Національний університет “Києво-Могилянська академія”

**Звіт до лабораторної роботи №2**

З дисципліни: Вступ до програмування

**Автоматизоване робоче місце**

Виконали : Студенти ФІ,ІПЗ-1

Пащук Назар Михайлович

Буковський Станіслав Валерійович

Викладач:

Кирієнко Оксана Валентинівна

**Програма**

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.

**Завдання**

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.
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.
7. Реалізувати інтерфейс додавання товару, інтерфейс списання товару
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, загальна вартість товарів в групі товарів.
10. До роботи додати звіт про виконання роботи з описом розподілу ролей.

**Розподіл задач**

Буковський Станіслав Валерійович:

* Реалізація графічного інтерфейсу
* Діалог з користувачем

Пащук Назар Михайлович:

* Реалізація логіки програми

**Опис класів і методів**

1. Group.java: клас груп продуктів, що реалізують відповідний функціонал

addGroup(String groupName, String groupDescription): метод для додавання групи

editGroup(int index, String newGroupName, String newGroupDescription):

метод для редагування групи

deleteGroup(String groupName): метод для видалення групи

remove(String groupName):методи для видалення товарів при видаленні групи

sortByName(): метод для сортування груп за назвою

toString(): об’єкт в стрічку

set/get Groups(): геттери і сеттери

set/get GroupName(): геттери і сеттери

set/get GroupDescription(): геттери і сеттери

2. Product.java:

addProduct(String groupNameInProduct, String productName, String productDescription, String producer, int productQuantityOnStock, double productPrice):додати продукт

editProduct( String groupNameInProduct, String productName, String productDescription, String producer, int productQuantityOnStock, double productPrice):редагувати продукт

delete(int index):видалити продукт за індексом

toString(): об’єкт в стрічку

addToStock(String productNameToAdd,int quantity): для додавання продуктів на склад

writeOffFromStock(String productNameToWriteOff, int quantity):для списання продуктів з складу

totalProductCost(): загальна ціна певного продукта на складі

valueOfProductsOnStock():загальна ціна продуктів на складі

sortByName(): метод для сортування за іменем групи

get/set for groupNameInProduct, productName,productDescription, producer, productQuantityOnStock, productPrice: геттери і сеттери;

3. DataIO.java

readInfoFromFile():читає дані з файлу

readGroups():читає групи з файлу

readProducts():читає продукти з складу

writeInfoToFile(): записує інформацію в файл

writeGroups(): записує групу в файл

writeProducts():записує продукти в файл

4. ActionWithData.java

find(int choose, String whatToFind): знаходить файл або групу за стрічкою

findGroup(String whatToFind):знаходить групу за стрічкою

findProduct(String whatToFind):знаходить продукт за стрічкою

5. MainFrame.java

6. Main.java

7. OpenDialog.java

**Діаграми класів**

**Проблеми в роботі**

В процесі роботи виникло дві проблеми, а саме: погана швидкість зв’язку, через що важко було працювати в режимі онлайн та обмінюватись данними. Наступна проблема внаслідок війни на території України . Загалом всі інші проблеми були незначними.

**Висновки**

Написання цієї програми покращило нашу командну роботу, допомогло розвинути навички в програмуванні, покращити якість коду та пристосування до командної роботи. Загалом це був чудовий досвід для обох.

**Код програми**

package source;  
  
import java.util.ArrayList;  
import java.util.HashMap;  
import java.util.Locale;  
  
*/\*\*  
 \* Існує декілька груп товарів (наприклад: Продовольчі, непродовольчі...).  
 \* Група товарів містить наступні властивості - назва, опис.\*/*public class Group {  
  
  
 private String groupName;  
 private String groupDescription;  
  
 private static ArrayList<Group> *groups* = new ArrayList<>();  
 private ArrayList<Product> products = new ArrayList<>();  
  
