Національний університет «Києво-Могилянська академія»

**Лабораторна робота №2**

Роботу виконали:

Семенюк Максим Володимирович

студент І року навчання

НАУКМА

Спеціальність: інженерія програмного забезпечення

Гущін Іван Олексійович

студент І року навчання

НАУКМА

Спеціальність: інженерія програмного забезпечення

2023

**Постановка задачі**

Необхідно автоматизувати роботу невеликого підприємства по роботі з складом.

Існує декілька груп товарів (наприклад: Продовольчі, непродовольчі...). В кожній групі товарів існують конкретні товари (наприклад: борошно, гречка ...). У кожного товару є наступні властивості - назва, опис, виробник, кількість на складі, ціна за одиницю. Група товарів містить наступні властивості - назва, опис.

Реалізувати:

1. Реалізувати графічний інтерфейс користувача
2. Збереження даних в файл/файли. Один з варіантів: Існує файл в якому знаходяться назви всіх груп товарів. Товари з кожної групи товарів знаходяться в окремому файлі.
3. Назва товару - унікальна (не може зустрічатися більше в жодній групі товарів).
4. Назва групи товарів - унікальна.
5. Реалізувати додавання/редагування/видалення групи товарів - при видаленні групи товарів, видаляти і всі товари.
6. Реалізувати додавання/редагування/видалення товару в групу товарів (мається на увазі назва, опис, виробник, ціна за одиницю).
7. Реалізувати інтерфейс додавання товару (прийшло на склад крупи гречаної - 10 штук), інтерфейс списання товару (продали крупи гречаної - 5 шт.)
8. Пошук товару.
9. Вивід статистичних даних: вивід всіх товарів з інформацією по складу, вивід усіх товарів по групі товарів з інформацією, загальна вартість товару на складі (кількість \* на ціну), загальна вартість товарів в групі товарів.
10. До роботи додати звіт про виконання роботи з описом розподілу ролей.

**Опис усіх реалізованих можливостей**

Пошук товару за патерном(шукає всі товари, що починаються на введені символи).

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 товарів.

Можливість додати, видалити, редагувати групу.

Можливість додати, видалити, редагувати товар, змінити кількість.

Можливість збереження даних про групи та товари у файли.

**Розподіл задач**

Семенюк Максим:

* Графічний інтерфейс, взаємодія з базовими класами
* Робота з файлами

Гущін Іван:

* Внутрішня структура проєкту(базові класи, їхня взаємодія)
* Рефакторинг

**Структура програми**

Зображення, що містить текст, схема, План, Креслення

Автоматично згенерований опис

**Опис методів і класів**

*/\*  
Main class to start a program  
 \*/*public class Main

*/\*  
Storage class wit all necessary methods for work with groups and statistics methods  
 \*/*

public class Storage

*/\*\* Uses constructor if instance is null, else returns reference on it  
 \* @return class reference  
 \*/*public static Storage getInstance()

*/\*\*  
 \* @return groups  
 \*/*public ArrayList<Group> getGroups()

*/\*\*  
 \* Adds a group and sorts  
 \* @param group  
 \*/*public void addGroup(Group group)

*/\*\*  
 \* Deletes a group  
 \* @param group  
 \*/*public void deleteGroup(Group group)

*/\*\*  
 \* @return price of all goods on storage  
 \*/*public String getStoragePrice()

*/\*\*  
 \* @return all storage goods in String  
 \*/*public String getAllStorageGoods()

*/\*\*  
 \* @param name - pattern to search  
 \* @return ArrayList of found goods  
 \*/*public ArrayList<Good> findGood(String name)

*/\*\*  
 \* Updates information in group files  
 \*/*public static void updateFiles()

*/\*\*  
 \* Opens and updates a file  
 \*  
 \* @param name of a group and a file  
 \* @throws IOException  
 \*/*private static void updateFile(String name) throws IOException

*/\*\*  
 \* reads info from files  
 \*/*public void readFiles()

*/\*\*  
 \* reads info from each file  
 \* @param file  
 \*/*private void readFile(File file)

*/\*\*  
 \* Adds a good according to read info  
 \* @param goodString  
 \* @param goods  
 \* @param groupName  
 \*/*private void createNewGood(String goodString, ArrayList<Good> goods, String groupName)

*\*\*  
 \* Finds group index  
 \*  
 \* @param name name of a group  
 \* @return index of a group in list of groups  
 \*/*public static int findGroup(String name)

*/\*\*  
 \* sorts groups by names  
 \*/*private void sortGroups()

*/\*\*  
 \* Searches and deletes a good  
 \* @param name of a good  
 \*/*public void deleteGood(String name)

*/\*  
Group class. Contains all necessary fields and methods  
 \*/*

public class Group

*/\*\*  
 \* Creates a new group  
 \*  
 \* @param name of a group  
 \*/*public Group(String name)

*/\*\*  
 \* Creates a new group when program starts(for initially created groups)  
 \*  
 \* @param name of a group  
 \* @param goods array of goods  
 \*/*public Group(String name, ArrayList<Good> goods)

*//Setters and getters*

public String getName()

public void setName(String name)

public ArrayList<Good> getGoods()

*/\*\*  
 \* @return all goods in group  
 \*/*public String getAllGroupGoods()

*/\*\* Adds a new good  
 \* @param good - good to be added  
 \*/*public void addGood(Good good)

*/\*\* Deletes a good  
 \* @param good - good to be deleted  
 \*/*public void deleteGood(Good good)

*/\*\*  
 \* @return price of all goods in group  
 \*/*public float getGroupPrice()

*/\*\*  
 \*  
 \* @param name - pattern for search  
 \* @return ArrayList of found goods  
 \*/*public ArrayList<Good> findGood(String name)

*/\*\*  
 \* sorts goods by name  
 \*/*private void sortGoods()

*/\*  
Good class. Contains all necessary fields  
 \*/*public class Good

*/\*\*  
 \* Full constructor  
 \* @param group  
 \* @param name  
 \* @param description  
 \* @param manufacturer  
 \* @param amount  
 \* @param price  
 \*/*public Good(String group, String name, String description, String manufacturer, int amount, float price)

*//Getters and setters*

public String getName()

public String getDescription()

public String getManufacturer()

public int getAmount()

public float getPrice()

public void setName(String name)

public void setDescription(String description)

public void setManufacturer(String manufacturer)

public void setAmount(int amount)

public void setPrice(float price)

public String getGroup()

public void setGroup(String group)

*/\*\*  
 \* @param amount to add to a good  
 \*/*public void addAmount(int amount)

*/\*\*  
 \* @param amount of goods to reduce  
 \*/*public void reduceAmount(int amount)

*/\*  
Main menu UI  
 \*/*

public class MainMenu extends JFrame implements ActionListener

*/\*  
UI for searching for a good  
 \*/*

public class FindGoodUI extends OutputUI implements ActionListener

*/\*  
UI for statistics methods  
 \*/*

public class StatisticsUI extends JFrame implements ActionListener

*/\*  
UI to show statistics info  
 \*/*

public class OutputUI extends JFrame implements ActionListener

*/\*  
UI for specific statistics method  
 \*/*

public class OutputUIGroups extends JFrame implements ActionListener

*/\*  
UI for working with groups  
 \*/*

public class WorkWithGroupUI extends JFrame implements ActionListener

*/\*  
UI for adding a group  
 \*/*

public class AddGroupUI extends JFrame implements ActionListener

*/\*  
UI for deleting a group  
 \*/*

public class DeleteGroupUI extends JFrame implements ActionListener

*/\*  
UI for editing a group  
 \*/*

public class EditGroupUI extends JFrame implements ActionListener

*/\*  
UI for working with goods  
 \*/*

public class WorkWithProductUI extends JFrame implements ActionListener

*/\*  
UI for adding a good  
 \*/*

public class AddGoodUI extends JFrame implements ActionListener

*/\*  
UI for deleting a good  
 \*/*

public class DeleteGoodUI extends JFrame implements ActionListener

*/\*  
UI for editing a good  
 \*/*

public class EditGoodUI extends JFrame implements ActionListener

*/\*  
UI for reducing amount of a good  
 \*/*

public class RedQuantityOfProductUI extends JFrame implements ActionListener

*/\*  
UI for adding amount of a good  
 \*/*

public class AddQuantityOfProductUI extends JFrame implements ActionListener

**Проблеми, які виникали та шляхи їх вирішення**

Були проблеми із доступом до класу Storage і, відповідно, до груп і товарів. Для вирішення проблеми ми переписали цей клас в singltone і отримували посилання на нього звідки завгодно.

Наступною проблемою була реалізація пошуку товару в реальному часі за словом при роботі з товарами буз натискань будь-яких кнопок. Рішенням було використання потоків для синхронного оновлення списку товарів при введенні певного слова в поле пошуку.

Останньою проблемою була проблема зі збереженням даних у файли( файли з відносоно великим ім’ям не зберігалися нормально). Для рішення цього питання було вирішено зберігати дані в файли з розширенням .txt.

**Програмний код**

*/\*  
Main class to start a program  
 \*/*public class Main {  
 private static Storage *storage*;  
 private static MainMenu *mainMenu*;  
  
 public static void main(String[] args) {  
 *storage* = Storage.*getInstance*();  
 *storage*.readFiles();  
 *mainMenu* = new MainMenu();  
 }  
}

*/\*  
Storage class wit all necessary methods for work with groups and statistacs methods  
 \*/*import java.io.\*;  
import java.nio.file.Files;  
import java.nio.file.Path;  
import java.nio.file.Paths;  
import java.util.ArrayList;  
import java.util.Arrays;  
import java.util.StringTokenizer;  
  
import static java.util.Comparator.*comparing*;  
  
public class Storage {  
 private static Storage *instance*;  
 private static ArrayList<Group> *groups* = new ArrayList<>();  
 private static BufferedWriter *bw*;  
  
 private Storage() {  
 }  
  
 */\*\* Uses constructor if instance is null, else returns reference on it  
 \* @return class reference  
 \*/* public static Storage getInstance() {  
 *// Якщо екземпляр ще не існує, створюємо його* if (*instance* == null) {  
 *instance* = new Storage();  
 }  
 *// Повертаємо єдиний екземпляр класу* return *instance*;  
 }  
  
 */\*\*  
 \* @return groups  
 \*/* public ArrayList<Group> getGroups() {  
 return *groups*;  
 }  
  
 */\*\*  
 \* Adds a group and sorts  
 \* @param group  
 \*/* public void addGroup(Group group) {  
 *groups*.add(group);  
 sortGroups();  
 }  
  
 */\*\*  
 \* Deletes a group  
 \* @param group  
 \*/* public void deleteGroup(Group group) {  
 *groups*.remove(group);  
 }  
  
 */\*\*  
 \* @return price of all goods on storage  
 \*/* public String getStoragePrice() {  
 StringBuilder text = new StringBuilder();  
 float sum = 0;  
 int amount = 0;  
 for (Group group : *groups*) {  
 sum += group.getGroupPrice();  
 amount += group.getGoods().size();  
 }  
 text.append("Кількість товарів на складі: " + amount + "\nЦіна товарів: " + sum);  
 return text.toString();  
 }  
  
 */\*\*  
 \* @return all storage goods in String  
 \*/* public String getAllStorageGoods() {  
 StringBuilder text = new StringBuilder();  
 for (Group group : *groups*) {  
 text.append(group.getAllGroupGoods());  
 }  
 return text.toString();  
 }  
  
 */\*\*  
 \* @param name - pattern to search  
 \* @return ArrayList of found goods  
 \*/* public ArrayList<Good> findGood(String name) {  
 ArrayList<Good> found = new ArrayList<>();  
 for (Group group : *groups*) {  
 found.addAll(group.findGood(name));  
 }  
 return found;  
 }  
  
 */\*\*  
 \* Updates information in group files  
 \*/* public static void updateFiles() {  
 File folder = new File("lab2/Groups");  
 if (folder.isDirectory()) {  
 File[] files = folder.listFiles();  
 if (files != null) {  
 for (File file : files) {  
 file.delete();  
 }  
 }  
 }  
 folder.delete();  
 String folderPathStr = "lab2/Groups";  
  
 *// Create a Path object representing the folder* Path folderPath = Paths.*get*(folderPathStr);  
 try {  
 *// Create the folder/directory* Files.*createDirectory*(folderPath);  
 } catch (IOException e) {  
 System.*err*.println("Failed to create folder: " + e.getMessage());  
 }  
 for (Group group : *groups*) {  
 try {  
 *updateFile*(group.getName());  
 } catch (IOException e) {  
 throw new RuntimeException(e);  
 }  
 }  
 }  
  
 */\*\*  
 \* Opens and updates a file  
 \*  
 \* @param name of a group and a file  
 \* @throws IOException  
 \*/* private static void updateFile(String name) throws IOException {  
 try {  
 *bw* = new BufferedWriter(new FileWriter("lab2/Groups/" + name + ".txt"));  
 } catch (IOException e) {  
 System.*out*.println("File not found");  
 }  
 for (Good good : (*groups*.get(*findGroup*(name))).getGoods()) {  
 *bw*.write(good + "\n");  
 }  
 *bw*.close();  
 }  
  
 */\*\*  
 \* reads info from files  
 \*/* public void readFiles() {  
 File folder = new File("lab2/Groups");  
 File[] fileList = folder.listFiles();  
 for (File file : fileList) {  
 readFile(file);  
 }  
 sortGroups();  
 }  
  
 */\*\*  
 \* reads info from each file  
 \* @param file  
 \*/* private void readFile(File file) {  
 try {  
 BufferedReader br = new BufferedReader(new FileReader(file.getPath()));  
 String groupName = file.getName();  
 groupName = groupName.replaceAll(".txt", "");  
 ArrayList<Good> goods = new ArrayList<>();  
 String goodString;  
 do {  
 goodString = br.readLine();  
 if (goodString != null && !goodString.isBlank()) {  
 createNewGood(goodString, goods, groupName);  
 }  
  
 } while (goodString != null);  
 *groups*.add(new Group(groupName, goods));  
 br.close();  
 } catch (FileNotFoundException e) {  
 System.*out*.println("File not found");  
 } catch (IOException e) {  
 System.*out*.println("Error in reading");  
 }  
 }  
  
