lettuce", "Spain", 300, 30),
                new Article("Carrot", "Amsterdam", "Ukraine", 700, 10),
                new Article("Beet", "Caroline", "Ukraine", 700, 15),
                new Article("Pepper", "Bell", "Mexico", 400, 70));
        warehouse.getGroup("VEGETABLES").setArticles(articles3);

        // DIARY
        warehouse.addGroup("DIARY", "milk-based products");
        ArrayList<Article> articles4 = new ArrayList<>();
        Collections.addAll(articles4,
                new Article("Cow milk", "2,5%", "Ukraine", 800, 30),
                new Article("Oat milk", "vegan", "Ukraine", 300, 45),
                new Article("Coconut milk", "vegan", "Ukraine", 200, 60),
                new Article("Almond milk", "vegan", "Poland", 250, 65),
                new Article("Cheddar cheese", "Ukraine", 400, 150),
                new Article("Maasdam cheese", "Poland", 350, 190),
                new Article("Parmesan cheese", "Italy", 130, 230),
                new Article("Mozarella cheese", "Italy", 100, 250),
                new Article("Sour cream", "15%", "Ukraine", 950, 27));
        warehouse.getGroup("DIARY").setArticles(articles4);

        // BAKERY
        warehouse.addGroup("BAKERY", "baked products");
        ArrayList<Article> articles5 = new ArrayList<>();
        Collections.addAll(articles5,
                new Article("Rye bread", "Ukraine", 800, 17),
                new Article("Toast bread", "Ukraine", 1000, 18),
                new Article("Baguette", "Ukraine", 400, 15),
                new Article("Croissant", "Cream, chocolate, cherry filled",
"Ukraine", 600, 18),
                         new Article("Pie", "Cherry, apple, peach filled", "Ukraine", 300, 
40),
                         new Article("Donut", "Glazed", "Ukraine", 400, 12));
             warehouse.getGroup("BAKERY").setArticles(articles5);

             // SWEETS
             warehouse.addGroup("SWEETS", "sugary products");
             ArrayList<Article> articles6 = new ArrayList<>();
             Collections.addAll(articles6,
                         new Article("Dark chocolate", "Ukraine", 800, 25),
                         new Article("Milk chocolate", "Ukraine", 1000, 23),
                         new Article("White chocolate", "Ukraine", 400, 26),
                         new Article("Cookies", "Chocolate chip", "Ukraine", 500, 27),
                         new Article("Candy", "Ukraine", 400, 16));
             warehouse.getGroup("SWEETS").setArticles(articles6);

             //TABLES
             JTable table = new JTable(new ModelData());
             JScrollPane scrollPane = new JScrollPane(table, 
JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED,
                         JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED);
             scrollPane.setPreferredSize(new Dimension(500, 400));

             JTable tableOfGroups = new JTable(new ModelGroup());
             JScrollPane scrollPaneG = new JScrollPane(tableOfGroups, 
JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED,
                         JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED);
             scrollPaneG.setPreferredSize(new Dimension(300, 100));
             f.add(scrollPaneG);

// ARTICLES
             // EDITING
             JButton editArticle = new JButton("EDIT ARTICLE");
             editArticle.addActionListener(new ActionListener() {

                   @Override
                   public void actionPerformed(ActionEvent e) {

                         String artName = "";
                         Group g;
                         Article a;

                         artName = JOptionPane.showInputDialog("Enter the name of article you 
want to edit: ");
                         if (artName == null) {
                               JOptionPane.showInternalMessageDialog(null, "No article 
entered", "Error: no input",
                                           JOptionPane.ERROR_MESSAGE);
                               return;
                         }
                         g = warehouse.getGroupByArticle(artName);
                         if (g == null) {
                               JOptionPane.showInternalMessageDialog(null, "The article does 
not exist!",
                                           "Error: non existing article",
JOptionPane.ERROR_MESSAGE);
                               return;
                         }
                         else {
                               Article art =g.getArticle(artName);
                               String n_n = JOptionPane.showInputDialog("New name:");
                               if(n_n != null &&!n_n.isEmpty()){
                                     art.setName(n_n);
```