 private Group(String groupName,String groupDescription){  
 this.groupName=groupName;  
 this.groupDescription=groupDescription;  
 }  
 public static void addGroup(String groupName, String groupDescription){  
 boolean unique = true;  
 for(Group group: *getGroups*()){  
 if(group.getGroupName().equals(groupName)){  
 unique = false;  
 System.*err*.println("Try to add non unique group)");  
 break;  
 }  
 }  
 if(unique)  
 *groups*.add(new Group(groupName,groupDescription));  
 }  
 public void editGroup(int index,String newGroupName, String newGroupDescription){  
 boolean unique = true;  
 for(Group group: *getGroups*()){  
 if(group.getGroupName().equals(newGroupName)){  
 unique = false;  
 System.*err*.println("Try to change group name for non unique");  
 break;  
 }  
 }  
 if(unique){  
 products = Product.*getProducts*();  
 String oldName = *groups*.get(index).getGroupName();  
 for(int i = 0; i<products.size(); i++){  
 if(products.get(i).getGroupNameInProduct().equals(oldName)){  
 products.get(i).setGroupNameInProduct(newGroupName);  
 }  
 }  
 Product.*setProducts*(products);  
 *groups*.get(index).setGroupName(newGroupName);  
 }  
 *groups*.get(index).setGroupDescription(newGroupDescription);  
 }  
 public void deleteGroup(String groupName){  
 for(Group group: *groups*){  
 if(group.getGroupName().equals(groupName)){  
 *groups*.remove(group);  
 remove(group.getGroupName());  
 }  
 }  
 }  
 */\*\*Use when you remove group to remove all products in this group\*/* public void remove(String groupName){  
 products = Product.*getProducts*();  
 for(int i=0; i<products.size(); i++)  
 if(groupName.equals(products.get(i).getGroupNameInProduct()))  
 products.remove(i);  
 Product.*setProducts*(products);  
 }  
 private static void sortByName(){  
 Group current;  
 Group previous;  
 for(int i=0; i<*groups*.size(); i++){  
 for(int j=1;j< *groups*.size() - i; j++){  
 current = *groups*.get(j);  
 previous = *groups*.get(j-1);  
 int res = previous.getGroupName().toLowerCase().compareTo(current.getGroupName().toLowerCase());  
 if(res > 0){  
 Group temp = previous;  
 *groups*.set(j-1,current);  
 *groups*.set(j,previous);  
 }  
 }  
 }  
 }  
 public String toString(){  
 return "\nGroup: "+getGroupName()+  
 "\nInfo: "+getGroupDescription();  
 }  
  
  
 */\*\*  
 \* block of getter and setter for fields  
 \* \*/* public static ArrayList<Group> getGroups(){  
 *sortByName*();  
 return *groups*;  
 }  
 public void setGroups(ArrayList<Group> groups) {  
 this.*groups* = groups;  
 }  
  
 public String getGroupName() {return groupName;}  
 public void setGroupName(String groupName) {this.groupName = groupName;}  
  
 public String getGroupDescription() {return groupDescription;}  
 public void setGroupDescription(String groupDescription) {this.groupDescription = groupDescription;}  
}

package source;  
  
import java.util.ArrayList;  
  
*/\*\*  
 \* 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.  
 \* \*/*public class Product {  
  
 private String groupNameInProduct;  
 private String productName;  
 private String productDescription;  
 private String producer;  
 private int productQuantityOnStock;  
 private double productPrice;  
  
 private static ArrayList<Product> *products* = new ArrayList<>();  
  
 private Product(String groupNameInProduct, String productName, String productDescription, String producer, int productQuantityOnStock, double productPrice) {  
 this.groupNameInProduct=groupNameInProduct;  
 this.productName=productName;  
 this.productDescription=productDescription;  
 this.producer=producer;  
 this.productQuantityOnStock=productQuantityOnStock;  
 this.productPrice=productPrice;  
 }  
 public static void addProduct(String groupNameInProduct, String productName, String productDescription, String producer, int productQuantityOnStock, double productPrice){  
 boolean unique = true;  
 for(Product product:*products*){  
 if(product.getProductName().equals(productName)){  
 unique = false;  
 System.*err*.println("Try to add non unique product");  
 break;  
 }  
 }  
 if(unique && productPrice>0 && productQuantityOnStock>=0)  
 *products*.add(new Product(groupNameInProduct,productName,productDescription,producer,productQuantityOnStock,productPrice));  
 else  
 System.*err*.println("Try to add product with (lower than 0 price or negative quantity)");  
 }  
 public void editProduct( String groupNameInProduct, String productName, String productDescription, String producer, int productQuantityOnStock, double productPrice){  
 boolean uniqueName = true;  
 boolean groupExist = false;  
 for(Product product: *products*){  
 if(product.getProductName().equals(productName)){  
 uniqueName = false;  
 System.*err*.println("Try to change product name to non unique");  
 break;  
 }  
 }  
 if(uniqueName)  
 setProductName(productName);  
 for(Group group: Group.*getGroups*()){  
 if(group.getGroupName().equals(groupNameInProduct)){  
 groupExist = true;  
 break;  
 }  
 }  
 if(groupExist)  
 setGroupNameInProduct(groupNameInProduct);  
 else  
 System.*err*.println("Try to change to non existing group");  
 if(productQuantityOnStock>0)  
 setProductQuantityOnStock(productQuantityOnStock);  
 else  
 System.*err*.println("Try to set incorrect quantity to product");  
 setProducer(producer);  
 setProductDescription(productDescription);  
 if(productPrice>0)  
 setProductPrice(productPrice);  
 else  
 System.*err*.println("Try to set incorrect price");  
  
