Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Інститут комп’ютерних наук та інформаційних технологій

Кафедра автоматизованих систем управління



**Звіт**

до лабораторної роботи № 6 з

курсу

*Прикладне програмувння*  на

тему:

**« Комплексна лабораторна робота »**

Виконав: студент ОІ-24

**Стащишин Юрій**

Прийняв:

**Чорненький В. Я.**

Львів – 2024

***Лабораторна робота № 6***

**Тема:** Комплексна лабораторна робота

**Мета роботи:** Навчитися використовувати патерн «Команда»

**Варіант - 13**

Реалізуйте функціонал програми. Обов’язково зверніть увагу на додаткові вимоги до програми (описані вище).

**Фургон кави.** Завантажити фургон певного об’єму вантажем на певну суму з різних видів кави, що може знаходитися у різних фізичних станах (зернова, мелена, розчинна в банках і пакетиках). Враховувати об’єм кави разом з упаковкою. Здійснити сортування товарів на основі співвідношення ціни та ваги. Знайти товар у фургоні, що відповідає заданому діапазону параметрів якості.

**Код програми**

package truck.command;  
  
import truck.model.Shop;  
import truck.model.Coffee;  
  
public class AddCoffeeCommand implements Command {  
 private Shop shop;  
 private Coffee coffee;  
  
 public AddCoffeeCommand(Shop shop) {  
 this.shop = shop;  
 }  
  
 // Метод для встановлення конкретної кави, яку потрібно додати  
 public void setCoffee(Coffee coffee) {  
 this.coffee = coffee;  
 }  
  
 @Override  
 public void execute() {  
 if (coffee != null) {  
 shop.addCoffee(coffee);  
 } else {  
 System.*out*.println("Необхідно встановити каву перед додаванням.");  
 }  
 }  
}

package truck.command;  
  
public interface Command {  
 void execute();  
}

package truck.command;  
  
public class ExitCommand implements Command{  
 @Override  
 public void execute(){  
 System.*out*.println("Вихід з програми...");  
 System.*exit*(0);  
 }  
}

package truck.command;  
  
import truck.model.Truck;  
import truck.model.Coffee;  
  
public class LoadTruckCommand implements Command{  
 private Truck truck;  
 private Coffee coffee;  
 private int quantity;  
  
 public LoadTruckCommand(Truck truckManager) {  
 this.truck = truckManager;  
 }  
  
 // Метод для встановлення кави, яку необхідно завантажити  
 public void setCoffee(Coffee coffee, int quantity) {  
 this.coffee = coffee;  
 this.quantity = quantity;  
 }  
  
 @Override  
 public void execute(){  
 if (coffee != null && quantity > 0) {  
 truck.loadTruck(coffee, quantity);  
 } else {  
 System.*out*.println("Необхідно встановити каву та кількість перед завантаженням.");  
 }  
 }  
}

package truck.command;  
  
import truck.model.Truck;  
  
public class SearchCoffeeCommand implements Command{  
 Truck truck;  
  
 public SearchCoffeeCommand(Truck truck) {  
 this.truck = truck;  
 }  
  
 @Override  
 public void execute(){  
 truck.searchCoffee();  
 }  
}

package truck.command;  
  
import truck.model.Shop;  
  
public class ShowAvailableCoffeeCommand implements Command{  
 Shop shop;  
  
 public ShowAvailableCoffeeCommand(Shop shop) {  
 this.shop = shop;  
 }  
  
 @Override  
 public void execute(){  
 shop.showAvailableCoffee();  
 }  
}

package truck.command;  
  
import truck.model.Truck;  
  
public class SortCoffeeCommand implements Command{  
 Truck truck;  
  
 public SortCoffeeCommand(Truck truck) {  
 this.truck = truck;  
 }  
  
 @Override  
 public void execute(){  
 truck.sortCoffee();  
 }  
}

package truck.model;  
import truck.command.\*;  
  