```
                                }
                                String info = JOptionPane.showInputDialog("New description:");
                                if(info != null &&!info.isEmpty()){
                                        art.setDescription(info);
                                }
                                String producer = JOptionPane.showInputDialog("New 
producer:");
                                if(producer != null &&!producer.isEmpty()){
                                        art.setProducer(producer);
                                }
                                String amount = JOptionPane.showInputDialog("New amount:");
                                if(amount != null &&!amount.isEmpty()){

                                        while (!isNumber(amount)){
                                                JOptionPane.showInternalMessageDialog(null, "You 
can enter only positive numbers!",
                                                                "Error: not a number",

JOptionPane.ERROR_MESSAGE);
                                                String amount1 = JOptionPane.showInputDialog("New 

amount:");
                                                amount=amount1;
                                        }
                                        if(isNumber(amount)) {
                                                int b = Integer.parseInt(amount);
                                                art.setAmount(b);
                                        }

                                }
                                String price = JOptionPane.showInputDialog("New price:");
                                if(price != null &&!price.isEmpty()){
                                        while (!isNumber(price)){
                                                JOptionPane.showInternalMessageDialog(null, "You 
can enter only numbers!",
                                                                "Error: not a number",

JOptionPane.ERROR_MESSAGE);
                                                String pr = JOptionPane.showInputDialog("New 

amount:");
                                                price=pr;
                                        }
                                        if(isNumber(price)) {
                                                double p = Double.parseDouble(price);
                                                art.setPrice(p);
                                        }
                                }

                                table.setModel(new ModelData());
                                JOptionPane.showInternalMessageDialog(null, "The article was 
edited.");
                                return;
                                //}
                        }

                }

        });

        // ADDING
        JButton addArticle = new JButton("ADD ARTICLE");
        addArticle.addActionListener(new ActionListener() {

                @Override
                public void actionPerformed(ActionEvent e) {
```

```
                                String artName = "";
                                Group g;
                                Article a;

                                artName = JOptionPane.showInputDialog("Enter the name of article you 
want to add: ");
                                while (artName == null) {
                                        JOptionPane.showInternalMessageDialog(null, "No article 
entered", "Error: no input",
                                                        JOptionPane.ERROR_MESSAGE);
                                        String artName1 = JOptionPane.showInputDialog("Enter the name 
of article you want to add: ");
                                        artName= artName1;
                                }
                                g = warehouse.getGroupByArticle(artName);
                                if (g == null) {
                                        // adding article
                                        String g_name = JOptionPane.showInputDialog("Group:");
                                        // ПЕРЕВІРКА НА ІСНУВАННЯ ГРУПИ if (groupName)
                                        if (warehouse.getGroup(g_name)==null) {
                                                JOptionPane.showInternalMessageDialog(null, "No group 
found, try again", "Error",
                                                                JOptionPane.ERROR_MESSAGE);
                                                return;
                                        }
                                        else {
                                                Group group = warehouse.getGroup(g_name);
                                                String info = 
JOptionPane.showInputDialog("Description:");
                                                String producer = 
JOptionPane.showInputDialog("Producer:");
                                                String amount = JOptionPane.showInputDialog("New 
amount:");

                                                int b;
                                                if(amount != null &&!amount.isEmpty()){

                                                        while (!isNumber(amount)){
                                                                JOptionPane.showInternalMessageDialog(null, 
"You can enter only positive numbers!",
                                                                                "Error: not a number", 
JOptionPane.ERROR_MESSAGE);
                                                                String amount1 = 
JOptionPane.showInputDialog("New amount:");
                                                                amount=amount1;
                                                        }
                                                        if(isNumber(amount)) {
                                                                 b = Integer.parseInt(amount);
                                                                String price = 
JOptionPane.showInputDialog("New price:");
                                                                if(price != null &&!price.isEmpty()){
                                                                        while (!isNumber(price)){
        JOptionPane.showInternalMessageDialog(null, "You can enter only numbers!", 
                                                                                        "Error: not a 
number", JOptionPane.ERROR_MESSAGE);
                                                                                String pr = 
JOptionPane.showInputDialog("New price:");
                                                                                price=pr;
                                                                        }
                                                                        if(isNumber(price)) {
                                                                                double p = 
Double.parseDouble(price);
```