 }  
 public void delete(int index){  
 *products*.remove(index);  
 }  
  
 */\*\*toString() in Product.class\*/* public String toString(){  
 return "\n#"+groupNameInProduct+  
 "\nНазва: "+productName+  
 "\nОпис: "+productDescription+  
 "\nВиробник: "+producer+  
 "\nНа складі: "+productQuantityOnStock+  
 "\nЦіна за штуку: "+productPrice+  
 "\nЗагальна вартість: "+totalProductCost();  
 }  
 */\*\*  
 \* Method to add product on stock(change quantity)  
 \* \*/* public void addToStock(String productNameToAdd,int quantity){  
 for(int i=0;i<*getProducts*().size();i++){  
 if(productNameToAdd.equals(*products*.get(i).getProductName())){  
 *products*.get(i).setProductQuantityOnStock(getProductQuantityOnStock()+quantity);  
 break;  
 }  
 }  
 }  
 */\*\*  
 \* Method to get product from stock(change quantity)  
 \* \*/* public void writeOffFromStock(String productNameToWriteOff, int quantity){  
 for(int i=0;i<*getProducts*().size();i++){  
 if(productNameToWriteOff.equals(*products*.get(i).getProductName())){  
 if(*products*.get(i).getProductQuantityOnStock() >= quantity)  
 *products*.get(i).setProductQuantityOnStock(getProductQuantityOnStock()+quantity);  
 else  
 System.*err*.println("Try to write off "+quantity+" product(On stock:"+*products*.get(i).getProductQuantityOnStock()+")");  
 break;  
 }  
 }  
 }  
  
 */\*\*  
 \* For count total product cost(one specific product)  
 \* \*/* public double totalProductCost(){  
 return getProductQuantityOnStock()\*getProductPrice();  
 }  
  
 */\*\*  
 \* For count total product value on stock(All products value)  
 \* \*/* public double valueOfProductsOnStock(){  
 double total = 0.0;  
 for(Product product: *getProducts*()){  
 total += product.getProductPrice()\*product.getProductQuantityOnStock();  
 }  
 return total;  
 }  
  
 private static void sortByName(){  
 Product current;  
 Product previous;  
 for(int i=0; i<*products*.size(); i++){  
 for(int j=1;j< *products*.size() - i; j++){  
 current = *products*.get(j);  
 previous = *products*.get(j-1);  
 int res = previous.getProductName().toLowerCase().compareTo(current.getProductName().toLowerCase());  
 if(res > 0){  
 Product temp = previous;  
 *products*.set(j-1,current);  
 *products*.set(j,previous);  
 }  
 }  
 }  
 }  
  
 */\*\*  
 \* getters and setter to fields  
 \* \*/* public static ArrayList<Product> getProducts(){  
 *sortByName*();  
 return *products*;  
 }  
 public static void setProducts(ArrayList<Product> newProducts){*products* = newProducts;}  
  
 public String getGroupNameInProduct() {return groupNameInProduct;}  
 public void setGroupNameInProduct(String groupNameInProduct) {this.groupNameInProduct = groupNameInProduct;}  
  
 public String getProductName() {return productName;}  
 public void setProductName(String productName) {this.productName = productName;}  
  
 public String getProductDescription() {return productDescription;}  
 public void setProductDescription(String productDescription) {this.productDescription = productDescription;}  
  
 public String getProducer() {return producer;}  
 public void setProducer(String producer) {this.producer = producer;}  
  
 public int getProductQuantityOnStock() {return productQuantityOnStock;}  
 public void setProductQuantityOnStock(int productQuantityOnStock) {this.productQuantityOnStock = productQuantityOnStock;}  
  
 public double getProductPrice() {return productPrice;}  
 public void setProductPrice(double productPrice) {this.productPrice = productPrice;}  
  