 */\*\*  
 \* Adds a good according to read info  
 \* @param goodString  
 \* @param goods  
 \* @param groupName  
 \*/* private void createNewGood(String goodString, ArrayList<Good> goods, String groupName) {  
 StringTokenizer st = new StringTokenizer(goodString, "|");  
  
 String name = st.nextToken();  
  
 StringTokenizer desSt = new StringTokenizer(st.nextToken(), ":");  
 String descr = desSt.nextToken();  
 descr = desSt.nextToken();  
  
 StringTokenizer mnfSt = new StringTokenizer(st.nextToken(), ":");  
 String mnf = mnfSt.nextToken();  
 mnf = mnfSt.nextToken();  
  
 StringTokenizer amountSt = new StringTokenizer(st.nextToken(), ":");  
 String amountStr = amountSt.nextToken();  
 amountStr = amountSt.nextToken();  
 int amount = Integer.*parseInt*(amountStr);  
  
 StringTokenizer priceSt = new StringTokenizer(st.nextToken(), ":");  
 String priceStr = priceSt.nextToken();  
 priceStr = priceSt.nextToken();  
 float price = Float.*parseFloat*(priceStr);  
  
 goods.add(new Good(groupName, name, descr, mnf, amount, price));  
 }  
  
 */\*\*  
 \* Finds group index  
 \*  
 \* @param name name of a group  
 \* @return index of a group in list of groups  
 \*/* public static int findGroup(String name) {  
 for (Group group : *groups*) {  
 if (group.getName().equals(name)) return *groups*.indexOf(group);  
 }  
 return -1;  
 }  
  
 */\*\*  
 \* sorts groups by names  
 \*/* private void sortGroups() {  
 *groups*.sort(*comparing*(Group::getName));  
 }  
  
 */\*\*  
 \* Searches and deletes a good  
 \* @param name of a good  
 \*/* public void deleteGood(String name) {  
 for (Group group : *groups*) {  
 for (Good good : group.getGoods()) {  
 if (good.getName().equals(name)) {  
 group.deleteGood(good);  
 return;  
 }  
 }  
 }  
 }  
}

*/\*  
Group class. Contains all necessary fields and methods  
 \*/*import java.util.ArrayList;  
  
import static java.util.Comparator.*comparing*;  
  
public class Group {  
 private ArrayList<Good> goods;  
 private String name;  
  
 */\*\*  
 \* Creates a new group  
 \*  
 \* @param name of a group  
 \*/* public Group(String name) {  
 this.name = name;  
 goods = new ArrayList<>(1);  
 }  
  
 */\*\*  
 \* Creates a new group when program starts(for initially created groups)  
 \*  
 \* @param name of a group  
 \* @param goods array of goods  
 \*/* public Group(String name, ArrayList<Good> goods) {  
 this.name = name;  
 this.goods = goods;  
 sortGoods();  
 }  
 *//Setters and getters* public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 for (Good good : goods) {  
 good.setGroup(name);  
 }  
 }  
  
 public ArrayList<Good> getGoods() {  
 return goods;  
 }  
  
 */\*\*  
 \* @return all goods in group  
 \*/* public String getAllGroupGoods() {  
 StringBuilder text = new StringBuilder();  
 text.append("Група " + name + ":\n");  
 for (Good good : goods) {  
 text.append(good + "\n");  
 }  
 return text.toString();  
 }  
  
 */\*\* Adds a new good  
 \* @param good - good to be added  
 \*/* public void addGood(Good good) {  
 goods.add(good);  
 }  
  
 */\*\* Deletes a good  
 \* @param good - good to be deleted  
 \*/* public void deleteGood(Good good) {  
 goods.remove(good);  
 }  
  
 */\*\*  
 \* @return price of all goods in group  
 \*/* public float getGroupPrice() {  
 float sum = 0;  
 for (Good good : goods) {  
 sum += good.getAmount() \* good.getPrice();  
 }  
 return sum;  
 }  
  
 */\*\*  
 \*  
 \* @param name - pattern for search  
 \* @return ArrayList of found goods  
 \*/* public ArrayList<Good> findGood(String name) {  
 ArrayList<Good> found = new ArrayList<>();  
 for (Good good : goods) {  
 if (good.getName().matches(name + "([0-9\_A-Za-zА-Яа-я]\*\\s\*){3}")) {  
 found.add(good);  
 }  
 }  
 return found;  
 }  
  
 */\*\*  
 \* sorts goods by name  
 \*/* private void sortGoods() {  
 goods.sort(*comparing*(Good::getName));  
 }  
  
 @Override  
 public String toString() {  
 return name + " containing " + goods.size() + " goods";  
 }  
}

*/\*  
Good class. Contains all necessary fields  
 \*/*public class Good {  
 private String name;  
 private String description;  
 private String manufacturer;  
 private int amount;  
 private float price;  
 private String group;  
  
 */\*\*  
 \* Full constructor  
 \* @param group  
 \* @param name  
 \* @param description  
 \* @param manufacturer  
 \* @param amount  
 \* @param price  
 \*/* public Good(String group, String name, String description, String manufacturer, int amount, float price) {  
 this.group = group;  
 this.name = name;  
 this.description = description;  
 this.manufacturer = manufacturer;  
 this.amount = amount;  
 this.price = price;  
 }  
 *//Getters and setters* public String getName() {  
 return name;  
 }  
 public String getDescription() {  
 return description;  
 }  
 public String getManufacturer() {  
 return manufacturer;  
 }  
 public int getAmount() {  
 return amount;  
 }  
  
 public float getPrice() {  
 return price;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public void setDescription(String description) {  
 this.description = description;  
 }  
  
 public void setManufacturer(String manufacturer) {  
 this.manufacturer = manufacturer;  
 }  
  
 public void setAmount(int amount) {  
 this.amount = amount;  
 }  
  
 public void setPrice(float price) {  
 this.price = price;  
 }  
  
 */\*\*  
 \* @param amount to add to a good  
 \*/* public void addAmount(int amount) {  
 this.amount += amount;  
 }  
  
 */\*\*  
 \* @param amount of goods to reduce  
 \*/* public void reduceAmount(int amount) {  
 this.amount -= amount;  
 }  
  
 public String getGroup() {  
 return group;  
 }  
  
 public void setGroup(String group) {  
 this.group = group;  
 }  
  
 @Override  
 public String toString() {  
 return name + " | опис: " + description + " | виробник: " + manufacturer + " | кількість на складі: " + amount + " | ціна: " + price;  
 }  
}

*/\*  
Main menu UI  
 \*/*import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
  
public class MainMenu extends JFrame implements ActionListener {  
 JButton workWithGoods;  
 JButton workWithGroups;  
 JButton statistics;  
 JButton saveData;  
 JButton search;  
 JButton closeProgram;  
 WorkWithProductUI prodUI;  
 WorkWithGroupUI groupUI;  
 StatisticsUI statUI;  
 OutputUI outputUI;  
 FindGoodUI findGoodUI;  
 JFrame mainFrame;  
  
 public MainMenu() {  
 super("Програма для роботи з складом");  
 this.setSize(700, 500);  
 this.setLayout(new GridLayout(1, 2));  
 this.setDefaultCloseOperation(*EXIT\_ON\_CLOSE*);  
 this.setIconImage(new ImageIcon("lab2/Images/wareHouseIcon.png").getImage());  
 init();  
 this.setVisible(true);  
 this.setLocationRelativeTo(null);  
 }  
  
 private void init() {  
 *// add photo to left part of window* addPhoto();  
 *//add menuPanel* addMenuPanel();  
  
 }  
  
 private void addPhoto() {  
 ImageIcon wareHousePicture = new ImageIcon("lab2/Images/WareHouse.png");  
 *// Create a JLabel to display the image* JLabel wareHouseLabel = new JLabel();  
 wareHouseLabel.setIcon(wareHousePicture);  
 wareHouseLabel.setHorizontalAlignment(JLabel.*CENTER*);  
 wareHouseLabel.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 4));  
 this.add(wareHouseLabel);  
 }  
  
 private void addMenuPanel() {  
 JPanel menuPanel = new JPanel();  
 menuPanel.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 4));  
 menuPanel.setLayout(new BorderLayout());  
 *//add label* addMenuLabel(menuPanel);  
 *//add buttons* setAllButtons(menuPanel);  
 this.add(menuPanel);  
 }  
  
 private void addMenuLabel(JPanel menuPanel) {  
 JLabel menuLabel = new JLabel("Меню");  
 menuLabel.setFont(new Font("Default", Font.*BOLD*, 50));  
 menuLabel.setHorizontalAlignment(JLabel.*CENTER*);  
 menuLabel.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 2));  
 menuPanel.add(menuLabel, "North");  
  
 }  
  
 private void setAllButtons(JPanel menuPanel) {  
 JPanel centerButtonPanel = new JPanel();  
 centerButtonPanel.setLayout(new GridLayout(4, 1));  
  
 JPanel workWithGoodsPanel = new JPanel();  
 workWithGoodsPanel.setLayout(new FlowLayout());  
 workWithGoods = new JButton("Робота з товарами");  
 workWithGoods.addActionListener(this);  
 workWithGoods.setPreferredSize(new Dimension(220, 70));  
 workWithGoods.setFont(new Font("Default", Font.*BOLD*, 17));  
 workWithGoodsPanel.add(workWithGoods);  
  
 JPanel workWithGroupsPanel = new JPanel();  
 workWithGroupsPanel.setLayout(new FlowLayout());  
 workWithGroups = new JButton("Робота з групами");  
 workWithGroups.addActionListener(this);  
 workWithGroups.setPreferredSize(new Dimension(220, 70));  
 workWithGroups.setFont(new Font("Default", Font.*BOLD*, 17));  
 workWithGroupsPanel.add(workWithGroups);  
  
 JPanel statisticsPanel = new JPanel();  
 statisticsPanel.setLayout(new FlowLayout());  
 statistics = new JButton("Вивести статистику");  
 statistics.addActionListener(this);  
 statistics.setPreferredSize(new Dimension(220, 70));  
 statistics.setFont(new Font("Default", Font.*BOLD*, 17));  
 statisticsPanel.add(statistics);  
  
 JPanel searchPanel = new JPanel();  
 searchPanel.setLayout(new FlowLayout());  
 search = new JButton("Пошук товарів");  
 search.addActionListener(this);  
 search.setPreferredSize(new Dimension(220, 70));  
 search.setFont(new Font("Default", Font.*BOLD*, 17));  
 searchPanel.add(search);  
  
 centerButtonPanel.add(workWithGoodsPanel, "North");  
 centerButtonPanel.add(workWithGroupsPanel, "Center");  
 centerButtonPanel.add(statisticsPanel, "South");  
 centerButtonPanel.add(searchPanel, "North");  
 menuPanel.add(centerButtonPanel);  
  
 JPanel lowButtonPanel = new JPanel();  
 lowButtonPanel.setLayout(new BorderLayout());  
 saveData = new JButton("<html>Зберегти дані<br> в файл</html>");  
 saveData.addActionListener(this);  
 saveData.setPreferredSize(new Dimension(160, 70));  
 saveData.setFont(new Font("Default", Font.*BOLD*, 16));  
  
 closeProgram = new JButton("<html>Закрити<br> програму</html>");  
 closeProgram.addActionListener(this);  
 closeProgram.setPreferredSize(new Dimension(160, 70));  
 closeProgram.setFont(new Font("Default", Font.*BOLD*, 16));  
  
 lowButtonPanel.add(saveData, "West");  
 lowButtonPanel.add(closeProgram, "East");  
 menuPanel.add(lowButtonPanel, "South");  
 setWindows();  
 }  
  
 private void setWindows() {  
 prodUI = new WorkWithProductUI();  
 prodUI.setMainMenu(this);  
  
 groupUI = new WorkWithGroupUI();  
 groupUI.setMainMenu(this);  
  
 outputUI = new OutputUI();  
 outputUI.setStatisticsUI(statUI);  
 outputUI.setMainMenu(this);  
  
 statUI = new StatisticsUI();  
 statUI.setMainMenu(this);  
 statUI.setOutputUI(outputUI);  
  
 findGoodUI = new FindGoodUI();  
 findGoodUI.setMainMenu(this);  
 }  
  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 this.setVisible(false);  
 if (e.getSource().equals(workWithGoods)) {  
 prodUI.setVisible(true);  
 } else if (e.getSource().equals(workWithGroups)) {  
 groupUI.setVisible(true);  
 } else if (e.getSource().equals(statistics)) {  
 statUI.setVisible(true);  
 } else if (e.getSource().equals(search)) {  
 findGoodUI.setVisible(true);  
 } else if (e.getSource().equals(saveData)) {  
 Storage.*updateFiles*();  
 JOptionPane.*showMessageDialog*(null, "Дані успішно збережено.", "Успіх", JOptionPane.*INFORMATION\_MESSAGE*);  
 returned();  
 } else {  
 int option = JOptionPane.*showConfirmDialog*(null, "Ви точно хочете закрити програму?\nНезбережені дані буде назавжди втрачено", "!!!", JOptionPane.*YES\_NO\_OPTION*);  
 if (option == JOptionPane.*YES\_OPTION*) {  
 System.*exit*(0);  
 } else {  
 returned();  
 }  
 }  
 }  
  
 */\*\*  
 \* Used by other UI classes to return to main menu  
 \*/* public void returned() {  
 this.setVisible(true);  
 }  
}