```
                                                                    warehouse.addArticle(group, 
artName, info, producer, b, p);
                                                                    table.setModel(new 
ModelData());
                                                                    
      JOptionPane.showInternalMessageDialog(null, "The article was added.");
                                                                    return;
                                                                    
                                                  }
                                                  
                                            }else {
                                                              warehouse.addArticle(group, artName, 
info, producer, b, 0.0);
                                                              table.setModel(new ModelData());
                                                              
      JOptionPane.showInternalMessageDialog(null, "The article was added.");
                                                              return;
                                                              
                                                        }
                                                  }
                                            }
                                      }
                                }
                                else {
                                      JOptionPane.showInternalMessageDialog(null, "The article 
already exists!",
                                                  "Error: existing article", 
JOptionPane.ERROR_MESSAGE);
                                      return;
                                }
                          }
                          
                    });
                    
                    // DELETING
                    JButton deleteArticle = new JButton("DELETE ARTICLE");
                    deleteArticle.addActionListener(new ActionListener() {
                          
                          @Override
                          public void actionPerformed(ActionEvent e) {
                                
                                String artName = "";
                                Group g;
                                Article a;
                                
                                artName = JOptionPane.showInputDialog("Enter the name of article you 
want to delete: ");
                                if (artName == null) {
                                      JOptionPane.showInternalMessageDialog(null, "No article 
entered", "Error: no input",
                                                  JOptionPane.ERROR_MESSAGE);
                                      return;
                                } else {
                                      a = warehouse.getArticle(artName);
                                      if (a == null) {
                                            JOptionPane.showInternalMessageDialog(null, "There is no 
such article found!",
                                                        "Error: no article found", 
JOptionPane.ERROR_MESSAGE);
                                            return;
                                      } else {
                                            warehouse.deleteArticle(artName);
                                            table.setModel(new ModelData());
```

```
                                        JOptionPane.showInternalMessageDialog(null, "The " +
artName + " was deleted.");
                                        return;
                                  }
                            }
                      }
                });

                // FINDING
                findArticle = new JButton("FIND ARTICLE");
                findArticle.addActionListener(new ActionListener() {

                      @Override
                      public void actionPerformed(ActionEvent e) {

                            String artName = "";
                            Group g;
                            Article a;

                            artName = JOptionPane.showInputDialog("Enter the article you want to
find: ");
                            if (artName == null) {
                                  JOptionPane.showInternalMessageDialog(null, "No article
entered, cannot find", "Error: no input",
                                              JOptionPane.ERROR_MESSAGE);
                                  return;
                            }
                            g = warehouse.getGroupByArticle(artName);
                            if (g == null) {
                                  JOptionPane.showInternalMessageDialog(null, "There is no such
article found!",
                                              "Error: no article found",
JOptionPane.ERROR_MESSAGE);
                                  return;
                            }
                            a = warehouse.getArticle(artName);
                            JOptionPane.showInternalMessageDialog(null, "Article found in " +
g.getName() + ": " + a.toString(),
                                        "Article found!", JOptionPane.INFORMATION_MESSAGE);
                      }

                });



// GROUP
                // DELETE
                JButton deleteGroup = new JButton("DELETE GROUP");
                deleteGroup.addActionListener(new ActionListener() {

                      @Override
                      public void actionPerformed(ActionEvent e) {

                            String groupName = "";
                            Group g;
                            Article a;