// Invoker  
public class Carrier {  
 Command load;  
 Command search;  
 Command sort;  
  
 public Carrier(Command load, Command search, Command sort) {  
 this.load = load;  
 this.search = search;  
 this.sort = sort;  
 }  
  
 public void loadTruck(){  
 load.execute();  
 }  
  
 public void searchCoffee(){  
 search.execute();  
 }  
  
 public void sortCoffee(){  
 sort.execute();  
 }  
}

package truck.model;  
  
public class Coffee {  
 private String name;  
 private double weight;  
 private double price;  
 private double volume;  
 private Package packaging;  
 private int quantity; // Кількість упаковок  
 private int quantityInTruck = 0;  
  
 public Coffee(String name, double weight, double price, double volume, Package packaging, int quantity) {  
 this.name = name;  
 this.weight = weight;  
 this.price = price;  
 this.volume = volume;  
 this.packaging = packaging;  
 this.quantity = quantity;  
 }  
  
  
 public String getName() {  
 return name;  
 }  
  
 public double getWeight() {  
 return weight;  
 }  
  
 public double getPrice() {  
 return price;  
 }  
  
 public double getVolume() {  
 return volume;  
 }  
  
 public Package getPackaging() {  
 return packaging;  
 }  
  
 public int getQuantity() {  
 return quantity;  
 }  
  
 public void setQuantity(int quantity) {  
 this.quantity = quantity;  
 }  
  
 public int getQuantityInTruck() {  
 return quantityInTruck;  
 }  
  
 public void setQuantityInTruck(int quantityInTruck) {  
 this.quantityInTruck = quantityInTruck;  
 }  
  
 public double getTotalVolume() {  
 return packaging.getVolume();  
 }  
  
 @Override  
 public String toString() {  
 return String.*format*("%s: вага = %.2f г, ціна = %.2f грн, об'єм = %.2f л, кількість упаковок = %d",  
 name, weight, price, packaging.getVolume(), quantity);  
 }  
  
 public String toStringInTruck() {  
 return String.*format*("%s: вага = %.2f г, ціна = %.2f грн, об'єм = %.2f л, кількість упаковок = %d",  
 name, weight, price, packaging.getVolume(), quantityInTruck);  
 }  
  
}

package truck.model;  
  
public class GrainCoffee extends Coffee{  
 public GrainCoffee(String name, double weight, double price, double volume, Package packaging, int quantity) {  
 super(name, weight, price, volume, packaging, quantity);  
 }  
}

package truck.model;  
  
public class GroundCoffee extends Coffee{  
 public GroundCoffee(String name, double weight, double price, double volume, Package packaging, int quantity) {  
 super(name, weight, price, volume, packaging, quantity);  
 }  
}

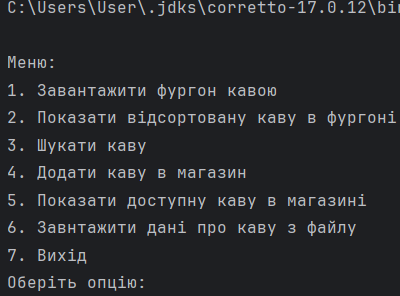
package truck.model;  
  
public class InstantCoffee extends Coffee{  
 public InstantCoffee(String name, double weight, double price, double volume, Package packaging, int quantity) {  
 super(name, weight, price, volume, packaging, quantity);  
 }  
}

package truck.model;  
  
// Клас для збереження даних про упаковку  
public class Package {  
 private String material;  
 private double weight;  
 private double volume;  
  
 public Package(String material, double weight, double volume) {  
 this.material = material;  
 this.weight = weight;  
 this.volume = volume;  
 }  
  
 public String getMaterial() {  
 return material;  
 }  
  
 public double getWeight() {  
 return weight;  
 }  
  
 public double getVolume() {  
 return volume;  
 }  
  
 @Override  
 public String toString() {  
 return String.*format*("Упаковка: матеріал = %s, вага = %.2f г, об'єм = %.2f л", material, weight, volume);  
 }  
}

package truck.model;  
import truck.command.\*;  
  
public class Seller {  
 Command add;  
 Command show;  
  
 public Seller(Command show, Command add) {  
 this.add = add;  
 this.show = show;  
 }  
  