*/\*  
UI for searching for a good  
 \*/*import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
  
public class FindGoodUI extends OutputUI implements ActionListener {  
 Storage storage;  
 JTextField textField;  
 JButton enter;  
  
 public FindGoodUI() {  
 super();  
 storage = Storage.*getInstance*();  
 setInputUI();  
 }  
  
 private void setInputUI() {  
 textField = new JTextField();  
 textField.setPreferredSize(new Dimension(220, 60));  
 textField.setFont(new Font("Default", Font.*PLAIN*, 17));  
 lowerPanel.add(textField);  
  
 enter = new JButton();  
 createButtonWithAndAddToPanel(enter, "Знайти", lowerPanel);  
 }  
  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 if (e.getSource().equals(back)) {  
 this.setVisible(false);  
 menu.returned();  
 } else {  
 String findName = textField.getText();  
 StringBuilder text = new StringBuilder();  
 if (findName.isEmpty()) {  
 text.append("Не знайдено жодного товару");  
 } else {  
 for (Good good : storage.findGood(findName)) {  
 text.append(good + "\n");  
 }  
 if (text.isEmpty()) {  
 text.append("Не знайдено жодного товару");  
 } else {  
 text.insert(0, "Знайдено:\n");  
 }  
 }  
  
 output.setText(text.toString());  
 }  
 }  
}

*/\*  
UI for statistics methods  
 \*/*import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
  
public class StatisticsUI extends JFrame implements ActionListener {  
 OutputUI output;  
 Storage storage;  
 MainMenu menu;  
 JButton showAllStorageGoods;  
 JButton showAllGroupGoods;  
 JButton showStoragePrice;  
 JButton showGroupPrice;  
 JButton back;  
  
 public StatisticsUI() {  
 super("Статистика");  
 this.setSize(700, 500);  
 this.setLayout(new BorderLayout());  
 this.setDefaultCloseOperation(*EXIT\_ON\_CLOSE*);  
 this.setLocationRelativeTo(null);  
 this.setIconImage(new ImageIcon("lab2/Images/wareHouseIcon.png").getImage());  
  
 setWindow();  
  
 this.setVisible(false);  
  
 storage = Storage.*getInstance*();  
 }  
  
 private void setWindow() {  
 *//create upper part label with picture* addUpperPart();  
 *//create central part with buttons to deal with product* addCentralPart();  
 *//create lower part with back button* addLowerPart();  
 }  
  
 private void addUpperPart() {  
 JPanel upperPart = new JPanel(new BorderLayout());  
 upperPart.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 4));  
 ImageIcon productPicture = new ImageIcon("lab2/Images/stonks.jpg");*//add picture* JLabel productLabel = new JLabel(); *// Create a JLabel to display the image* productLabel.setIcon(productPicture);  
 productLabel.setHorizontalAlignment(JLabel.*CENTER*);  
 productLabel.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 2));  
 upperPart.add(productLabel, "West");  
  
 JLabel workWithProductLabel = new JLabel("Статистика"); *// Create a JLabel to display the name* workWithProductLabel.setHorizontalAlignment(JLabel.*CENTER*);  
 workWithProductLabel.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 2));  
 workWithProductLabel.setFont(new Font("Default", Font.*BOLD*, 17));  
 upperPart.add(workWithProductLabel, "Center");  
  
 this.add(upperPart, "North");  
 }  
  
 private void addCentralPart() {  
 JPanel centralPart = new JPanel(new GridLayout(0, 1));  
 centralPart.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 2));  
  
 JPanel centralLeft = new JPanel(new GridLayout(4, 0));  
  
 showAllStorageGoods = new JButton();  
 createButtonWithAndAddToPanel(showAllStorageGoods, "Вивід усіх товарів", centralLeft);  
  
 showAllGroupGoods = new JButton();  
 createButtonWithAndAddToPanel(showAllGroupGoods, "Вивід усіх товарів по групі", centralLeft);  
  
 showStoragePrice = new JButton();  
 createButtonWithAndAddToPanel(showStoragePrice, "Загальна вартість товарів на складі", centralLeft);  
  
 showGroupPrice = new JButton();  
 createButtonWithAndAddToPanel(showGroupPrice, "Загальна вартість товарів у групі товарів", centralLeft);  
  
 centralPart.add(centralLeft);  
 this.add(centralPart, "Center");  
 }  
  
 private void addLowerPart() {  
 JPanel lowerPanel = new JPanel(new FlowLayout());  
 lowerPanel.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 4));  
 back = new JButton();  
 createButtonWithAndAddToPanel(back, "Повернутися назад", lowerPanel);  
 lowerPanel.setBackground(Color.*GRAY*);  
 this.add(lowerPanel, "South");  
 }  
  
 public void setMainMenu(MainMenu menu) {  
 this.menu = menu;  
 }  
  
 public void setOutputUI(OutputUI output) {  
 this.output = output;  
 this.output.setStatisticsUI(this);  
 }  
  
 private void createButtonWithAndAddToPanel(JButton button, String buttonLabel, JPanel originPanel) {  
 JPanel buttonPanel = new JPanel();  
 buttonPanel.setLayout(new FlowLayout());  
 if (!button.equals(back)) {  
 buttonPanel.setBackground(Color.*LIGHT\_GRAY*);  
 button.setPreferredSize(new Dimension(400, 55));  
 } else {  
 buttonPanel.setBackground(Color.*GRAY*);  
 button.setPreferredSize(new Dimension(220, 60));  
 }  
 button.setText(buttonLabel);  
 button.setHorizontalAlignment(JButton.*CENTER*);  
 button.addActionListener(this);  
  
 button.setFont(new Font("Default", Font.*BOLD*, 17));  
 buttonPanel.add(button);  
 originPanel.add(buttonPanel);  
 }  
  
 public void returned() {  
 this.setVisible(true);  
 }  
  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 this.setVisible(false);  
 if (e.getSource().equals(showAllStorageGoods)) {  
 this.setVisible(false);  
 output.setText(storage.getAllStorageGoods());  
 } else if (e.getSource().equals(showAllGroupGoods)) {  
 OutputUIGroups outGr = new OutputUIGroups(0);  
 outGr.setStatisticsUI(this);  
 outGr.setVisible(true);  
 } else if (e.getSource().equals(showStoragePrice)) {  
 this.setVisible(false);  
 output.setText(storage.getStoragePrice());  
 } else if (e.getSource().equals(showGroupPrice)) {  
 OutputUIGroups outGr = new OutputUIGroups(1);  
 outGr.setStatisticsUI(this);  
 outGr.setVisible(true);  
 } else {  
 this.setVisible(false);  
 menu.returned();  
 }  
 }  
}

*/\*  
UI to show statistics info  
 \*/*import javax.swing.\*;  
import javax.swing.text.DefaultCaret;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
  
public class OutputUI extends JFrame implements ActionListener {  
 protected MainMenu menu;  
 StatisticsUI stats;  
 JPanel lowerPanel;  
 JButton back;  
 JTextArea output;  
 JScrollPane scroll;  
  
 public OutputUI() {  
 super("Output");  
 this.setSize(700, 500);  
 this.setLayout(new BorderLayout());  
 this.setDefaultCloseOperation(*EXIT\_ON\_CLOSE*);  
 this.setLocationRelativeTo(null);  
 this.setIconImage(new ImageIcon("lab2/Images/wareHouseIcon.png").getImage());  
  
 setOutput();  
  
 this.setVisible(false);  
 }  
  
 public void setStatisticsUI(StatisticsUI stats) {  
 this.stats = stats;  
 }  
  
 public void setMainMenu(MainMenu menu) {  
 this.menu = menu;  
 }  
  
 public void setText(String text) {  
 this.setVisible(true);  
 output.setText(text);  
 }  
  
 private void setOutput() {  
 *//set text area* setUpperPanel();  
 *//set panel with button to return* setLowerPanel();  
 }  
  
 private void setUpperPanel() {  
 JPanel upperPanel = new JPanel(new GridLayout(0, 1));  
  
 setOutputTextArea(); *//set textArea* scroll = new JScrollPane(output);  
  
 upperPanel.add(scroll);  
  
 this.add(upperPanel, "Center");  
 }  
  
 private void setOutputTextArea() {  
 output = new JTextArea();  
 output.setEditable(false); *//blocks editing text* output.setFont(new Font("Default", Font.*PLAIN*, 17)); *//sets font* output.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 2)); *//sets border* output.setCaret(new DefaultCaret() { *//sets invisible caret* @Override  
 public void paint(Graphics g) {  
 }  
 });  
 }  
  
 private void setLowerPanel() {  
 lowerPanel = new JPanel(new FlowLayout());  
 lowerPanel.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 4));  
 lowerPanel.setBackground(Color.*GRAY*);  
  
 back = new JButton();  
 createButtonWithAndAddToPanel(back, "Повернутися назад", lowerPanel);  
  
 this.add(lowerPanel, "South");  
 }  
  
 protected void createButtonWithAndAddToPanel(JButton button, String buttonLabel, JPanel originPanel) {  
 JPanel buttonPanel = new JPanel();  
 buttonPanel.setLayout(new FlowLayout());  
 buttonPanel.setBackground(Color.*GRAY*);  
 button.setPreferredSize(new Dimension(210, 60));  
 button.setText(buttonLabel);  
 button.setHorizontalAlignment(JButton.*CENTER*);  
 button.addActionListener(this);  
  
 button.setFont(new Font("Default", Font.*BOLD*, 17));  
 buttonPanel.add(button);  
 originPanel.add(buttonPanel);  
 }  
  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 this.setVisible(false);  
 stats.returned();  
 }  
}

*/\*  
UI for specific statistics method  
 \*/*import javax.swing.\*;  
import javax.swing.text.DefaultCaret;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.util.ArrayList;  
import java.util.Scanner;  
  
public class OutputUIGroups extends JFrame implements ActionListener {  
 protected MainMenu menu;  
 StatisticsUI stats;  
 JPanel lowerPanel;  
 Storage storage;  
 ArrayList<Group> groupsList;  
 JComboBox groups;  
 JButton back;  
 JButton print;  
 JTextArea output;  
 JScrollPane scroll;  
 int mode;  
  
 public OutputUIGroups(int mode) {  
 super("Output");  
 this.setSize(700, 500);  
 this.setLayout(new BorderLayout());  
 this.setDefaultCloseOperation(*EXIT\_ON\_CLOSE*);  
 this.setLocationRelativeTo(null);  
 this.setIconImage(new ImageIcon("lab2/Images/wareHouseIcon.png").getImage());  
  
 this.mode = mode;  
 storage = Storage.*getInstance*();  
 groupsList = storage.getGroups();  
 setOutput();  
  
 this.setVisible(false);  
 }  
  
 public void setStatisticsUI(StatisticsUI stats) {  
 this.stats = stats;  
 }  
  
 public void setMainMenu(MainMenu menu) {  
 this.menu = menu;  
 }  
  
 public void setText(String text) {  
 this.setVisible(true);  
 output.setText(text);  
 }  
  
 private void setOutput() {  
 *//set text area* setUpperPanel();  
 *//set panel with button to return* setLowerPanel();  
 }  
  
 private void setUpperPanel() {  
 JPanel upperPanel = new JPanel(new GridLayout(0, 1));  
  
 setOutputTextArea(); *//set textArea* scroll = new JScrollPane(output);  
  
 upperPanel.add(scroll);  
  
 this.add(upperPanel, "Center");  
 }  
  
 private void setOutputTextArea() {  
 output = new JTextArea();  
 output.setEditable(false); *//blocks editing text* output.setFont(new Font("Default", Font.*PLAIN*, 17)); *//sets font* output.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 2)); *//sets border* output.setCaret(new DefaultCaret() { *//sets invisible caret* @Override  
 public void paint(Graphics g) {  
 }  
 });  
 }  
  
 private void setLowerPanel() {  
 lowerPanel = new JPanel(new FlowLayout());  
 lowerPanel.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 4));  
 lowerPanel.setBackground(Color.*GRAY*);  
  
 back = new JButton();  
 createButtonWithAndAddToPanel(back, "Повернутися назад", lowerPanel);  
  
 groups = new JComboBox();  
 createComboBoxWithLabelWithAndAddToPanel(groups, "Група:", lowerPanel);  
 print = new JButton();  
 createButtonWithAndAddToPanel(print, "Вивести", lowerPanel);  
 this.add(lowerPanel, "South");  
 }  
  
 protected void createButtonWithAndAddToPanel(JButton button, String buttonLabel, JPanel originPanel) {  
 JPanel buttonPanel = new JPanel();  
 buttonPanel.setLayout(new FlowLayout());  
 buttonPanel.setBackground(Color.*GRAY*);  
 button.setPreferredSize(new Dimension(210, 60));  
 button.setText(buttonLabel);  
 button.setHorizontalAlignment(JButton.*CENTER*);  
 button.addActionListener(this);  
  
 button.setFont(new Font("Default", Font.*BOLD*, 17));  
 buttonPanel.add(button);  
 originPanel.add(buttonPanel);  
 }  
  
 private void createComboBoxWithLabelWithAndAddToPanel(JComboBox comboBox, String labelText, JPanel originPanel) {  
 JPanel labelAndTextPanel = new JPanel(new BorderLayout());  
 labelAndTextPanel.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 2));  
 labelAndTextPanel.setBackground(Color.*GRAY*);  
 JLabel label = new JLabel(labelText);  
  
 labelAndTextPanel.add(label, "North");  
  
 for (int i = 0; i < groupsList.size(); i++) {  
 comboBox.addItem(groupsList.get(i).getName());  
 }  
  
 labelAndTextPanel.add(comboBox, "Center");  
  
 originPanel.add(labelAndTextPanel);  
 }  
  
 @Override  
 public void actionPerformed(ActionEvent e) {  
  
 this.setVisible(false);  
 if (e.getSource().equals(print)) {  
 int groupNumb = groups.getSelectedIndex();  
 Group gr = groupsList.get(groupNumb);  
 if (mode == 0) {  
 setText(gr.getAllGroupGoods());  
 } else {  
 setText("Кількість товарів у групі: " + gr.getGoods().size() + "\nЗагальна вартість товарів в групі:" + gr.getGroupPrice());  
  
 }  
  
 this.setVisible(true);  
 } else {  
 stats.returned();  
 }  
  
 }  
}

*/\*  
UI for working with groups  
 \*/*import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
  
public class WorkWithGroupUI extends JFrame implements ActionListener {  
 MainMenu menu;  
 JButton addNewGroup;  
 JButton removeGroup;  
 JButton changeGroup;  
 JButton back;  
  
 public WorkWithGroupUI() {  
 super("Робота з групами");  
 this.setSize(700, 500);  
 this.setLayout(new BorderLayout());  
 this.setDefaultCloseOperation(*EXIT\_ON\_CLOSE*);  
 this.setLocationRelativeTo(null);  
 this.setIconImage(new ImageIcon("lab2/Images/wareHouseIcon.png").getImage());  
  
 setWindow();  
  
  
 this.setVisible(false);  
  