                            groupName = JOptionPane.showInputDialog("Enter the name of group you
want to delete: ");
                            if (groupName == null) {
                                  JOptionPane.showInternalMessageDialog(null, "No group
entered", "Error: no input",
                                              JOptionPane.ERROR_MESSAGE);
                                        return;
                                }
                                // ПЕРЕВІРКА НА ІСНУВАННЯ ГРУПИ if (groupName)
                                if (warehouse.getGroup(groupName)==null) {
                                        JOptionPane.showInternalMessageDialog(null, "No group found",
"Error",
                                                        JOptionPane.ERROR_MESSAGE);
                                        return;
                                }
                                else {
                                        warehouse.removeGroup(groupName);
                                        table.setModel(new ModelData());
                                        tableOfGroups.setModel(new ModelGroup());
                                        JOptionPane.showInternalMessageDialog(null, "The " +
groupName +" group was deleted.");
                                        return;
                                }
                        }

                });

                // EDIT
                JButton editGroup = new JButton("EDIT GROUP");
                editGroup.addActionListener(new ActionListener() {

                        @Override
                        public void actionPerformed(ActionEvent e) {

                                String groupName = "";
                                groupName = JOptionPane.showInputDialog("Enter the name of group you
want to edit: ");
                                if (groupName == null) {
                                        JOptionPane.showInternalMessageDialog(null, "No group
entered", "Error: no input",
                                                        JOptionPane.ERROR_MESSAGE);
                                        return;
                                }
                                if (warehouse.getGroup(groupName)==null) {
                                        JOptionPane.showInternalMessageDialog(null, "The group does
not exist", "Error",
                                                        JOptionPane.ERROR_MESSAGE);
                                        return;
                                }
                                else {
                                        Group g =warehouse.getGroup(groupName);
                                        String group_n = JOptionPane.showInputDialog("Enter a new
name: ");
                                        if(group_n!= null &&!group_n.isEmpty()){
                                                g.setName(group_n);
                                        }
                                        String group_desc = JOptionPane.showInputDialog("Enter a new
description: ");
                                        if(group_desc!= null &&!group_desc.isEmpty()){
                                                g.setDescription(group_desc);
                                        }


                                        table.setModel(new ModelData());
                                        tableOfGroups.setModel(new ModelGroup());
                                        JOptionPane.showInternalMessageDialog(null, "The group was
edited.");
                                        return;
                                }
```

```
            }

        });

        // ADD
        JButton addGroup = new JButton("ADD GROUP");
        addGroup.addActionListener(new ActionListener() {

            @Override
            public void actionPerformed(ActionEvent e) {
                String group_desc = "";
                String groupName = "";

                groupName = JOptionPane.showInputDialog("Enter the name of group you
want to add: ");
                if (groupName == null) {
                    JOptionPane.showInternalMessageDialog(null, "No group
entered", "Error: no input",
                            JOptionPane.ERROR_MESSAGE);
                    return;
                }
                // перевірка на неіснування групи
                if (warehouse.getGroup(groupName)!=null) {
                    JOptionPane.showInternalMessageDialog(null, "The group exists
already", "Error",
                            JOptionPane.ERROR_MESSAGE);
                    return;
                }
                else {
                    group_desc = JOptionPane.showInputDialog("Enter the
description of group you want to add: ");
                    if(group_desc!= null &&!group_desc.isEmpty()){
                        warehouse.addGroup(groupName, group_desc);
                    }else
                        warehouse.addGroup(groupName, "Unknown");

                    table.setModel(new ModelData());
                    tableOfGroups.setModel(new ModelGroup());
                    JOptionPane.showInternalMessageDialog(null, "The " +
groupName +" group was added.");
                    return;
                }

            }

        });



        JButton sell = new JButton("SELL SOMETHING");
        sell.addActionListener(new ActionListener() {

            @Override
            public void actionPerformed(ActionEvent e) {

                String artName = "";
                artName = JOptionPane.showInputDialog("What do you want to sell?");
                if (artName == null) {
                    JOptionPane.showInternalMessageDialog(null, "Nothing was
entered", "Error: no input",
                            JOptionPane.ERROR_MESSAGE);
                    return;
```