 public void addCoffee(){  
 add.execute();  
 }  
  
 public void showCoffee(){  
 show.execute();  
 }  
}

package truck.model;  
  
import java.util.ArrayList;  
import java.util.List;  
import java.io.BufferedReader;  
import java.io.FileReader;  
import java.io.IOException;  
  
public class Shop {  
 private List<Coffee> availableCoffee;  
  
 public Shop() {  
 this.availableCoffee = new ArrayList<>();  
 initializeDefaultCoffees();  
 }  
  
 private void initializeDefaultCoffees() {  
 // Створення кількох видів кави за замовчуванням  
 Package package1 = new Package("Банка", 0.5, 1.0);  
 Package package2 = new Package("Пакет", 0.2, 0.5);  
  
 availableCoffee.add(new GrainCoffee("Зернова кава", 500, 100, 1, package1, 1000));  
 availableCoffee.add(new GroundCoffee("Мелена кава", 260, 50, 0.5, package2, 500));  
 availableCoffee.add(new InstantCoffee("Розчинна кава", 100, 30, 0.25, package2, 2000));  
 }  
  
 // Завантажує каву з файлу та додає її до списку доступних кав  
 public void loadCoffeesFromFile(String filename) {  
 try (BufferedReader br = new BufferedReader(new FileReader(filename))) {  
 String line;  
 while ((line = br.readLine()) != null) {  
 String[] parts = line.split(",");  
 if (parts.length == 8) {  
 String name = parts[0];  
 double weight = Double.*parseDouble*(parts[1]);  
 double price = Double.*parseDouble*(parts[2]);  
 double volume = Double.*parseDouble*(parts[3]);  
 String packagingMaterial = parts[4];  
 double packageWeight = Double.*parseDouble*(parts[5]);  
 double packageVolume = Double.*parseDouble*(parts[6]);  
 int quantity = Integer.*parseInt*(parts[7]);  
  
 Package packaging = new Package(packagingMaterial, packageWeight, packageVolume);  
 Coffee coffee = new GrainCoffee(name, weight, price, volume, packaging, quantity);  
 availableCoffee.add(coffee);  
 }  
 }  
 System.*out*.println("Каву з файлу добавлено");  
 } catch (IOException e) {  
 System.*out*.println("Помилка при зчитуванні файлу: " + e.getMessage());  
 }  
 }  
  
 // Додає новий об'єкт кави до списку доступних кав  
 public void addCoffee(Coffee coffee){  
 availableCoffee.add(coffee);  
 System.*out*.println("Додана кава: " + coffee);  
 }  
  
 // Виводить список доступної кави  
 public void showAvailableCoffee(){  
 if (availableCoffee.isEmpty()) {  
 System.*out*.println("Кава відсутня.");  
 } else {  
 System.*out*.println("Кава на вибір:");  
 for (Coffee coffee : availableCoffee) {  
 System.*out*.println(coffee);  
 }  
 }  
 }  
  
 public List<Coffee> getAvailableCoffee() {  
 return availableCoffee;  
 }  
}

package truck.model;  
  
import java.util.\*;  
  
// Reciver  
public class Truck {  
 private List<Coffee> loadedCoffees;  
 private double maxVolume; // в літрах (1м^3 = 1000л)  
 private double currentVolume;  
  
 public Truck(double maxVolume) {  
 this.loadedCoffees = new ArrayList<>();  
 this.maxVolume = maxVolume;  
 this.currentVolume = 0;  
 }  
  
 // Метод для завантаження фургону кавою  
 public void loadTruck(Coffee coffee, int quantity){  
 double coffeeVolume = coffee.getTotalVolume() \* quantity;  
  
 if (currentVolume + coffeeVolume <= maxVolume) {  
 int currentQuantity = coffee.getQuantity() - quantity;  
 coffee.setQuantityInTruck(quantity + coffee.getQuantityInTruck());  
 loadedCoffees.add(coffee);  
 currentVolume += coffeeVolume;  
 System.out.println(quantity + " упаковок " + coffee.getName() + " завантажено у фургон. Поточний об'єм: " + currentVolume + "/" + maxVolume);  
 coffee.setQuantity(currentQuantity);  
 } else {  
 System.out.println("Недостатньо місця у фургоні для " + quantity + " упаковок " + coffee.getName() + ". Поточний об'єм: " + currentVolume + "/" + maxVolume);  
 }  
 }  
  