 }  
  
 private void setWindow() {  
 *//create upper part label with picture* addUpperPart();  
 *//create central part with buttons to deal with product* addCentralPart();  
 *//create lower part with back button* addLowerPart();  
  
 }  
  
 private void addUpperPart() {  
 JPanel upperPart = new JPanel(new BorderLayout());  
 upperPart.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 4));  
 ImageIcon productPicture = new ImageIcon("lab2/Images/workWithGroup.jpg");*//add picture* JLabel productLabel = new JLabel(); *// Create a JLabel to display the image* productLabel.setIcon(productPicture);  
 productLabel.setHorizontalAlignment(JLabel.*CENTER*);  
 productLabel.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 2));  
 upperPart.add(productLabel, "West");  
  
 JLabel workWithProductLabel = new JLabel("Робота з групами"); *// Create a JLabel to display the name* workWithProductLabel.setHorizontalAlignment(JLabel.*CENTER*);  
 workWithProductLabel.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 2));  
 workWithProductLabel.setFont(new Font("Default", Font.*BOLD*, 17));  
 upperPart.add(workWithProductLabel, "Center");  
  
 this.add(upperPart, "North");  
 }  
  
 private void addCentralPart() {  
 JPanel centralPart = new JPanel(new GridLayout(0, 1));  
 centralPart.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 2));  
  
 JPanel centralLeft = new JPanel(new GridLayout(3, 0));  
 addNewGroup = new JButton();  
 createButtonWithAndAddToPanel(addNewGroup, "Додати групу", centralLeft);  
  
 removeGroup = new JButton();  
 createButtonWithAndAddToPanel(removeGroup, "Видалити групу", centralLeft);  
  
 changeGroup = new JButton();  
 createButtonWithAndAddToPanel(changeGroup, "Редагувати групу", centralLeft);  
  
 centralPart.add(centralLeft);  
 this.add(centralPart, "Center");  
 }  
  
 private void addLowerPart() {  
 JPanel lowerPanel = new JPanel(new FlowLayout());  
 lowerPanel.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 4));  
 back = new JButton();  
 createButtonWithAndAddToPanel(back, "Повернутися назад", lowerPanel);  
 lowerPanel.setBackground(Color.*GRAY*);  
 this.add(lowerPanel, "South");  
 }  
  
 public void setMainMenu(MainMenu menu) {  
 this.menu = menu;  
 }  
  
  
 private void createButtonWithAndAddToPanel(JButton button, String buttonLabel, JPanel originPanel) {  
 JPanel buttonPanel = new JPanel();  
 buttonPanel.setLayout(new FlowLayout());  
 if (!button.equals(back)) {  
 buttonPanel.setBackground(Color.*LIGHT\_GRAY*);  
 } else {  
 buttonPanel.setBackground(Color.*GRAY*);  
 }  
 button.setText(buttonLabel);  
 button.setHorizontalAlignment(JButton.*CENTER*);  
 button.addActionListener(this);  
 button.setPreferredSize(new Dimension(220, 70));  
 button.setFont(new Font("Default", Font.*BOLD*, 17));  
 buttonPanel.add(button);  
 originPanel.add(buttonPanel);  
 }  
  
 public void returned() {  
 this.setVisible(true);  
 }  
  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 this.setVisible(false);  
 if (e.getSource().equals(addNewGroup)) {  
 AddGroupUI addGr = new AddGroupUI();  
 addGr.setWorkWithGroupUI(this);  
 addGr.setVisible(true);  
 } else if (e.getSource().equals(removeGroup)) {  
 DeleteGroupUI delGr = new DeleteGroupUI();  
 delGr.setWorkWithGroupUI(this);  
 delGr.setVisible(true);  
 } else if (e.getSource().equals(changeGroup)) {  
 EditGroupUI delGr = new EditGroupUI();  
 delGr.setWorkWithGroupUI(this);  
 delGr.setVisible(true);  
 } else {  
 this.setVisible(false);  
 menu.returned();  
 }  
 }  
}

*/\*  
UI for adding a group  
 \*/*import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
  
public class AddGroupUI extends JFrame implements ActionListener {  
 Storage storage;  
 WorkWithGroupUI workWithGroupUI;  
 JTextField groupName;  
 JButton addGroup;  
 JButton back;  
  
 public AddGroupUI() {  
 super("Додати групу");  
 this.setSize(700, 500);  
 this.setLayout(new BorderLayout());  
 this.setDefaultCloseOperation(*EXIT\_ON\_CLOSE*);  
 this.setLocationRelativeTo(null);  
 ImageIcon icon = new ImageIcon("lab2/Images/wareHouseIcon.png");  
 this.setIconImage(icon.getImage());  
  
 storage = Storage.*getInstance*();  
 setWindow();  
  
 this.setVisible(true);  
 }  
  
 private void setWindow() {  
 *//create upper part label with picture* addUpperPart();  
 *//create central part with buttons to deal with product* addCentralPart();  
 *//create lower part with back button* addLowerPart();  
  
 }  
  
 private void addUpperPart() {  
 JPanel upperPart = new JPanel(new BorderLayout());  
 upperPart.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 4));  
 ImageIcon productPicture = new ImageIcon("lab2/Images/addProduct.png");*//add picture* JLabel productLabel = new JLabel(); *// Create a JLabel to display the image* productLabel.setIcon(productPicture);  
 productLabel.setHorizontalAlignment(JLabel.*CENTER*);  
 productLabel.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 2));  
 upperPart.add(productLabel, "West");  
  
 JLabel workWithProductLabel = new JLabel("Додавання групи"); *// Create a JLabel to display the name* workWithProductLabel.setHorizontalAlignment(JLabel.*CENTER*);  
 workWithProductLabel.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 2));  
 workWithProductLabel.setFont(new Font("Default", Font.*BOLD*, 17));  
 upperPart.add(workWithProductLabel, "Center");  
  
 this.add(upperPart, "North");  
 }  
  
 private void addCentralPart() {  
 JPanel centralPart = new JPanel(new BorderLayout());  
 centralPart.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 2));  
  
 JPanel centralUp = new JPanel(new GridLayout(1, 0));  
 groupName = new JTextField();  
 createTextFiedWithLabelWithAndAddToPanel(groupName, "Назва групи:", centralUp);  
  
 JPanel centralDown = new JPanel(new GridLayout(1, 0));  
 addGroup = new JButton();  
 createButtonWithAndAddToPanel(addGroup, "Додати групу", centralDown);  
  
  
 centralPart.add(centralUp, "North");  
 centralPart.add(centralDown, "Center");  
 this.add(centralPart, "Center");  
  
 }  
  
  
 private void createTextFiedWithLabelWithAndAddToPanel(JTextField textField, String labelText, JPanel originPanel) {  
 JPanel labelAndTextPanel = new JPanel(new BorderLayout());  
 labelAndTextPanel.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 2));  
 labelAndTextPanel.setBackground(Color.*LIGHT\_GRAY*);  
 JLabel label = new JLabel(labelText);  
  
 labelAndTextPanel.add(label, "North");  
 labelAndTextPanel.add(textField, "Center");  
  
 originPanel.add(labelAndTextPanel);  
 }  
  
 private void addLowerPart() {  
 JPanel lowerPanel = new JPanel(new FlowLayout());  
 lowerPanel.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 4));  
 back = new JButton();  
 createButtonWithAndAddToPanel(back, "Повернутися назад", lowerPanel);  
 lowerPanel.setBackground(Color.*GRAY*);  
  
 this.add(lowerPanel, "South");  
 }  
  
 public void setWorkWithGroupUI(WorkWithGroupUI workWithGroupUI) {  
 this.workWithGroupUI = workWithGroupUI;  
 }  
  
 private void createButtonWithAndAddToPanel(JButton button, String buttonLabel, JPanel originPanel) {  
 JPanel buttonPanel = new JPanel();  
 buttonPanel.setLayout(new FlowLayout());  
 if (!button.equals(back)) {  
 buttonPanel.setBackground(Color.*LIGHT\_GRAY*);  
 } else {  
 buttonPanel.setBackground(Color.*GRAY*);  
 }  
  
 button.setText(buttonLabel);  
 button.setHorizontalAlignment(JButton.*CENTER*);  
 button.addActionListener(this);  
 button.setPreferredSize(new Dimension(220, 70));  
 button.setFont(new Font("Default", Font.*BOLD*, 17));  
 buttonPanel.add(button);  
 originPanel.add(buttonPanel);  
 }  
  
 @Override  
 public void actionPerformed(ActionEvent e) {  
  
 this.setVisible(false);  
 if (e.getSource().equals(addGroup)) {  
 if (Storage.*findGroup*(groupName.getText()) == -1) {  
 if (!groupName.getText().isEmpty()) {  
 storage.addGroup(new Group(groupName.getText()));  
 JOptionPane.*showMessageDialog*(null, "Групу додано", "Успіх", JOptionPane.*INFORMATION\_MESSAGE*);  
 this.setVisible(true);  
 } else {  
 JOptionPane.*showMessageDialog*(null, "Не можна створити групу з порожнім ім'ям", "Помилка", JOptionPane.*ERROR\_MESSAGE*);  
 this.setVisible(true);  
 }  
 } else {  
 JOptionPane.*showMessageDialog*(null, "Така назва групи вже існує", "Помилка", JOptionPane.*ERROR\_MESSAGE*);  
 this.setVisible(true);  
 }  
 } else {  
 workWithGroupUI.returned();  
 }  
  
 }  
}

*/\*  
UI for deleting a group  
 \*/*import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.util.ArrayList;  
  
public class DeleteGroupUI extends JFrame implements ActionListener {  
 Storage storage;  
 WorkWithGroupUI workWithGroupUI;  
 ArrayList<Group> groupsList;  
 JComboBox groups;  
 JButton deleteGroup;  
 JButton back;  
  
 public DeleteGroupUI() {  
 super("Видалити товар");  
 this.setSize(700, 500);  
 this.setLayout(new BorderLayout());  
 this.setDefaultCloseOperation(*EXIT\_ON\_CLOSE*);  
 this.setLocationRelativeTo(null);  
 ImageIcon icon = new ImageIcon("lab2/Images/wareHouseIcon.png");  
 this.setIconImage(icon.getImage());  
  
 storage = Storage.*getInstance*();  
 groupsList = storage.getGroups();  
 setWindow();  
 this.setVisible(true);  
  
 }  
  
 private void setWindow() {  
 *//create upper part label with picture* addUpperPart();  
 *//create central part with buttons to deal with product* addCentralPart();  
 *//create lower part with back button* addLowerPart();  
  
 }  
  
 private void addUpperPart() {  
 JPanel upperPart = new JPanel(new BorderLayout());  
 upperPart.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 4));  
 ImageIcon productPicture = new ImageIcon("lab2/Images/deleteProduct.png");*//add picture* JLabel productLabel = new JLabel(); *// Create a JLabel to display the image* productLabel.setIcon(productPicture);  
 productLabel.setHorizontalAlignment(JLabel.*CENTER*);  
 productLabel.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 2));  
 upperPart.add(productLabel, "West");  
  
 JLabel workWithProductLabel = new JLabel("Видалення групи"); *// Create a JLabel to display the name* workWithProductLabel.setHorizontalAlignment(JLabel.*CENTER*);  
 workWithProductLabel.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 2));  
 workWithProductLabel.setFont(new Font("Default", Font.*BOLD*, 17));  
 upperPart.add(workWithProductLabel, "Center");  
  
 this.add(upperPart, "North");  
 }  
  
 private void addCentralPart() {  
 JPanel centralPart = new JPanel(new GridLayout(0, 2));  
 centralPart.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 2));  
  
 JPanel centralLeft = new JPanel(new GridLayout(1, 0));  
 centralLeft.setBackground(Color.*lightGray*);  
 groups = new JComboBox();  
 createComboBoxWithLabelWithAndAddToPanel(groups, "Групи", centralLeft);  
  
  
 JPanel centralRight = new JPanel(new GridLayout(1, 0));  
 centralRight.setBackground(Color.*lightGray*);  
 deleteGroup = new JButton();  
 createButtonWithAndAddToPanel(deleteGroup, "Видалити групу", centralRight);  
  
 centralPart.add(centralLeft);  
 centralPart.add(centralRight);  
 this.add(centralPart, "Center");  
 }  
  
 private void createComboBoxWithLabelWithAndAddToPanel(JComboBox comboBox, String labelText, JPanel originPanel) {  
 JPanel labelAndTextPanel = new JPanel(new BorderLayout());  
 labelAndTextPanel.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 2));  
 labelAndTextPanel.setBackground(Color.*LIGHT\_GRAY*);  
 JLabel label = new JLabel(labelText);  
  
 labelAndTextPanel.add(label, "North");  
  
 for (int i = 0; i < groupsList.size(); i++) {  
 comboBox.addItem(groupsList.get(i).getName());  
 }  
 labelAndTextPanel.add(comboBox, "Center");  
  
 originPanel.add(labelAndTextPanel);  
 }  
  
 private void addLowerPart() {  
 JPanel lowerPanel = new JPanel(new FlowLayout());  
 lowerPanel.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 4));  
 back = new JButton();  
 createButtonWithAndAddToPanel(back, "Повернутися назад", lowerPanel);  
 lowerPanel.setBackground(Color.*GRAY*);  
 this.add(lowerPanel, "South");  
 }  
  
 public void setWorkWithGroupUI(WorkWithGroupUI workWithGroupUI) {  
 this.workWithGroupUI = workWithGroupUI;  
 }  
  
  
 private void createButtonWithAndAddToPanel(JButton button, String buttonLabel, JPanel originPanel) {  
 JPanel buttonPanel = new JPanel();  
 buttonPanel.setLayout(new FlowLayout());  
 if (button.equals(deleteGroup)) {  
 buttonPanel.setBackground(Color.*LIGHT\_GRAY*);  
 } else {  
 buttonPanel.setBackground(Color.*GRAY*);  
 }  
 button.setText(buttonLabel);  
 button.setHorizontalAlignment(JButton.*CENTER*);  
 button.addActionListener(this);  
 button.setPreferredSize(new Dimension(220, 70));  
 button.setFont(new Font("Default", Font.*BOLD*, 17));  
 buttonPanel.add(button);  
 originPanel.add(buttonPanel);  
 }  
  