```
                        }
                        if (warehouse.getArticle(artName)==null) {
                              JOptionPane.showInternalMessageDialog(null, "The article does
not exist", "Error",
                                          JOptionPane.ERROR_MESSAGE);
                              return;
                        }
                        else {
                              String count = JOptionPane.showInputDialog("How much do you
want to sell?");
                              while (!isNumber(count)){
                                    JOptionPane.showInternalMessageDialog(null, "You can
enter only numbers!",
                                                "Error: not a number",
JOptionPane.ERROR_MESSAGE);
                                    String amount1 = JOptionPane.showInputDialog("New
amount:");
                                    count=amount1;
                              }
                              int counter = Integer.parseInt(count);
                              if (warehouse.sellArticle(artName, counter)==0) {
                                    JOptionPane.showInternalMessageDialog(null, "Retry, you
can't sell this much", "Error",
                                                JOptionPane.ERROR_MESSAGE);
                              }
                              else {
                                    table.setModel(new ModelData());
                                    JOptionPane.showInternalMessageDialog(null, "The article
was sold.");
                                    return;
                              }
                        }
                  }

            });


            JButton importing = new JButton("IMPORT SOMETHING");
            importing.addActionListener(new ActionListener() {

                  @Override
                  public void actionPerformed(ActionEvent e) {

                        String artName = "";
                        artName = JOptionPane.showInputDialog("What do you want to import?");
                        if (artName == null) {
                              JOptionPane.showInternalMessageDialog(null, "Nothing was
entered", "Error: no input",
                                          JOptionPane.ERROR_MESSAGE);
                              return;
                        }
                        if (warehouse.getArticle(artName)==null) {
                              JOptionPane.showInternalMessageDialog(null, "The article does
not exist", "Error",
                                          JOptionPane.ERROR_MESSAGE);
                              return;
                        }
                        else {
                              String count = JOptionPane.showInputDialog("How much do you
want to import?");
                              while (!isNumber(count)){
                                    JOptionPane.showInternalMessageDialog(null, "You can
enter only positive numbers!",
                                                "Error: not a number",
JOptionPane.ERROR_MESSAGE);
                                        String amount1 = JOptionPane.showInputDialog("New
amount:");
                                        count=amount1;
                                }
                                int counter = Integer.parseInt(count);
                                if (warehouse.importArticle(artName, counter)==0) {
                                        JOptionPane.showInternalMessageDialog(null, "Retry, you
can't import this much", "Error",
                                                        JOptionPane.ERROR_MESSAGE);
                                }
                                else {
                                        table.setModel(new ModelData());
                                        JOptionPane.showInternalMessageDialog(null, "The article
was imported.");
                                        return;
                                }
                        }
                }

        });



        // OVERALL STATS
        getStatistics = new JButton("GET STATS");
        getStatistics.addActionListener(new ActionListener() {

                @Override
                public void actionPerformed(ActionEvent e) {
                        String[] options = { "Get list of all products", "Get list of all
articles in a group",
                                        "Cost of a group of articles", "Cost of warehouse",
"Cancel" };
                        int choice = JOptionPane.showOptionDialog(null, "What statistics do
you want to receive?",
                                        "Statistics receive", JOptionPane.DEFAULT_OPTION,
JOptionPane.QUESTION_MESSAGE, null, options,
                                        options[4]);
                        String s;
                        Group g;
                        String gName;
                        double cost;
                        switch (choice) {
                                case 0:
                                        s = warehouse.toString();
                                        JTextArea list = new JTextArea(s);
                                        JScrollPane scroll = new JScrollPane(list);
                                        scroll.setPreferredSize(new Dimension(550,550));