 // Метод для сортування кави на основі співвідношення ціни та ваги  
 public void sortCoffee(){  
 if (loadedCoffees.isEmpty()) {  
 System.out.println("Фургон порожній. Немає кави для сортування.");  
 return;  
 }  
  
 Collections.*sort*(loadedCoffees, new Comparator<Coffee>() {  
 @Override  
 public int compare(Coffee c1, Coffee c2) {  
 double ratio1 = c1.getPrice() / c1.getWeight();  
 double ratio2 = c2.getPrice() / c2.getWeight();  
 return Double.*compare*(ratio1, ratio2);  
 }  
 });  
  
  
 System.*out*.println("Кава відсортована за співвідношенням ціни до ваги:");  
 for (Coffee coffee : loadedCoffees) {  
 System.*out*.println(coffee.toStringInTruck());  
 }  
 }  
  
 // Метод для знаходження товару у фургоні  
 public void searchCoffee(){  
 if (loadedCoffees.isEmpty()) {  
 System.*out*.println("Фургон порожній. Немає кави для пошуку.");  
 return;  
 }  
  
 Scanner scanner = new Scanner(System.*in*);  
 System.*out*.print("Введіть назву кави для пошуку: ");  
 String coffeeName = scanner.nextLine();  
  
 boolean found = false;  
 for (Coffee coffee : loadedCoffees) {  
 if (coffee.getName().equalsIgnoreCase(coffeeName)) {  
 System.*out*.println("Кава знайдена: " + coffee);  
 found = true;  
 break;  
 }  
 }  
  
 if (!found) {  
 System.out.println("Кава з назвою \"" + coffeeName + "\" не знайдена у фургоні.");  
 }  
 }  
  
}

package truck;  
  
import truck.command.\*;  
import truck.model.\*;  
import truck.model.Package;  
  
import java.util.InputMismatchException;  
import java.util.List;  
import java.util.Scanner;  
  
public class ConsoleMenu{  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 Truck truck = new Truck(2000);  
 Shop shop = new Shop();  
  
 LoadTruckCommand loadCommand = new LoadTruckCommand(truck);  
 Command searchCommand = new SearchCoffeeCommand(truck);  
 Command sortCommand = new SortCoffeeCommand(truck);  
 AddCoffeeCommand addCoffeeCommand = new AddCoffeeCommand(shop);  
 Command showCoffeeCommand = new ShowAvailableCoffeeCommand(shop);  
 Command exitCommand = new ExitCommand();  
  
 Carrier carrier = new Carrier(loadCommand, searchCommand, sortCommand);  
 Seller seller = new Seller(showCoffeeCommand, addCoffeeCommand);  
  
 boolean running = true;  
 while (running) {  
 System.*out*.println();  
 System.*out*.println("Меню:");  
 System.*out*.println("1. Завантажити фургон кавою");  
 System.*out*.println("2. Показати відсортовану каву в фургоні");  
 System.*out*.println("3. Шукати каву");  
 System.*out*.println("4. Додати каву в магазин");  
 System.*out*.println("5. Показати доступну каву в магазині");  
 System.*out*.println("6. Завнтажити дані про каву з файлу");  
 System.*out*.println("7. Вихід");  
  