 @Override  
 public void actionPerformed(ActionEvent e) {  
  
 this.setVisible(false);  
 if (e.getSource().equals(deleteGroup)) {  
 int groupNumb = groups.getSelectedIndex();  
 Group gr = groupsList.get(groupNumb);  
 storage.deleteGroup(gr);  
 JOptionPane.*showMessageDialog*(null, "Групу видалено", "Успіх", JOptionPane.*INFORMATION\_MESSAGE*);  
  
 workWithGroupUI.returned();  
  
 } else {  
 workWithGroupUI.returned();  
 }  
  
 }  
}

*/\*  
UI for editing a group  
 \*/*import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.util.ArrayList;  
  
public class EditGroupUI extends JFrame implements ActionListener {  
 Storage storage;  
 WorkWithGroupUI workWithGroupUI;  
 JTextField groupName;  
 JComboBox groups;  
 ArrayList<Group> groupsList;  
 JButton changeGroup;  
 JButton back;  
  
 public EditGroupUI() {  
 super("Додати товар");  
 this.setSize(700, 500);  
 this.setLayout(new BorderLayout());  
 this.setDefaultCloseOperation(*EXIT\_ON\_CLOSE*);  
 this.setLocationRelativeTo(null);  
 ImageIcon icon = new ImageIcon("lab2/Images/wareHouseIcon.png");  
 this.setIconImage(icon.getImage());  
  
 storage = Storage.*getInstance*();  
 groupsList = storage.getGroups();  
  
 setWindow();  
 this.setVisible(true);  
  
 }  
  
 private void setWindow() {  
 *//create upper part label with picture* addUpperPart();  
 *//create central part with buttons to deal with product* addCentralPart();  
 *//create lower part with back button* addLowerPart();  
  
 }  
  
 private void addUpperPart() {  
 JPanel upperPart = new JPanel(new BorderLayout());  
 upperPart.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 4));  
 ImageIcon productPicture = new ImageIcon("lab2/Images/editProduct.png");*//add picture* JLabel productLabel = new JLabel(); *// Create a JLabel to display the image* productLabel.setIcon(productPicture);  
 productLabel.setHorizontalAlignment(JLabel.*CENTER*);  
 productLabel.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 2));  
 upperPart.add(productLabel, "West");  
  
 JLabel workWithProductLabel = new JLabel("Зміна групи"); *// Create a JLabel to display the name* workWithProductLabel.setHorizontalAlignment(JLabel.*CENTER*);  
 workWithProductLabel.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 2));  
 workWithProductLabel.setFont(new Font("Default", Font.*BOLD*, 17));  
 upperPart.add(workWithProductLabel, "Center");  
  
 this.add(upperPart, "North");  
 }  
  
 private void addCentralPart() {  
 JPanel centralPart = new JPanel(new GridLayout(0, 2));  
 centralPart.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 2));  
  
 JPanel centralLeft = new JPanel(new GridLayout(3, 0));  
 groupName = new JTextField();  
 createTextFieldWithLabelWithAndAddToPanel(groupName, "Нова назва групи", centralLeft);  
  
 JPanel centralRight = new JPanel(new GridLayout(2, 0));  
  
 groups = new JComboBox();  
 createComboBoxWithLabelWithAndAddToPanel(groups, "Група для редагування", centralRight);  
  
 changeGroup = new JButton();  
 createButtonWithAndAddToPanel(changeGroup, "Змінити групу", centralRight);  
 centralPart.add(centralLeft);  
 centralPart.add(centralRight);  
 this.add(centralPart, "Center");  
  
 }  
  
 private void createComboBoxWithLabelWithAndAddToPanel(JComboBox comboBox, String labelText, JPanel originPanel) {  
 JPanel labelAndTextPanel = new JPanel(new BorderLayout());  
 labelAndTextPanel.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 2));  
 labelAndTextPanel.setBackground(Color.*LIGHT\_GRAY*);  
 JLabel label = new JLabel(labelText);  
  
 labelAndTextPanel.add(label, "North");  
  
 for (int i = 0; i < groupsList.size(); i++) {  
 comboBox.addItem(groupsList.get(i).getName());  
 }  
  
 labelAndTextPanel.add(comboBox, "Center");  
  
 originPanel.add(labelAndTextPanel);  
 }  
  
 private void createTextFieldWithLabelWithAndAddToPanel(JTextField textField, String labelText, JPanel originPanel) {  
 JPanel labelAndTextPanel = new JPanel(new BorderLayout());  
 labelAndTextPanel.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 2));  
 labelAndTextPanel.setBackground(Color.*LIGHT\_GRAY*);  
 JLabel label = new JLabel(labelText);  
  
 labelAndTextPanel.add(label, "North");  
 labelAndTextPanel.add(textField, "Center");  
  
 originPanel.add(labelAndTextPanel);  
 }  
  
 private void addLowerPart() {  
 JPanel lowerPanel = new JPanel(new FlowLayout());  
 lowerPanel.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 4));  
 back = new JButton();  
 createButtonWithAndAddToPanel(back, "Повернутися назад", lowerPanel);  
 lowerPanel.setBackground(Color.*GRAY*);  
  
 this.add(lowerPanel, "South");  
 }  
  
 public void setWorkWithGroupUI(WorkWithGroupUI workWithGroupUI) {  
 this.workWithGroupUI = workWithGroupUI;  
 }  
  
  
 private void createButtonWithAndAddToPanel(JButton button, String buttonLabel, JPanel originPanel) {  
 JPanel buttonPanel = new JPanel();  
 buttonPanel.setLayout(new FlowLayout());  
 if (button.equals(changeGroup)) {  
 buttonPanel.setBackground(Color.*LIGHT\_GRAY*);  
 } else {  
 buttonPanel.setBackground(Color.*GRAY*);  
 }  
 button.setText(buttonLabel);  
 button.setHorizontalAlignment(JButton.*CENTER*);  
 button.addActionListener(this);  
 button.setPreferredSize(new Dimension(220, 70));  
 button.setFont(new Font("Default", Font.*BOLD*, 17));  
 buttonPanel.add(button);  
 originPanel.add(buttonPanel);  
 }  
  
 @Override  
 public void actionPerformed(ActionEvent e) {  
  
 this.setVisible(false);  
 if (e.getSource().equals(changeGroup)) {  
 if (Storage.*findGroup*(groupName.getText()) == -1) {  
 if (!groupName.getText().isEmpty()) {  
 int groupNumb = groups.getSelectedIndex();  
 Group group = groupsList.get(groupNumb);  
 group.setName(groupName.getText());  
 JOptionPane.*showMessageDialog*(null, "Групу змінено", "Успіх", JOptionPane.*INFORMATION\_MESSAGE*);  
 workWithGroupUI.returned();  
 } else {  
 JOptionPane.*showMessageDialog*(null, "Не можна створити групу з порожнім ім'ям", "Помилка", JOptionPane.*ERROR\_MESSAGE*);  
 this.setVisible(true);  
 }  
 } else {  
 JOptionPane.*showMessageDialog*(null, "Така назва групи вже існує", "Помилка", JOptionPane.*ERROR\_MESSAGE*);  
 this.setVisible(true);  
 }  
 } else {  
 workWithGroupUI.returned();  
  
 }  
  
 }  
}

*/\*  
UI for working with goods  
 \*/*import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
  
public class WorkWithProductUI extends JFrame implements ActionListener {  
 MainMenu menu;  
 JButton addNewProduct;  
 JButton removeProduct;  
 JButton changeProduct;  
 JButton addAmount;  
 JButton reduceAmount;  
 JButton back;  
  
 public WorkWithProductUI() {  
 super("Робота з товарами");  
 this.setSize(700, 500);  
 this.setLayout(new BorderLayout());  
 this.setDefaultCloseOperation(*EXIT\_ON\_CLOSE*);  
 this.setLocationRelativeTo(null);  
 this.setIconImage(new ImageIcon("lab2/Images/wareHouseIcon.png").getImage());  
  
 setWindow();  
 this.setVisible(false);  
  
 }  
  
 private void setWindow() {  
 *//create upper part label with picture* addUpperPart();  
 *//create central part with buttons to deal with product* addCentralPart();  
 *//create lower part with back button* addLowerPart();  
  
 }  
  
 private void addUpperPart() {  
 JPanel upperPart = new JPanel(new BorderLayout());  
 upperPart.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 4));  
 ImageIcon productPicture = new ImageIcon("lab2/Images/workWithProduct.jpg");*//add picture* JLabel productLabel = new JLabel(); *// Create a JLabel to display the image* productLabel.setIcon(productPicture);  
 productLabel.setHorizontalAlignment(JLabel.*CENTER*);  
 productLabel.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 2));  
 upperPart.add(productLabel, "West");  
  
 JLabel workWithProductLabel = new JLabel("Робота з продуктом"); *// Create a JLabel to display the name* workWithProductLabel.setHorizontalAlignment(JLabel.*CENTER*);  
 workWithProductLabel.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 2));  
 workWithProductLabel.setFont(new Font("Default", Font.*BOLD*, 17));  
 upperPart.add(workWithProductLabel, "Center");  
  
 this.add(upperPart, "North");  
 }  
  
 private void addCentralPart() {  
 JPanel centralPart = new JPanel(new GridLayout(0, 2));  
  
 JPanel centralLeft = new JPanel(new GridLayout(3, 0));  
 addNewProduct = new JButton();  
 createButtonWithAndAddToPanel(addNewProduct, "Додати новий товар", centralLeft);  
  
 removeProduct = new JButton();  
 createButtonWithAndAddToPanel(removeProduct, "Видалити товар", centralLeft);  
  
 changeProduct = new JButton();  
 createButtonWithAndAddToPanel(changeProduct, "Редагувати товар", centralLeft);  
  
 JPanel centralRight = new JPanel(new GridLayout(2, 0));  
 addAmount = new JButton();  
 createButtonWithAndAddToPanel(addAmount, "Додати кількість", centralRight);  
  
 reduceAmount = new JButton();  
 createButtonWithAndAddToPanel(reduceAmount, "Зменшити кількість", centralRight);  
  
 centralPart.add(centralLeft);  
 centralPart.add(centralRight);  
 this.add(centralPart, "Center");  
 }  
  
 private void addLowerPart() {  
 JPanel lowerPanel = new JPanel(new FlowLayout());  
 lowerPanel.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 4));  
 lowerPanel.setBackground(Color.*GRAY*);  
 back = new JButton();  
 createButtonWithAndAddToPanel(back, "Повернутися назад", lowerPanel);  
  
 this.add(lowerPanel, "South");  
 }  
  
 public void setMainMenu(MainMenu menu) {  
 this.menu = menu;  
 }  
  
  
 private void createButtonWithAndAddToPanel(JButton button, String buttonLabel, JPanel originPanel) {  
 JPanel buttonPanel = new JPanel();  
 buttonPanel.setLayout(new FlowLayout());  
 if (!button.equals(back)) {  
 buttonPanel.setBackground(Color.*LIGHT\_GRAY*);  
 } else {  
 buttonPanel.setBackground(Color.*GRAY*);  
 }  
 button.setText(buttonLabel);  
 button.setHorizontalAlignment(JButton.*CENTER*);  
 button.addActionListener(this);  
 button.setPreferredSize(new Dimension(220, 70));  
 button.setFont(new Font("Default", Font.*BOLD*, 17));  
 buttonPanel.add(button);  
 originPanel.add(buttonPanel);  
 }  
  
 public void returned() {  
 this.setVisible(true);  
 }  
  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 this.setVisible(false);  
 if (e.getSource().equals(addNewProduct)) {  
 AddGoodUI addGoodUI = new AddGoodUI();  
 addGoodUI.setWorkWithProductUI(this);  
 addGoodUI.setVisible(true);  
 } else if (e.getSource().equals(removeProduct)) {  
 DeleteGoodUI delGoodUI = new DeleteGoodUI();  
 delGoodUI.setWorkWithProductUI(this);  
 delGoodUI.setVisible(true);  
 } else if (e.getSource().equals(changeProduct)) {  
 EditGoodUI editGoodUI = new EditGoodUI();  
 editGoodUI.setWorkWithProductUI(this);  
 editGoodUI.setVisible(true);  
 } else if (e.getSource().equals(addAmount)) {  
 AddQuantityOfProductUI addProd = new AddQuantityOfProductUI();  
 addProd.setWorkWithProductUI(this);  
 addProd.setVisible(true);  
 } else if (e.getSource().equals(reduceAmount)) {  
 RedQuantityOfProductUI reduceProd = new RedQuantityOfProductUI();  
 reduceProd.setWorkWithProductUI(this);  
 reduceProd.setVisible(true);  
 } else {  
 this.setVisible(false);  
 menu.returned();  
 }  
 }  
}

*/\*  
UI for adding a good  
 \*/*import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.util.ArrayList;  
  
public class AddGoodUI extends JFrame implements ActionListener {  
 Storage storage;  
 WorkWithProductUI workWithProductUI;  
 JTextField productName;  
 JTextArea description;  
 JTextField manufacturer;  
 JSpinner amount;  
 JSpinner price;  
 JComboBox groups;  
 ArrayList<Group> groupsList;  
 JButton createProduct;  
 JButton back;  
 JFrame imageFrame;  
  
 public AddGoodUI() {  
 super("Додати товар");  
 this.setSize(700, 500);  
 this.setLayout(new BorderLayout());  
 this.setDefaultCloseOperation(*EXIT\_ON\_CLOSE*);  
 this.setLocationRelativeTo(null);  
 ImageIcon icon = new ImageIcon("lab2/Images/wareHouseIcon.png");  
 this.setIconImage(icon.getImage());  
  
 storage = Storage.*getInstance*();  
 groupsList = storage.getGroups();  
 setWindow();  
  
  
 this.setVisible(true);  
 }  
  
 private void setWindow() {  
 *//create upper part label with picture* addUpperPart();  
 *//create central part with buttons to deal with product* addCentralPart();  
 *//create lower part with back button* addLowerPart();  
  