                                        JOptionPane.showInternalMessageDialog(null,scroll,"List
of all products",
                                                        JOptionPane.INFORMATION_MESSAGE);
                                        break;
                                case 1:
                                        gName = JOptionPane.showInputDialog("Enter the group
from which you want to get the list: ");
                                        if (gName == null) {
                                                JOptionPane.showInternalMessageDialog(null, "No
group entered, cannot find", "Error: no input",
                                                                JOptionPane.ERROR_MESSAGE);
                                                return;
                                        }
                                        g = warehouse.getGroup(gName);
                                        if (g == null) {
                                                JOptionPane.showInternalMessageDialog(null, "There
is no such Group found!",
                                                                "Error: no group found",
JOptionPane.ERROR_MESSAGE);
                                                return;
                                        }
                                        s = g.toString();
                                        JOptionPane.showInternalMessageDialog(null, s, "List of
all products of group " + g.getName(),
                                                        JOptionPane.INFORMATION_MESSAGE);
                                        break;
                                case 2:
                                        gName = JOptionPane.showInputDialog("Enter the group
from which you want to get the list: ");
                                        if (gName == null) {
                                                JOptionPane.showInternalMessageDialog(null, "No
group entered, cannot find", "Error: no input",
                                                                JOptionPane.ERROR_MESSAGE);
                                                return;
                                        }
                                        g = warehouse.getGroup(gName);
                                        if (g == null) {
                                                JOptionPane.showInternalMessageDialog(null, "There
is no such Group found!",
                                                                "Error: no group found",
JOptionPane.ERROR_MESSAGE);
                                                return;
                                        }
                                        cost = g.getTotalPrice();
                                        JOptionPane.showInternalMessageDialog(null,
                                                        "The cost of all products in " +
g.getName() + " is " + cost,
                                                        "Cost of all products of group " +
g.getName(), JOptionPane.INFORMATION_MESSAGE);
                                        break;
                                case 3:
                                        cost = warehouse.getTotalPrice();
                                        JOptionPane.showInternalMessageDialog(null, "The cost of
all products in warehouse is " + cost,
                                                        "Cost of all products in warehouse",
JOptionPane.INFORMATION_MESSAGE);
                                        break;
                                case 4:
                                        break;
                        }
                }

        });

        saveToFile = new JButton("SAVE DATA");
        saveToFile.addActionListener(new ActionListener() {

                @Override
                public void actionPerformed(ActionEvent e) {

                        String[] options = { "Save all product names", "Save warehouse
structure", "Save all product data",
                                        "Save all costs", "Cancel" };
```

```
				int choice = JOptionPane.showOptionDialog(null, "What statistics do 
you want to save?",
						"Statistics save", JOptionPane.DEFAULT_OPTION, 
JOptionPane.QUESTION_MESSAGE, null, options,
						options[4]);
				if (choice == 4) {
					return;
				}
				int returnValue = fileChooser.showOpenDialog(UserInterface.this);

				if (returnValue == JFileChooser.APPROVE_OPTION) {
					File file = fileChooser.getSelectedFile();
					System.out.println(file.getPath());

					switch (choice) {
						case 0:
							try {
								File slovar = new File(file.getPath() + 
"\\productNames.txt");
								if (!slovar.createNewFile()) {
									slovar.delete();
									slovar = new File(file.getPath() + 
"\\productNames.txt");
								}
								FileWriter myWriter = new 
FileWriter(file.getPath() + "\\productNames.txt");

								String s = warehouse.getArticleNames();

								myWriter.write(s + "\n");

								myWriter.close();
							} catch (IOException e1) {
								System.out.println("An error occurred.");
								e1.printStackTrace();
							}
							break;
						case 1:
							try {
								File slovar = new File(file.getPath() + 
"\\warehouse.txt");
								if (!slovar.createNewFile()) {
									slovar.delete();
									slovar = new File(file.getPath() + 
"\\warehouse.txt");
								}
								FileWriter myWriter = new 
FileWriter(file.getPath() + "\\warehouse.txt");