 System.*out*.print("Оберіть опцію: ");  
  
 int choice = -1;  
 while (true) {  
 try {  
 choice = scanner.nextInt();  
 if (choice < 1 || choice > 7) {  
 System.*out*.println("Будь ласка, виберіть число від 1 до 7.");  
 } else {  
 break;  
 }  
 } catch (InputMismatchException e) {  
 System.*out*.println("Помилка: Введіть коректне число.");  
 scanner.next();  
 }  
 }  
  
 switch (choice) {  
 case 1:  
 Coffee load = *selectCoffee*(scanner, shop);  
 if (load != null) {  
 System.*out*.print("Введіть суму на яку хочете зробити замовлення: ");  
 int sum = scanner.nextInt();  
 int quantity = (int) (sum / load.getPrice());  
 if (quantity > load.getQuantity()){  
 System.*out*.println("Недостатньо кави в магазині");  
 break;  
 }  
 loadCommand.setCoffee(load, quantity);  
 loadCommand.execute();  
 } else {  
 System.*out*.println("Помилка: Кава не вибрана.");  
 }  
 break;  
 case 2:  
 carrier.sortCoffee();  
 break;  
 case 3:  
 carrier.searchCoffee();  
 break;  
 case 4:  
 Coffee coffee = *createCoffee*(scanner);  
 if (coffee != null) {  
 addCoffeeCommand.setCoffee(coffee);  
 seller.addCoffee();  
 }  
 break;  
 case 5:  
 seller.showCoffee();  
 break;  
 case 6:  
 shop.loadCoffeesFromFile("coffees.txt");  
 break;  
 case 7:  
 exitCommand.execute();  
 break;  
 default:  
 System.*out*.println("Невірний вибір, спробуйте ще раз.");  
 }  
 }  
  
 scanner.close();  
 }  
  
 private static Coffee selectCoffee(Scanner scanner, Shop shop){  
 // Виведення списку доступної кави  
 System.*out*.println("Доступна кава в магазині:");  
 List<Coffee> availableCoffee = shop.getAvailableCoffee();  
  
 if (availableCoffee.isEmpty()) {  
 System.*out*.println("Кава відсутня.");  
 return null;  
 }  
  
 // Відображення кави з індексами для вибору  
 for (int i = 0; i < availableCoffee.size(); i++) {  
 System.*out*.printf("%d. %s%n", i + 1, availableCoffee.get(i));  
 }  
  
 int choice = -1;  
 while (true) {  
 System.*out*.print("Оберіть номер кави для завантаження у фургон: ");  
 try {  
 choice = scanner.nextInt();  
 if (choice > 0 && choice <= availableCoffee.size()) {  
 return availableCoffee.get(choice - 1);  
 } else {  
 System.*out*.println("Невірний вибір. Будь ласка, виберіть номер від 1 до " + availableCoffee.size() + ".");  
 }  
 } catch (InputMismatchException e) {  
 System.*out*.println("Помилка: Введіть коректне число.");  
 scanner.next();  
 }  
 }  
 }  
  
 private static Coffee createCoffee(Scanner scanner) {  
 System.*out*.println("Виберіть тип кави: ");  
 System.*out*.println("1. Grain Coffee");  
 System.*out*.println("2. Ground Coffee");  
 System.*out*.println("3. Instant Coffee");  
  
 int typeChoice = -1;  
 while (true) {  
 try {  
 typeChoice = scanner.nextInt();  
 if (typeChoice < 1 || typeChoice > 3) {  
 System.*out*.println("Невірний вибір. Будь ласка, виберіть тип кави від 1 до 3.");  
 } else {  
 break; // Вихід з циклу, якщо введено коректне число  
 }  
 } catch (InputMismatchException e) {  
 System.*out*.println("Помилка: Введіть коректне число.");  
 scanner.next(); // Очищення буфера  
 }  
 }  
  
 scanner.nextLine(); // Очищення буфера після введення числа  
  
 System.*out*.print("Введіть назву кави: ");  
 String name = scanner.nextLine();  
  
 double weight = -1;  
 while (weight < 0) {  
 System.*out*.print("Введіть вагу кави (в грамах): ");  
 try {  
 weight = scanner.nextDouble();  
 if (weight < 0) {  
 System.*out*.println("Вага не може бути негативною. Спробуйте ще раз.");  
 }  
 } catch (InputMismatchException e) {  
 System.*out*.println("Помилка: Введіть коректну вагу.");  
 scanner.next(); // Очищення буфера  
 }  
 }  
  