 }  
  
 private void addUpperPart() {  
 JPanel upperPart = new JPanel(new BorderLayout());  
 upperPart.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 4));  
 ImageIcon productPicture = new ImageIcon("lab2/Images/addProduct.png");*//add picture* JLabel productLabel = new JLabel(); *// Create a JLabel to display the image* productLabel.setIcon(productPicture);  
 productLabel.setHorizontalAlignment(JLabel.*CENTER*);  
 productLabel.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 2));  
 upperPart.add(productLabel, "West");  
  
 JLabel workWithProductLabel = new JLabel("Додавання товару"); *// Create a JLabel to display the name* workWithProductLabel.setHorizontalAlignment(JLabel.*CENTER*);  
 workWithProductLabel.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 2));  
 workWithProductLabel.setFont(new Font("Default", Font.*BOLD*, 17));  
 upperPart.add(workWithProductLabel, "Center");  
  
 this.add(upperPart, "North");  
 }  
  
 private void addCentralPart() {  
 JPanel centralPart = new JPanel(new GridLayout(0, 2));  
 centralPart.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 2));  
  
 JPanel centralLeft = new JPanel(new GridLayout(2, 0));  
 JPanel centralLeftUp = new JPanel(new GridLayout(2, 1));  
 productName = new JTextField();  
 createTextFieldWithLabelWithAndAddToPanel(productName, "Назва товару", centralLeftUp);  
 manufacturer = new JTextField();  
 createTextFieldWithLabelWithAndAddToPanel(manufacturer, "Виробник", centralLeftUp);  
 centralLeft.add(centralLeftUp);  
 description = new JTextArea();  
 createAreaFieldWithLabelWithAndAddToPanel(description, "Опис", centralLeft);  
  
  
 JPanel centralRight = new JPanel(new GridLayout(3, 0));  
  
 groups = new JComboBox();  
 createComboBoxWithLabelWithAndAddToPanel(groups, "Група", centralRight);  
  
 amount = new JSpinner();  
 createSpinnerWithLabelWithAndAddToPanel(amount, "Кількість", centralRight);  
  
 price = new JSpinner();  
 createSpinnerWithLabelWithAndAddToPanel(price, "Ціна", centralRight);  
 centralPart.add(centralLeft);  
 centralPart.add(centralRight);  
 this.add(centralPart, "Center");  
 }  
  
 private void createComboBoxWithLabelWithAndAddToPanel(JComboBox comboBox, String labelText, JPanel originPanel) {  
 JPanel labelAndTextPanel = new JPanel(new BorderLayout());  
 labelAndTextPanel.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 2));  
 labelAndTextPanel.setBackground(Color.*LIGHT\_GRAY*);  
 JLabel label = new JLabel(labelText);  
  
 labelAndTextPanel.add(label, "North");  
  
 for (int i = 0; i < groupsList.size(); i++) {  
 comboBox.addItem(groupsList.get(i).getName());  
 }  
 labelAndTextPanel.add(comboBox, "Center");  
  
 originPanel.add(labelAndTextPanel);  
 }  
  
 private void createSpinnerWithLabelWithAndAddToPanel(JSpinner spinner, String labelText, JPanel originPanel) {  
 JPanel labelAndTextPanel = new JPanel(new BorderLayout());  
 labelAndTextPanel.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 2));  
 labelAndTextPanel.setBackground(Color.*LIGHT\_GRAY*);  
 JLabel label = new JLabel(labelText);  
  
 labelAndTextPanel.add(label, "North");  
 if (spinner.equals(amount)) {  
 spinner.setModel(new SpinnerNumberModel(1, 1, Integer.*MAX\_VALUE*, 1));  
 } else {  
 spinner.setModel(new SpinnerNumberModel(0.1, 0.1, Integer.*MAX\_VALUE*, 0.1));  
 }  
 labelAndTextPanel.add(spinner, "Center");  
  
 originPanel.add(labelAndTextPanel);  
 }  
  
 private void createAreaFieldWithLabelWithAndAddToPanel(JTextArea textArea, String labelText, JPanel originPanel) {  
 JPanel labelAndTextPanel = new JPanel(new BorderLayout());  
 labelAndTextPanel.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 2));  
 labelAndTextPanel.setBackground(Color.*LIGHT\_GRAY*);  
 JLabel label = new JLabel(labelText);  
  
 labelAndTextPanel.add(label, "North");  
 JScrollPane textAreaScroll = new JScrollPane(textArea);  
 labelAndTextPanel.add(textAreaScroll, "Center");  
  
 originPanel.add(labelAndTextPanel);  
 }  
  
 private void createTextFieldWithLabelWithAndAddToPanel(JTextField textField, String labelText, JPanel originPanel) {  
 JPanel labelAndTextPanel = new JPanel(new BorderLayout());  
 labelAndTextPanel.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 2));  
 labelAndTextPanel.setBackground(Color.*LIGHT\_GRAY*);  
 JLabel label = new JLabel(labelText);  
  
 labelAndTextPanel.add(label, "North");  
 labelAndTextPanel.add(textField, "Center");  
  
 originPanel.add(labelAndTextPanel);  
 }  
  
 private void addLowerPart() {  
 JPanel lowerPanel = new JPanel(new FlowLayout());  
 lowerPanel.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 4));  
 back = new JButton();  
 createButtonWithAndAddToPanel(back, "Повернутися назад", lowerPanel);  
 lowerPanel.setBackground(Color.*GRAY*);  
 createProduct = new JButton();  
 createButtonWithAndAddToPanel(createProduct, "Додати товар", lowerPanel);  
 this.add(lowerPanel, "South");  
 }  
  
 public void setWorkWithProductUI(WorkWithProductUI workWithProductUI) {  
 this.workWithProductUI = workWithProductUI;  
 }  
  
  
 private void createButtonWithAndAddToPanel(JButton button, String buttonLabel, JPanel originPanel) {  
 JPanel buttonPanel = new JPanel();  
 buttonPanel.setLayout(new FlowLayout());  
 buttonPanel.setBackground(Color.*GRAY*);  
 button.setText(buttonLabel);  
 button.setHorizontalAlignment(JButton.*CENTER*);  
 button.addActionListener(this);  
 button.setPreferredSize(new Dimension(220, 70));  
 button.setFont(new Font("Default", Font.*BOLD*, 17));  
 buttonPanel.add(button);  
 originPanel.add(buttonPanel);  
 }  
  
 @Override  
 public void actionPerformed(ActionEvent e) {  
  
 this.setVisible(false);  
 if (e.getSource().equals(createProduct)) {  
  
 if (storage.findGood(productName.getText()).isEmpty() ||  
 !storage.findGood(productName.getText()).getFirst().getName().equals(productName.getText())) {  
 String text = productName.getText();  
 if (!text.isBlank()) {  
 if (!text.equals("Окуляри")) {  
 int groupNumb = groups.getSelectedIndex();  
 Group tempGr = groupsList.get(groupNumb);  
 tempGr.addGood(new Good(tempGr.getName(), productName.getText(), description.getText(), manufacturer.getText(), (Integer) amount.getValue(), Float.*parseFloat*(Double.*toString*((Double) price.getValue()))));  
 JOptionPane.*showMessageDialog*(null, "Товар додано", "Успіх", JOptionPane.*INFORMATION\_MESSAGE*);  
 this.setVisible(true);  
 } else {  
 ImageIcon imageIcon = new ImageIcon("lab2/Images/estrEgg.gif");  
  
 imageFrame = new JFrame("Easter Egg");  
 imageFrame.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
 imageFrame.setSize(700, 500);  
 imageFrame.setLayout(new BorderLayout());  
 imageFrame.setLocationRelativeTo(null);  
 imageFrame.setVisible(true);  
  
 JLabel imageLabel = new JLabel(imageIcon);  
 imageLabel.setHorizontalAlignment(JLabel.*CENTER*);  
 imageLabel.setVerticalAlignment(JLabel.*CENTER*);  
  
 imageFrame.add(imageLabel, "Center");  
 JPanel backButton = new JPanel(new FlowLayout());  
 backButton.setBackground(Color.*GRAY*);  
 back = new JButton();  
 createButtonWithAndAddToPanel(back, "Повернутися назад", backButton);  
  
 imageFrame.add(backButton, "South");  
  
 }  
 } else {  
 JOptionPane.*showMessageDialog*(null, "Не можна створити товар з порожнім ім'ям", "Помилка", JOptionPane.*ERROR\_MESSAGE*);  
 this.setVisible(true);  
 }  
 } else {  
 JOptionPane.*showMessageDialog*(null, "Така назва товару вже існує", "Помилка", JOptionPane.*ERROR\_MESSAGE*);  
 this.setVisible(true);  
 }  
 } else {  
 if (imageFrame != null) {  
 imageFrame.setVisible(false);  
 }  
 workWithProductUI.returned();  
 }  
  
 }  
}

*/\*  
UI for deleting a good  
 \*/*import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.util.ArrayList;  
  
public class DeleteGoodUI extends JFrame implements ActionListener {  
 Storage storage;  
 WorkWithProductUI workWithProductUI;  
 JTextField oldProductName;  
 String rememOldProdName;  
 ArrayList<Good> productsList;  
 JComboBox products;  
 JButton deleteProduct;  
 JButton back;  
  
 public DeleteGoodUI() {  
 super("Видалити товар");  
 this.setSize(700, 500);  
 this.setLayout(new BorderLayout());  
 this.setDefaultCloseOperation(*EXIT\_ON\_CLOSE*);  
 this.setLocationRelativeTo(null);  
 ImageIcon icon = new ImageIcon("lab2/Images/wareHouseIcon.png");  
 this.setIconImage(icon.getImage());  
  
 storage = Storage.*getInstance*();  
 productsList = storage.findGood("");  
 setWindow();  
  
 class listOfGoodsUpdateThread implements Runnable {  
 public void run() {  
 while (true) {  
 if (!oldProductName.getText().equals(rememOldProdName)) {  
 productsList = storage.findGood(oldProductName.getText());  
 products.removeAllItems();  
 for (int i = 0; i < productsList.size(); i++) {  
 products.addItem(productsList.get(i).getName());  
 }  
 rememOldProdName = oldProductName.getText();  
 }  
  
 try {  
 Thread.*sleep*(500);  
 } catch (InterruptedException e) {  
 System.*out*.println("Error");  
 }  
 }  
 }  
 }  
 Thread updateProdList = new Thread(new listOfGoodsUpdateThread());  
 updateProdList.start();  
 this.setVisible(true);  
  
 }  
  
 private void setWindow() {  
 *//create upper part label with picture* addUpperPart();  
 *//create central part with buttons to deal with product* addCentralPart();  
 *//create lower part with back button* addLowerPart();  
  
 }  
  
 private void addUpperPart() {  
 JPanel upperPart = new JPanel(new BorderLayout());  
 upperPart.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 4));  
 ImageIcon productPicture = new ImageIcon("lab2/Images/deleteProduct.png");*//add picture* JLabel productLabel = new JLabel(); *// Create a JLabel to display the image* productLabel.setIcon(productPicture);  
 productLabel.setHorizontalAlignment(JLabel.*CENTER*);  
 productLabel.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 2));  
 upperPart.add(productLabel, "West");  
  
 JLabel workWithProductLabel = new JLabel("Видалення товару"); *// Create a JLabel to display the name* workWithProductLabel.setHorizontalAlignment(JLabel.*CENTER*);  
 workWithProductLabel.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 2));  
 workWithProductLabel.setFont(new Font("Default", Font.*BOLD*, 17));  
 upperPart.add(workWithProductLabel, "Center");  
  
 this.add(upperPart, "North");  
 }  
  
 private void addCentralPart() {  
 JPanel centralPart = new JPanel(new GridLayout(0, 2));  
 centralPart.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 2));  
  
 JPanel centralLeft = new JPanel(new GridLayout(2, 0));  
 oldProductName = new JTextField();  
 createTextFieldWithLabelWithAndAddToPanel(oldProductName, "Назва товару", centralLeft);  
  
 products = new JComboBox();  
 createComboBoxWithLabelWithAndAddToPanel(products, "Товари", centralLeft);  
  
  
 JPanel centralRight = new JPanel(new GridLayout(1, 0));  
  
 deleteProduct = new JButton();  
 createButtonWithAndAddToPanel(deleteProduct, "Видалити товар", centralRight);  
  
 centralPart.add(centralLeft);  
 centralPart.add(centralRight);  
 this.add(centralPart, "Center");  
 }  
  
 private void createComboBoxWithLabelWithAndAddToPanel(JComboBox comboBox, String labelText, JPanel originPanel) {  
 JPanel labelAndTextPanel = new JPanel(new BorderLayout());  
 labelAndTextPanel.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 2));  
 labelAndTextPanel.setBackground(Color.*LIGHT\_GRAY*);  
 JLabel label = new JLabel(labelText);  
  
 labelAndTextPanel.add(label, "North");  
  
 for (int i = 0; i < productsList.size(); i++) {  
 comboBox.addItem(productsList.get(i).getName());  
 }  
 labelAndTextPanel.add(comboBox, "Center");  
  
 originPanel.add(labelAndTextPanel);  
 }  
  
 private void createTextFieldWithLabelWithAndAddToPanel(JTextField textField, String labelText, JPanel originPanel) {  
 JPanel labelAndTextPanel = new JPanel(new BorderLayout());  
 labelAndTextPanel.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 2));  
 labelAndTextPanel.setBackground(Color.*LIGHT\_GRAY*);  
 JLabel label = new JLabel(labelText);  
  
 labelAndTextPanel.add(label, "North");  
 labelAndTextPanel.add(textField, "Center");  
  
 originPanel.add(labelAndTextPanel);  
 }  
  
 private void addLowerPart() {  
 JPanel lowerPanel = new JPanel(new FlowLayout());  
 lowerPanel.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 4));  
 back = new JButton();  
 createButtonWithAndAddToPanel(back, "Повернутися назад", lowerPanel);  
 lowerPanel.setBackground(Color.*GRAY*);  
 this.add(lowerPanel, "South");  
 }  
  
 public void setWorkWithProductUI(WorkWithProductUI workWithProductUI) {  
 this.workWithProductUI = workWithProductUI;  
 }  
  
  
 private void createButtonWithAndAddToPanel(JButton button, String buttonLabel, JPanel originPanel) {  
 JPanel buttonPanel = new JPanel();  
 buttonPanel.setLayout(new FlowLayout());  
 if (button.equals(deleteProduct)) {  
 buttonPanel.setBackground(Color.*LIGHT\_GRAY*);  
 } else {  
 buttonPanel.setBackground(Color.*GRAY*);  
 }  
 button.setText(buttonLabel);  
 button.setHorizontalAlignment(JButton.*CENTER*);  
 button.addActionListener(this);  
 button.setPreferredSize(new Dimension(220, 70));  
 button.setFont(new Font("Default", Font.*BOLD*, 17));  
 buttonPanel.add(button);  
 originPanel.add(buttonPanel);  
 }  
  