								String s = warehouse.toString();

								myWriter.write(s + "\n");

								myWriter.close();
							} catch (IOException e1) {
								System.out.println("An error occurred.");
								e1.printStackTrace();
							}
							break;
						case 2:
							try {
								File slovar = new File(file.getPath() + 
"\\articleInfo.txt");
```

```
                                                if (!slovar.createNewFile()) {
                                                      slovar.delete();
                                                      slovar = new File(file.getPath() + 
"\\articleInfo.txt");
                                                }
                                                FileWriter myWriter = new 
FileWriter(file.getPath() + "\\articleInfo.txt");

                                                String s = warehouse.getArticleInfo();

                                                myWriter.write(s + "\n");

                                                myWriter.close();
                                          } catch (IOException e1) {
                                                System.out.println("An error occurred.");
                                                e1.printStackTrace();
                                          }
                                          break;
                                    case 3:
                                          try {
                                                File slovar = new File(file.getPath() + 
"\\warehousePrice.txt");
                                                if (!slovar.createNewFile()) {
                                                      slovar.delete();
                                                      slovar = new File(file.getPath() + 
"\\warehousePrice.txt");
                                                }
                                                FileWriter myWriter = new 
FileWriter(file.getPath() + "\\warehousePrice.txt");

                                                String s = "The cost of all the products of 
warehous is " + warehouse.getTotalPrice();

                                                myWriter.write(s + "\n");

                                                myWriter.close();
                                          } catch (IOException e1) {
                                                System.out.println("An error occurred.");
                                                e1.printStackTrace();
                                          }
                                          break;
                              }

                        }
                  }
            });


            f.add(scrollPane);
            f.add(addArticle);
            f.add(editArticle);
            f.add(deleteArticle);
            f.add(addGroup);
            f.add(editGroup);
            f.add(deleteGroup);

            f.add(findArticle);
            f.add(sell);
            f.add(importing);
            f.add(getStatistics);
            f.add(saveToFile);

            f.setVisible(true);
```

```
	}
	public boolean isNumber(String str) {
		if (str == null || str.isEmpty()) return false;
		for (int i = 0; i < str.length(); i++) {
			if (!Character.isDigit(str.charAt(i))) return false;
		}
		return true;
	}
}
```

```
import java.util.ArrayList;
import java.util.Locale;

public class Warehouse {
	private static ArrayList<Group> groups;

	/**
	 * Adds group to ArrayList
	 * @param name
	 * @param description
	 * @param articles
	 */
	public void addGroup(String name, String description, ArrayList<Article> articles) {
		if (groups == null) {
			groups = new ArrayList<>();
			groups.add(new Group(name, description, articles));
		} else {
			groups.add(new Group(name, description, articles));
		}
	}
	public static ArrayList<Group> allGroups(){
		return  groups;
	}
	/**
	 * Adds group to ArrayList
	 * @param name
	 * @param description
	 */
	public void addGroup(String name, String description) {
		if (groups == null)
			groups = new ArrayList<>();
		groups.add(new Group(name, description));
	}



	/**
	 * Removes group from ArrayList
	 * @param name
	 */
	public void removeGroup(String name) {
		if (groups != null)
			for (int i = 0; i < groups.size(); i++) {
				if 
(groups.get(i).getName().toLowerCase(Locale.ROOT).equals(name.toLowerCase(Locale.ROOT)))
					groups.remove(groups.get(i));
			}
	}

	/**
	 * Gets article by name
	 * @param name
	 * @return Article
	 */
	public Article getArticle(String name) {
		for (Group group : groups) {
			if(group.getArticles() == null) {
				continue;
			}
			for (Article a : group.getArticles()) {
				if 
(a.getName().toLowerCase(Locale.ROOT).equals(name.toLowerCase(Locale.ROOT)))
					return a;
```

```
                }
            }
            return null;
        }

        public void deleteArticle(String name) {
            Group group = getGroupByArticle(name);
            group.removeArticle(name);

        }

        public Article addArticle(Group group, String name, String description,
                                                String producer, int amount, double price) {
            group.addArticle(name, description, producer, amount, price);
            return null;
        }