}

package source;  
  
import java.io.\*;  
import java.util.Scanner;  
  
public class DataIO {  
 private final static File *groupList* = new File("src/resources/groupList.txt");  
 private final static File *productList* = new File("src/resources/productList.txt");  
  
 */\*\*  
 \* Method that read groups and info from file(use when program begin)  
 \* \*/* public static void readInfoFromFile(){  
 *readGroups*();  
 *readProducts*();  
 }  
 public static void readGroups(){  
 try{  
 Scanner scanner = new Scanner(*groupList*);  
 while(scanner.hasNext()){  
 String[] groupInfo = scanner.nextLine().split("[#%]");  
 Group.*addGroup*(groupInfo[1],groupInfo[2]);  
 }  
 } catch(IOException e){  
 System.*err*.println("IO failed");  
 e.printStackTrace();  
 }  
 }  
 public static void readProducts(){  
 try{  
 Scanner scanner = new Scanner(*productList*);  
 while(scanner.hasNext()){  
 String[] productInfo = scanner.nextLine().split("[#%]");  
 try{  
 Product.*addProduct*(productInfo[1], productInfo[2], productInfo[3], productInfo[4],  
 Integer.*parseInt*(productInfo[5]), Double.*parseDouble*(productInfo[6]));  
 } catch(IllegalArgumentException e){  
 System.*err*.println("Wrong type when add product)");  
 e.printStackTrace();  
 }  
 }  
 } catch(IOException e){  
 System.*err*.println("IO failed");  
 }  
 }  
  
 */\*\*  
 \* Use at the end of program to write info to file  
 \* \*/* public static void writeInfoToFile(){  
 *writeGroups*();  
 *writeProducts*();  
 }  
 public static void writeGroups(){  
 try{  
 StringBuilder result;  
 BufferedWriter bw = new BufferedWriter(new FileWriter(*groupList*));  
 for(Group group : Group.*getGroups*()){  
 result = new StringBuilder("#");  
 result.append(group.getGroupName());  
 result.append("%");  
 result.append(group.getGroupDescription());  
 result.append("\n");  
 bw.write(String.*valueOf*(result));  
 }  
 bw.close();  
 }catch(IOException e){  
 System.*err*.println("File writing failed");  
 e.printStackTrace();  
 }  
 }  
 public static void writeProducts(){  
 try{  
 StringBuilder result;  
 BufferedWriter bw = new BufferedWriter(new FileWriter(*productList*));  
 for(Product product: Product.*getProducts*()){  
 result = new StringBuilder("#");  
 result.append(product.getProductName());  
 result.append("%");  
 result.append(product.getProductDescription());  
 result.append("%");  
 result.append(product.getProducer());  
 result.append("%");  
 result.append(product.getProductQuantityOnStock());  
 result.append("%");  
 result.append(product.getProductPrice());  
 result.append("\n");  
 bw.write(String.*valueOf*(result));  
 }  
 bw.close();  
 }catch(IOException e){  
 System.*err*.println("File writing failed");  
 e.printStackTrace();  
 }  
 }  
}

package source;  
  
import java.util.ArrayList;  
  
public class ActionWithData {  
 public ArrayList<?> find(int choose, String whatToFind){  
 if(choose == 1)  
 return findGroup(whatToFind);  
 if(choose == 2)  
 return findProduct(whatToFind);  
 return null;  
 }  
 public ArrayList<Group> findGroup(String whatToFind){  
 ArrayList<Group> result = new ArrayList<>();  
 for(Group group: Group.*getGroups*()){  
 if(group.getGroupName().contains(whatToFind)){  
 result.add(group);  
 continue;  
 }  
 if(group.getGroupDescription().contains(whatToFind))  
 result.add(group);  
 }  
 return result;  
 }  
 public ArrayList<Product> findProduct(String whatToFind){  
 ArrayList<Product> result = new ArrayList<>();  
 for(Product product: Product.*getProducts*()) {  
 if (product.getGroupNameInProduct().contains(whatToFind)) {  
 result.add(product);  
 continue;  
 }  
 if (product.getProductName().contains(whatToFind)){  
 result.add(product);  
 continue;  
 }  
 if (product.getProductDescription().contains(whatToFind)){  
 result.add(product);  
 continue;  
 }  
 }  
 return result;  
 }  
}