 @Override  
 public void actionPerformed(ActionEvent e) {  
  
 this.setVisible(false);  
 if (e.getSource().equals(deleteProduct)) {  
 int prodNumb = products.getSelectedIndex();  
 Good prod = productsList.get(prodNumb);  
 storage.deleteGood(prod.getName());  
 JOptionPane.*showMessageDialog*(null, "Товар видалено", "Успіх", JOptionPane.*INFORMATION\_MESSAGE*);  
 oldProductName.setText("");  
 rememOldProdName = "NO text";  
 workWithProductUI.returned();  
  
 } else {  
 workWithProductUI.returned();  
 }  
  
 }  
}

*/\*  
UI for editing a good  
 \*/*import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.util.ArrayList;  
  
public class EditGoodUI extends JFrame implements ActionListener {  
 Storage storage;  
 WorkWithProductUI workWithProductUI;  
 JTextField oldProductName;  
 String rememOldProdName;  
 JTextField productName;  
 JTextArea description;  
 JTextField manufacturer;  
 JSpinner amount;  
 JSpinner price;  
 JComboBox products;  
 JComboBox groups;  
 ArrayList<Group> groupsList;  
 ArrayList<Good> productsList;  
 JButton changeProduct;  
 JButton back;  
 Thread updateProdList;  
  
 public EditGoodUI() {  
 super("Додати товар");  
 this.setSize(700, 500);  
 this.setLayout(new BorderLayout());  
 this.setDefaultCloseOperation(*EXIT\_ON\_CLOSE*);  
 this.setLocationRelativeTo(null);  
 ImageIcon icon = new ImageIcon("lab2/Images/wareHouseIcon.png");  
 this.setIconImage(icon.getImage());  
  
 storage = Storage.*getInstance*();  
 groupsList = storage.getGroups();  
 productsList = storage.findGood("");  
 setWindow();  
 class listOfGoodsUpdateThread implements Runnable {  
 public void run() {  
 while (true) {  
 if (!oldProductName.getText().equals(rememOldProdName)) {  
 productsList = storage.findGood(oldProductName.getText());  
 products.removeAllItems();  
 for (int i = 0; i < productsList.size(); i++) {  
 products.addItem(productsList.get(i).getName());  
 }  
 rememOldProdName = oldProductName.getText();  
 }  
  
 try {  
 Thread.*sleep*(500);  
 } catch (InterruptedException e) {  
 System.*out*.println("Error");  
 }  
 }  
 }  
 }  
 updateProdList = new Thread(new listOfGoodsUpdateThread());  
 updateProdList.start();  
 this.setVisible(true);  
  
 }  
  
 private void setWindow() {  
 *//create upper part label with picture* addUpperPart();  
 *//create central part with buttons to deal with product* addCentralPart();  
 *//create lower part with back button* addLowerPart();  
  
 }  
  
 private void addUpperPart() {  
 JPanel upperPart = new JPanel(new BorderLayout());  
 upperPart.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 4));  
 ImageIcon productPicture = new ImageIcon("lab2/Images/editProduct.png");*//add picture* JLabel productLabel = new JLabel(); *// Create a JLabel to display the image* productLabel.setIcon(productPicture);  
 productLabel.setHorizontalAlignment(JLabel.*CENTER*);  
 productLabel.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 2));  
 upperPart.add(productLabel, "West");  
  
 JLabel workWithProductLabel = new JLabel("Зміна товару"); *// Create a JLabel to display the name* workWithProductLabel.setHorizontalAlignment(JLabel.*CENTER*);  
 workWithProductLabel.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 2));  
 workWithProductLabel.setFont(new Font("Default", Font.*BOLD*, 17));  
 upperPart.add(workWithProductLabel, "Center");  
  
 this.add(upperPart, "North");  
 }  
  
 private void addCentralPart() {  
 JPanel centralPart = new JPanel(new GridLayout(0, 2));  
 centralPart.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 2));  
  
 JPanel centralLeft = new JPanel(new GridLayout(2, 0));  
 JPanel centralLeftUp = new JPanel(new GridLayout(2, 1));  
 productName = new JTextField();  
 createTextFieldWithLabelWithAndAddToPanel(productName, "Нова назва товару", centralLeftUp);  
 manufacturer = new JTextField();  
 createTextFieldWithLabelWithAndAddToPanel(manufacturer, "Виробник", centralLeftUp);  
 centralLeft.add(centralLeftUp);  
 description = new JTextArea();  
 createAreaFieldWithLabelWithAndAddToPanel(description, "Опис", centralLeft);  
  
 JPanel centralRight = new JPanel(new GridLayout(5, 0));  
  
 oldProductName = new JTextField();  
 createTextFieldWithLabelWithAndAddToPanel(oldProductName, "Назва товару", centralRight);  
  
 products = new JComboBox();  
 createComboBoxWithLabelWithAndAddToPanel(products, "Товари", centralRight);  
  
 groups = new JComboBox();  
 createComboBoxWithLabelWithAndAddToPanel(groups, "Група", centralRight);  
  
 amount = new JSpinner();  
 createSpinnerWithLabelWithAndAddToPanel(amount, "Кількість", centralRight);  
  
 price = new JSpinner();  
 createSpinnerWithLabelWithAndAddToPanel(price, "Ціна", centralRight);  
 centralPart.add(centralLeft);  
 centralPart.add(centralRight);  
 this.add(centralPart, "Center");  
  
 }  
  
 private void createComboBoxWithLabelWithAndAddToPanel(JComboBox comboBox, String labelText, JPanel originPanel) {  
 JPanel labelAndTextPanel = new JPanel(new BorderLayout());  
 labelAndTextPanel.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 2));  
 labelAndTextPanel.setBackground(Color.*LIGHT\_GRAY*);  
 JLabel label = new JLabel(labelText);  
  
 labelAndTextPanel.add(label, "North");  
 if (comboBox.equals(groups)) {  
 for (int i = 0; i < groupsList.size(); i++) {  
 comboBox.addItem(groupsList.get(i).getName());  
 }  
 } else {  
 for (int i = 0; i < productsList.size(); i++) {  
 comboBox.addItem(productsList.get(i).getName());  
 }  
 }  
 labelAndTextPanel.add(comboBox, "Center");  
  
 originPanel.add(labelAndTextPanel);  
 }  
  
 private void createSpinnerWithLabelWithAndAddToPanel(JSpinner spinner, String labelText, JPanel originPanel) {  
 JPanel labelAndTextPanel = new JPanel(new BorderLayout());  
 labelAndTextPanel.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 2));  
 labelAndTextPanel.setBackground(Color.*LIGHT\_GRAY*);  
 JLabel label = new JLabel(labelText);  
  
 labelAndTextPanel.add(label, "North");  
 if (spinner.equals(amount)) {  
 spinner.setModel(new SpinnerNumberModel(1, 1, Integer.*MAX\_VALUE*, 1));  
 } else {  
 spinner.setModel(new SpinnerNumberModel(0.1, 0.1, Integer.*MAX\_VALUE*, 0.1));  
 }  
 labelAndTextPanel.add(spinner, "Center");  
  
 originPanel.add(labelAndTextPanel);  
 }  
  
 private void createAreaFieldWithLabelWithAndAddToPanel(JTextArea textArea, String labelText, JPanel originPanel) {  
 JPanel labelAndTextPanel = new JPanel(new BorderLayout());  
 labelAndTextPanel.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 2));  
 labelAndTextPanel.setBackground(Color.*LIGHT\_GRAY*);  
 JLabel label = new JLabel(labelText);  
  
 labelAndTextPanel.add(label, "North");  
 JScrollPane textAreaScroll = new JScrollPane(textArea);  
 labelAndTextPanel.add(textAreaScroll, "Center");  
  
 originPanel.add(labelAndTextPanel);  
 }  
  
 private void createTextFieldWithLabelWithAndAddToPanel(JTextField textField, String labelText, JPanel originPanel) {  
 JPanel labelAndTextPanel = new JPanel(new BorderLayout());  
 labelAndTextPanel.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 2));  
 labelAndTextPanel.setBackground(Color.*LIGHT\_GRAY*);  
 JLabel label = new JLabel(labelText);  
  
 labelAndTextPanel.add(label, "North");  
 labelAndTextPanel.add(textField, "Center");  
  
 originPanel.add(labelAndTextPanel);  
 }  
  
 private void addLowerPart() {  
 JPanel lowerPanel = new JPanel(new FlowLayout());  
 lowerPanel.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 4));  
 back = new JButton();  
 createButtonWithAndAddToPanel(back, "Повернутися назад", lowerPanel);  
 lowerPanel.setBackground(Color.*GRAY*);  
 changeProduct = new JButton();  
 createButtonWithAndAddToPanel(changeProduct, "Змінити товар", lowerPanel);  
 this.add(lowerPanel, "South");  
 }  
  
 public void setWorkWithProductUI(WorkWithProductUI workWithProductUI) {  
 this.workWithProductUI = workWithProductUI;  
 }  
  
  
 private void createButtonWithAndAddToPanel(JButton button, String buttonLabel, JPanel originPanel) {  
 JPanel buttonPanel = new JPanel();  
 buttonPanel.setLayout(new FlowLayout());  
 buttonPanel.setBackground(Color.*GRAY*);  
 button.setText(buttonLabel);  
 button.setHorizontalAlignment(JButton.*CENTER*);  
 button.addActionListener(this);  
 button.setPreferredSize(new Dimension(220, 70));  
 button.setFont(new Font("Default", Font.*BOLD*, 17));  
 buttonPanel.add(button);  
 originPanel.add(buttonPanel);  
 }  
  
 @Override  
 public void actionPerformed(ActionEvent e) {  
  
 this.setVisible(false);  
 if (e.getSource().equals(changeProduct)) {  
 int prodNumb = products.getSelectedIndex();  
 Good prod = productsList.get(prodNumb);  
 if (storage.findGood(productName.getText()).isEmpty() ||  
 storage.findGood(productName.getText()).getFirst().getName().equals(productName.getText())) {  
 String text = productName.getText();  
 if (!text.isBlank()) {  
  
 storage.deleteGood(prod.getName());  
 int groupNumb = groups.getSelectedIndex();  
 Group tempGr = groupsList.get(groupNumb);  
 tempGr.addGood(new Good(tempGr.getName(), productName.getText(), description.getText(), manufacturer.getText(), (Integer) amount.getValue(), Float.*parseFloat*(Double.*toString*((Double) price.getValue()))));  
 JOptionPane.*showMessageDialog*(null, "Товар змінено", "Успіх", JOptionPane.*INFORMATION\_MESSAGE*);  
  
 oldProductName.setText("");  
 rememOldProdName = "NO text";  
 this.setVisible(true);  
 } else {  
 JOptionPane.*showMessageDialog*(null, "Не можна створити товар з порожнім ім'ям", "Помилка", JOptionPane.*ERROR\_MESSAGE*);  
 this.setVisible(true);  
 }  
 } else {  
 JOptionPane.*showMessageDialog*(null, "Така назва товару вже існує", "Помилка", JOptionPane.*ERROR\_MESSAGE*);  
 this.setVisible(true);  
 }  
 } else {  
 workWithProductUI.returned();  
 }  
  
 }  
}

*/\*  
UI for adding amount of a good  
 \*/*import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.util.ArrayList;  
  
public class AddQuantityOfProductUI extends JFrame implements ActionListener {  
 Storage storage;  
 WorkWithProductUI workWithProductUI;  
 JTextField oldProductName;  
 String rememOldProdName;  
 JSpinner amount;  
 JComboBox products;  
 JComboBox groups;  
 ArrayList<Group> groupsList;  
 ArrayList<Good> productsList;  
 JButton changeQuantityProduct;  
 JButton back;  
 Thread updateProdList;  
  
 public AddQuantityOfProductUI() {  
 super("Додати кількість");  
 this.setSize(700, 500);  
 this.setLayout(new BorderLayout());  
 this.setDefaultCloseOperation(*EXIT\_ON\_CLOSE*);  
 this.setLocationRelativeTo(null);  
 ImageIcon icon = new ImageIcon("lab2/Images/wareHouseIcon.png");  
 this.setIconImage(icon.getImage());  
  
 storage = Storage.*getInstance*();  
 productsList = storage.findGood("");  
 setWindow();  
 class listOfGoodsUpdateThread implements Runnable {  
 public void run() {  
 while (true) {  
 if (!oldProductName.getText().equals(rememOldProdName)) {  
 productsList = storage.findGood(oldProductName.getText());  
 products.removeAllItems();  
 for (int i = 0; i < productsList.size(); i++) {  
 products.addItem(productsList.get(i).getName());  
 }  
 rememOldProdName = oldProductName.getText();  
 }  
  
 try {  
 Thread.*sleep*(500);  
 } catch (InterruptedException e) {  
 System.*out*.println("Error");  
 }  
 }  
 }  
 }  
 updateProdList = new Thread(new listOfGoodsUpdateThread());  
 updateProdList.start();  
 this.setVisible(true);  
  
 }  
  
 private void setWindow() {  
 *//create upper part label with picture* addUpperPart();  
 *//create central part with buttons to deal with product* addCentralPart();  
 *//create lower part with back button* addLowerPart();  
  