        /**
         * Gets group by article name
         * @param name
         * @return Group
         */
        public static Group getGroupByArticle(String name) {
            if (groups != null)
                for (Group group : groups) {
                    if(group.getArticles() == null) {
                        continue;
                    }
                    for (Article a : group.getArticles()) {
                        if
(a.getName().toLowerCase(Locale.ROOT).equals(name.toLowerCase(Locale.ROOT)))
                            return group;
                    }
                    // if(group.getArticle(name)!=null)return group.getArticle(name);
                }
            return null;
        }

        /**
         * Gets ArrayList of groups
         * @return ArrayList<Group>
         */
        public  ArrayList<Group> getGroups() {
            return groups;
        }

        /**
         * Gets group by name
         * @param name
         * @return Group
         */
        public Group getGroup(String name) {
            if (groups != null) {
                for (Group group : groups) {
                    if
(group.getName().toLowerCase(Locale.ROOT).equals(name.toLowerCase(Locale.ROOT)))
                        return group;
                }
            }
            return null;
        }
```

```java
public static ArrayList<String> getGroupNames(){
	ArrayList<String> names = null;
	for(Group g : groups){
		names.add(g.getName());
	}
	return  names;
}

/**
 * Merges all articles in a single arraylist
 * @return ArrayList<Article>
 */
public static ArrayList<Article> mergeAll() {
	ArrayList<Article> result = new ArrayList<>();
	if (groups != null) {
		for (Group group : groups) {
			if (group.getArticles() != null)
				result.addAll(group.getArticles());
		}
	}
	return result;
}

/**
 * Sells article amount
 * @param name
 * @param amount
 * @return
 */
public int sellArticle(String name, int amount) {
	// NULL EXCEPTION POSSIBLE!!!
	Group group = getGroupByArticle(name);
	Article article = group.getArticle(name);
	if (amount>article.getAmount()||amount<=0) {
		return 0;
	}
	else {
		article.setAmount(article.getAmount() - amount);
		return article.getAmount();
	}
}

/**
 * Imports article amount
 * @param name
 * @param amount
 * @return
 */
public int importArticle(String name, int amount) {
	Group group = getGroupByArticle(name);
	Article article = group.getArticle(name);
	if (amount<=0) {
		return 0;
	}
	else {
		article.setAmount(article.getAmount()+amount);
		return article.getAmount();
	}
}

/**
 * Gets total price of all articles
 * @return double
 */
public double getTotalPrice() {
	double result = 0;
	if(groups == null) {
		return 0;
	}
	for (Group group : groups) {
		result += group.getTotalPrice();
	}
	return result;
}


/**
 * Gets list of all articles
 * @return String
 */
public String listArticles() {
	StringBuilder res = new StringBuilder();
	for (Article a : mergeAll()) {
		res.append("").append(a).append(";\n");
	}
	return "ALL ARTICLES LISTED: \n" + res;
}

/**
 * Gets article names
 * @return String
 */
public String getArticleNames() {
	String s = "";
	if(groups == null) {
		return "There are no articles in this warehouse";
	}
	for (Group g : groups) {
		if(g.getArticles() == null) {
			//s+= "There are no articles in this group";
			continue;
		}
		for (Article a : g.getArticles()) {
			s += a.getName() + "\n";
		}
	}
	return s;
}

/**
 * Gets all info about articles
 * @return String
 */
public String getArticleInfo() {
	String s = "";
	if(groups == null) {
		return "There are no articles in this warehouse";
	}
	for (Group g : groups) {
		if(g.getArticles() == null) {
			continue;
		}
		for (Article a : g.getArticles()) {
			s += a.toString() + "\n";
		}
	}
```

```
		return s;
	}

	@Override
	public String toString() {
		String s = "";
		if (groups == null || groups.size() == 0) {
			return "Warehouse is empty";
		}
		for (Group g : groups) {
			s += g.toString() + "\n";
		}
		return s;
	}

}
```