 double price = -1;  
 while (price < 0) {  
 System.*out*.print("Введіть ціну кави (в грн): ");  
 try {  
 price = scanner.nextDouble();  
 if (price < 0) {  
 System.*out*.println("Ціна не може бути негативною. Спробуйте ще раз.");  
 }  
 } catch (InputMismatchException e) {  
 System.*out*.println("Помилка: Введіть коректну ціну.");  
 scanner.next(); // Очищення буфера  
 }  
 }  
  
 double volume = -1;  
 while (volume < 0) {  
 System.*out*.print("Введіть об'єм кави (в літрах): ");  
 try {  
 volume = scanner.nextDouble();  
 if (volume < 0) {  
 System.*out*.println("Об'єм не може бути негативним. Спробуйте ще раз.");  
 }  
 } catch (InputMismatchException e) {  
 System.*out*.println("Помилка: Введіть коректний об'єм.");  
 scanner.next(); // Очищення буфера  
 }  
 }  
  
 // Вибір типу упаковки  
 System.*out*.println("Виберіть тип упаковки: ");  
 System.*out*.println("1. Банка");  
 System.*out*.println("2. Пакет");  
  
 int packagingChoice = -1;  
 while (true) {  
 try {  
 packagingChoice = scanner.nextInt();  
 if (packagingChoice < 1 || packagingChoice > 2) {  
 System.*out*.println("Невірний вибір типу упаковки. Будь ласка, виберіть 1 або 2.");  
 } else {  
 break; // Вихід з циклу, якщо введено коректне число  
 }  
 } catch (InputMismatchException e) {  
 System.*out*.println("Помилка: Введіть коректне число.");  
 scanner.next(); // Очищення буфера  
 }  
 }  
  
 String packagingMaterial = (packagingChoice == 1) ? "Банка" : "Пакет";  
  
 double packageWeight = -1;  
 while (packageWeight < 0) {  
 System.*out*.print("Введіть вагу упаковки (в грамах): ");  
 try {  
 packageWeight = scanner.nextDouble();  
 if (packageWeight < 0) {  
 System.*out*.println("Вага упаковки не може бути негативною. Спробуйте ще раз.");  
 }  
 } catch (InputMismatchException e) {  
 System.*out*.println("Помилка: Введіть коректну вагу упаковки.");  
 scanner.next(); // Очищення буфера  
 }  
 }  
  
 double packageVolume = -1;  
 while (packageVolume < 0) {  
 System.*out*.print("Введіть об'єм упаковки (в літрах): ");  
 try {  
 packageVolume = scanner.nextDouble();  
 if (packageVolume < 0) {  
 System.*out*.println("Об'єм упаковки не може бути негативним. Спробуйте ще раз.");  
 }  
 } catch (InputMismatchException e) {  
 System.*out*.println("Помилка: Введіть коректний об'єм упаковки.");  
 scanner.next(); // Очищення буфера  
 }  
 }  
  
 int quantity = -1;  
 while (quantity < 0) {  
 System.*out*.print("Введіть кількість упаковок: ");  
 try {  
 quantity = scanner.nextInt();  
 if (quantity < 0) {  
 System.*out*.println("Кількість не може бути негативною. Спробуйте ще раз.");  
 }  
 } catch (InputMismatchException e) {  
 System.*out*.println("Помилка: Введіть коректну кількість упаковок.");  
 scanner.next(); // Очищення буфера  
 }  
 }  
  
 Package packaging = new Package(packagingMaterial, packageWeight, packageVolume);  
  
 switch (typeChoice) {  
 case 1:  
 return new GrainCoffee(name, weight, price, volume, packaging, quantity);  
 case 2:  
 return new GroundCoffee(name, weight, price, volume, packaging, quantity);  
 case 3:  
 return new InstantCoffee(name, weight, price, volume, packaging, quantity);  
 default:  
 System.*out*.println("Невірний вибір типу кави.");  
 return null;  
 }  
 }  
  
}  


**Висновок**

В ході цієї лабораторної роботи я реалізував функціонал програми «Фургон з кавою» з використанням патерну «Команда». Розробив консольне меню з усіма потрібними пунктами.

Посилання на Git: https://github.com/Float36/labs