 }  
  
 private void addUpperPart() {  
 JPanel upperPart = new JPanel(new BorderLayout());  
 upperPart.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 4));  
 ImageIcon productPicture = new ImageIcon("lab2/Images/plusProduct.png");*//add picture* JLabel productLabel = new JLabel(); *// Create a JLabel to display the image* productLabel.setIcon(productPicture);  
 productLabel.setHorizontalAlignment(JLabel.*CENTER*);  
 productLabel.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 2));  
 upperPart.add(productLabel, "West");  
  
 JLabel workWithProductLabel = new JLabel("Додавання кількості товару"); *// Create a JLabel to display the name* workWithProductLabel.setHorizontalAlignment(JLabel.*CENTER*);  
 workWithProductLabel.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 2));  
 workWithProductLabel.setFont(new Font("Default", Font.*BOLD*, 17));  
 upperPart.add(workWithProductLabel, "Center");  
  
 this.add(upperPart, "North");  
 }  
  
 private void addCentralPart() {  
 JPanel centralPart = new JPanel(new GridLayout(0, 2));  
 centralPart.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 2));  
  
  
 JPanel centralLeft = new JPanel(new GridLayout(3, 0));  
 centralLeft.setBackground(Color.*LIGHT\_GRAY*);  
 oldProductName = new JTextField();  
 createTextFiedWithLabelWithAndAddToPanel(oldProductName, "Назва товару", centralLeft);  
  
 products = new JComboBox();  
 createComboBoxWithLabelWithAndAddToPanel(products, "Товари", centralLeft);  
  
 JPanel centralRight = new JPanel(new GridLayout(3, 0));  
 centralRight.setBackground(Color.*LIGHT\_GRAY*);  
  
 amount = new JSpinner();  
 createSpinnerWithLabelWithAndAddToPanel(amount, "Додати кількість на:", centralRight);  
  
 centralPart.add(centralLeft);  
 centralPart.add(centralRight);  
 this.add(centralPart, "Center");  
  
 }  
  
 private void createComboBoxWithLabelWithAndAddToPanel(JComboBox comboBox, String labelText, JPanel originPanel) {  
 JPanel labelAndTextPanel = new JPanel(new BorderLayout());  
 labelAndTextPanel.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 2));  
 labelAndTextPanel.setBackground(Color.*LIGHT\_GRAY*);  
 JLabel label = new JLabel(labelText);  
  
 labelAndTextPanel.add(label, "North");  
 if (comboBox.equals(groups)) {  
 for (int i = 0; i < groupsList.size(); i++) {  
 comboBox.addItem(groupsList.get(i).getName());  
 }  
 } else {  
 for (int i = 0; i < productsList.size(); i++) {  
 comboBox.addItem(productsList.get(i).getName());  
 }  
 }  
 labelAndTextPanel.add(comboBox, "Center");  
  
 originPanel.add(labelAndTextPanel);  
 }  
  
 private void createSpinnerWithLabelWithAndAddToPanel(JSpinner spinner, String labelText, JPanel originPanel) {  
 JPanel labelAndTextPanel = new JPanel(new BorderLayout());  
 labelAndTextPanel.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 2));  
 labelAndTextPanel.setBackground(Color.*LIGHT\_GRAY*);  
 JLabel label = new JLabel(labelText);  
  
 labelAndTextPanel.add(label, "North");  
 spinner.setModel(new SpinnerNumberModel(1, 1, Integer.*MAX\_VALUE*, 1));  
 labelAndTextPanel.add(spinner, "Center");  
  
 originPanel.add(labelAndTextPanel);  
 }  
  
 private void createTextFiedWithLabelWithAndAddToPanel(JTextField textField, String labelText, JPanel originPanel) {  
 JPanel labelAndTextPanel = new JPanel(new BorderLayout());  
 labelAndTextPanel.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 2));  
 labelAndTextPanel.setBackground(Color.*LIGHT\_GRAY*);  
 JLabel label = new JLabel(labelText);  
  
 labelAndTextPanel.add(label, "North");  
 labelAndTextPanel.add(textField, "Center");  
  
 originPanel.add(labelAndTextPanel);  
 }  
  
 private void addLowerPart() {  
 JPanel lowerPanel = new JPanel(new FlowLayout());  
 lowerPanel.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 4));  
 back = new JButton();  
 createButtonWithAndAddToPanel(back, "Повернутися назад", lowerPanel);  
 lowerPanel.setBackground(Color.*GRAY*);  
 changeQuantityProduct = new JButton();  
 createButtonWithAndAddToPanel(changeQuantityProduct, "Змінити кількість", lowerPanel);  
 this.add(lowerPanel, "South");  
 }  
  
 public void setWorkWithProductUI(WorkWithProductUI workWithProductUI) {  
 this.workWithProductUI = workWithProductUI;  
 }  
  
  
 private void createButtonWithAndAddToPanel(JButton button, String buttonLabel, JPanel originPanel) {  
 JPanel buttonPanel = new JPanel();  
 buttonPanel.setLayout(new FlowLayout());  
 buttonPanel.setBackground(Color.*GRAY*);  
 button.setText(buttonLabel);  
 button.setHorizontalAlignment(JButton.*CENTER*);  
 button.addActionListener(this);  
 button.setPreferredSize(new Dimension(220, 70));  
 button.setFont(new Font("Default", Font.*BOLD*, 17));  
 buttonPanel.add(button);  
 originPanel.add(buttonPanel);  
 }  
  
 @Override  
 public void actionPerformed(ActionEvent e) {  
  
 this.setVisible(false);  
 if (e.getSource().equals(changeQuantityProduct)) {  
 int prodNumb = products.getSelectedIndex();  
 Good prod = productsList.get(prodNumb);  
 prod.setAmount(prod.getAmount() + (Integer) amount.getValue());  
 JOptionPane.*showMessageDialog*(null, "Товар змінено", "Успіх", JOptionPane.*INFORMATION\_MESSAGE*);  
  
 oldProductName.setText("");  
 rememOldProdName = "NO text";  
 this.setVisible(true);  
 } else {  
 workWithProductUI.returned();  
 }  
  
 }  
}

*/\*  
UI for reducing amount of a good  
 \*/*import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.util.ArrayList;  
  
public class RedQuantityOfProductUI extends JFrame implements ActionListener {  
 Storage storage;  
 WorkWithProductUI workWithProductUI;  
 JTextField oldProductName;  
 String rememOldProdName;  
 JSpinner amount;  
 JComboBox products;  
 JComboBox groups;  
 ArrayList<Group> groupsList;  
 ArrayList<Good> productsList;  
 JButton changeQuantityProduct;  
 JButton back;  
 Thread updateProdList;  
  
 public RedQuantityOfProductUI() {  
 super("Зменшити кількість");  
 this.setSize(700, 500);  
 this.setLayout(new BorderLayout());  
 this.setDefaultCloseOperation(*EXIT\_ON\_CLOSE*);  
 this.setLocationRelativeTo(null);  
 ImageIcon icon = new ImageIcon("lab2/Images/wareHouseIcon.png");  
 this.setIconImage(icon.getImage());  
  
 storage = Storage.*getInstance*();  
 productsList = storage.findGood("");  
 setWindow();  
 class listOfGoodsUpdateThread implements Runnable {  
 public void run() {  
 while (true) {  
 if (!oldProductName.getText().equals(rememOldProdName)) {  
 productsList = storage.findGood(oldProductName.getText());  
 products.removeAllItems();  
 for (int i = 0; i < productsList.size(); i++) {  
 products.addItem(productsList.get(i).getName());  
 }  
 rememOldProdName = oldProductName.getText();  
 }  
  
 try {  
 Thread.*sleep*(500);  
 } catch (InterruptedException e) {  
 System.*out*.println("Error");  
 }  
 }  
 }  
 }  
 updateProdList = new Thread(new listOfGoodsUpdateThread());  
 updateProdList.start();  
 this.setVisible(true);  
  
 }  
  
 private void setWindow() {  
 *//create upper part label with picture* addUpperPart();  
 *//create central part with buttons to deal with product* addCentralPart();  
 *//create lower part with back button* addLowerPart();  
  
 }  
  
 private void addUpperPart() {  
 JPanel upperPart = new JPanel(new BorderLayout());  
 upperPart.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 4));  
 ImageIcon productPicture = new ImageIcon("lab2/Images/minusProduct.png");*//add picture* JLabel productLabel = new JLabel(); *// Create a JLabel to display the image* productLabel.setIcon(productPicture);  
 productLabel.setHorizontalAlignment(JLabel.*CENTER*);  
 productLabel.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 2));  
 upperPart.add(productLabel, "West");  
  
 JLabel workWithProductLabel = new JLabel("Зменшення кількості товару"); *// Create a JLabel to display the name* workWithProductLabel.setHorizontalAlignment(JLabel.*CENTER*);  
 workWithProductLabel.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 2));  
 workWithProductLabel.setFont(new Font("Default", Font.*BOLD*, 17));  
 upperPart.add(workWithProductLabel, "Center");  
  
 this.add(upperPart, "North");  
 }  
  
 private void addCentralPart() {  
 JPanel centralPart = new JPanel(new GridLayout(0, 2));  
 centralPart.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 2));  
  
  
 JPanel centralLeft = new JPanel(new GridLayout(3, 0));  
 centralLeft.setBackground(Color.*LIGHT\_GRAY*);  
 oldProductName = new JTextField();  
 createTextFieldWithLabelWithAndAddToPanel(oldProductName, "Назва товару", centralLeft);  
  
 products = new JComboBox();  
 createComboBoxWithLabelWithAndAddToPanel(products, "Товари", centralLeft);  
  
 JPanel centralRight = new JPanel(new GridLayout(3, 0));  
 centralRight.setBackground(Color.*LIGHT\_GRAY*);  
  
 amount = new JSpinner();  
 createSpinnerWithLabelWithAndAddToPanel(amount, "Зменшити кількість на:", centralRight);  
  
 centralPart.add(centralLeft);  
 centralPart.add(centralRight);  
 this.add(centralPart, "Center");  
  
 }  
  
 private void createComboBoxWithLabelWithAndAddToPanel(JComboBox comboBox, String labelText, JPanel originPanel) {  
 JPanel labelAndTextPanel = new JPanel(new BorderLayout());  
 labelAndTextPanel.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 2));  
 labelAndTextPanel.setBackground(Color.*LIGHT\_GRAY*);  
 JLabel label = new JLabel(labelText);  
  
 labelAndTextPanel.add(label, "North");  
 if (comboBox.equals(groups)) {  
 for (int i = 0; i < groupsList.size(); i++) {  
 comboBox.addItem(groupsList.get(i).getName());  
 }  
 } else {  
 for (int i = 0; i < productsList.size(); i++) {  
 comboBox.addItem(productsList.get(i).getName());  
 }  
 }  
 labelAndTextPanel.add(comboBox, "Center");  
  
 originPanel.add(labelAndTextPanel);  
 }  
  
 private void createSpinnerWithLabelWithAndAddToPanel(JSpinner spinner, String labelText, JPanel originPanel) {  
 JPanel labelAndTextPanel = new JPanel(new BorderLayout());  
 labelAndTextPanel.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 2));  
 labelAndTextPanel.setBackground(Color.*LIGHT\_GRAY*);  
 JLabel label = new JLabel(labelText);  
  
 labelAndTextPanel.add(label, "North");  
 spinner.setModel(new SpinnerNumberModel(1, 1, Integer.*MAX\_VALUE*, 1));  
 labelAndTextPanel.add(spinner, "Center");  
  
 originPanel.add(labelAndTextPanel);  
 }  
  
 private void createTextFieldWithLabelWithAndAddToPanel(JTextField textField, String labelText, JPanel originPanel) {  
 JPanel labelAndTextPanel = new JPanel(new BorderLayout());  
 labelAndTextPanel.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 2));  
 labelAndTextPanel.setBackground(Color.*LIGHT\_GRAY*);  
 JLabel label = new JLabel(labelText);  
  
 labelAndTextPanel.add(label, "North");  
 labelAndTextPanel.add(textField, "Center");  
  
 originPanel.add(labelAndTextPanel);  
 }  
  
 private void addLowerPart() {  
 JPanel lowerPanel = new JPanel(new FlowLayout());  
 lowerPanel.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 4));  
 back = new JButton();  
 createButtonWithAndAddToPanel(back, "Повернутися назад", lowerPanel);  
 lowerPanel.setBackground(Color.*GRAY*);  
 changeQuantityProduct = new JButton();  
 createButtonWithAndAddToPanel(changeQuantityProduct, "Змінити кількість", lowerPanel);  
 this.add(lowerPanel, "South");  
 }  
  
 public void setWorkWithProductUI(WorkWithProductUI workWithProductUI) {  
 this.workWithProductUI = workWithProductUI;  
 }  
  
  
 private void createButtonWithAndAddToPanel(JButton button, String buttonLabel, JPanel originPanel) {  
 JPanel buttonPanel = new JPanel();  
 buttonPanel.setLayout(new FlowLayout());  
 buttonPanel.setBackground(Color.*GRAY*);  
 button.setText(buttonLabel);  
 button.setHorizontalAlignment(JButton.*CENTER*);  
 button.addActionListener(this);  
 button.setPreferredSize(new Dimension(220, 70));  
 button.setFont(new Font("Default", Font.*BOLD*, 17));  
 buttonPanel.add(button);  
 originPanel.add(buttonPanel);  
 }  
  
 @Override  
 public void actionPerformed(ActionEvent e) {  
  
 this.setVisible(false);  
 if (e.getSource().equals(changeQuantityProduct)) {  
 int prodNumb = products.getSelectedIndex();  
 Good prod = productsList.get(prodNumb);  
 int amounts = prod.getAmount() - (Integer) amount.getValue();  
 if (amounts < 0) {  
 amounts = 0;  
 }  
 prod.setAmount(amounts);  
 JOptionPane.*showMessageDialog*(null, "Товар змінено", "Успіх", JOptionPane.*INFORMATION\_MESSAGE*);  
  
 oldProductName.setText("");  
 rememOldProdName = "NO text";  
 this.setVisible(true);  
 } else {  
 workWithProductUI.returned();  
 }  
  
 }  
}