­­­­­­­­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 дисципліни

*Прикладне програмування*

на тему:

**“ Комплексна лабораторна робота ”**

Виконав: студент ОІ-24

**Зубач Руслан**

Прийняв: асистент каф. АСУ

Чорненький В.Я.

**Львів – 2024**

*Лабораторна робота №* *6*

**Завдання лабораторної роботи**

Створити консольну програму на мові програмування Java згідно до індивідуального завдання (ваш номер варіанту, згенерований випадковим чином, можна знайти у файлі «Варіанти завдань.pdf»). Саме завдання описане досить поверхово, якщо ви претендуєте на високу оцінку – його потрібно розширити. Наприклад, у завданні може бути сказано про «створити кілька квіток, з наявних квіток сформувати букет». Цілком логічно буде додати можливість видалити квітку, поміняти параметри існуючої квітки тощо. Оцінка за лабораторну роботу залежатиме і від глибини пропрацювання предметної області.

Окрім цього, програма повинна відповідати таким додатковим вимогам:

• Використовувати можливості ООП: класи, успадкування, поліморфізм, інкапсуляцію.

• Кожний клас повинен мати назву, яка повністю описує його суть, і інформативний склад.

• Успадкування потрібно використовувати тільки тоді, коли воно має сенс.

• При записі програми потрібно використовувати домовленості щодо оформлення коду java code convention.

• Класи потрібно грамотно розкласти по пакетах.

• Для збереження параметрів ініціалізації потрібно використовувати файли.

• Повинне бути консольне меню для вибору різних дій, наприклад: «завантажити з файлу», «довідка», «вихід» тощо.

Етапи виконання завдання (кожен етап вважається окремою лабораторною роботою):

1. Уважно пропрацюйте предметну область згідно вашого варіанту завдання. Опишіть технологічне завдання (який функціонал матиме ваша програма). Створіть Use Case та Class діаграми для вашого завдання.

2. Реалізуйте консольне меню до вашого завдання з використанням паттерну «Команда». На цьому етапі не потрібно реалізовувати функціонал конкретних пунктів меню – тільки створити «кістяк» вашої програми.

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

4. Покрийте вашу програму юніт тестами (coverage >=90%).

5. Додайте логгер з логуванням основних дій та виняткових ситуацій. При цьому далі логу мають зберігатися у файлі, а критичні помилки – додатково надсилатися на e-mail

**Варіант 1**

Квіти. Визначити ієрархію квітів. Створити кілька об’єктів-квітів. Зібрати букет (використовуючи аксесуари) з визначенням його вартості. Здійснити сортування квітів у букеті на основі рівня свіжості. Знайти квітку в букеті, що відповідає заданому діапазону довжин стебел.

**Код програми**

**Flower.java**

package flower;  
  
public abstract class Flower {  
 private String name;  
 private int freshness; *//в діапазоні 1-10* private int stemLengthCm;  
 private double price;  
  
 public Flower(String name, int freshness, int stemLengthCm, double price) {  
 setName(name);  
 setFreshness(freshness);  
 setStemLengthCm(stemLengthCm);  
 setPrice(price);  
 }  
  
 public String getName() {return name;}  
 public int getFreshness() {return freshness;}  
 public int getStemLengthCm() {return stemLengthCm;}  
 public double getPrice() {return price;}  
  
 private void setName(String name) {this.name = name;}  
 private void setFreshness(int freshness) {this.freshness = freshness;}  
 private void setStemLengthCm(int stemLengthCm) {this.stemLengthCm = stemLengthCm;}  
 private void setPrice(double price) {this.price = price;}  
  
 public String toString() {  
 return "Назва: " + name +  
 ", Cвіжість: " + freshness + "/10" +  
 ", Довжина Стебла: " + stemLengthCm + " сантиметрів" +  
 ", Ціна: " + price + " грн.";  
 }  
}

**Rose.java**

package flower;  
  
public class Rose extends Flower{  
 private int thornLengthCm;  
 private int fragranceLevel; *//в діапазоні 1-10* public Rose(){  
 super("Троянда", 10, 30, 25);  
 setThornLengthCm(1);  
 setFragranceLevel(7);  
 }  
  
 public int getThornLengthCm() {return thornLengthCm;}  
 public int getFragranceLevel() {return fragranceLevel;}  
  
 private void setThornLengthCm(int thornLengthCm) {this.thornLengthCm = thornLengthCm;}  
 private void setFragranceLevel(int fragranceLevel) {this.fragranceLevel = fragranceLevel;}  
  
 @Override  
 public String toString(){  
 return super.toString() +  
 ", Довжина шипів: " + thornLengthCm + " сантиметрів" +  
 ", Аромат: " + fragranceLevel + "/10";  
 }  
}

**Tulip.java**

package flower;  
  
public class Tulip extends Flower {  
 private String bulbSize;  
 private String petalShape;  
  
 public Tulip() {  
 super("Тюльпан", 9, 7, 15);  
 setBulbSize("Велика");  
 setPetalShape("Загострена");  
 }  
  
 public String getBulbSize() {return bulbSize;}  
 public String getPetalShape() {return petalShape;}  
  
 private void setBulbSize(String bulbSize) {this.bulbSize = bulbSize;}  
 private void setPetalShape(String petalShape) {this.petalShape = petalShape;}  
  
 public String toString(){  
 return super.toString() +  
 ", Розмір цибулини: " + bulbSize +  
 ", Форма пелюсток: " + petalShape;  
 }  
}

**Daisy.java**

package flower;  
  
public class Daisy extends Flower{  
 private String centerColor;  
 private String leafShape;  
  
 public Daisy(){  
 super("Ромашка", 7, 5, 2);  
 setCenterColor("Жовтий");  
 setLeafShape("Овал");  
 }  
  
 public String getCenterColor() {return centerColor;}  
 public String getLeafShape() {return leafShape;}  
  
 private void setCenterColor(String centerColor) {this.centerColor = centerColor;}  
 private void setLeafShape(String leafShape) {this.leafShape = leafShape;}  
  
 @Override  
 public String toString(){  
 return super.toString() +  
 ", Колір середини: " + centerColor +  
 ", Форма пелюсток: " + leafShape;  
 }  
}

**Accessory.java**

package accessory;  
  
public abstract class Accessory {  
 private String name;  
 private double price;  
  
 public Accessory(String name, double price) {  
 setName(name);  
 setPrice(price);  
 }  
  
 public String getName() {return name;}  
 public double getPrice() {return price;}  
  
 private void setName(String name) {this.name = name;}  
 private void setPrice(double price) {this.price = price;}  
  
 public String toString() {  
 return "Товар: " + name +  
 ", Вартість: " + price;  
 }  
}

**Card.java**

package accessory;  
  
public class Card extends Accessory {  
 private String message;  
 private String occasion;  
  
 public Card() {  
 super("Листівка", 20);  
 setMessage("Вітаю!");  
 setOccasion("День Народження");  
 }  
  
 public String getMessage() {return message;}  
 public String getOccasion() {return occasion;}  
  
 private void setMessage(String message) {this.message = message;}  
 private void setOccasion(String occasion) {this.occasion = occasion;}  
  
 @Override  
 public String toString() {  
 return super.toString() + ", Подія: " + occasion + ", Повідомлення: " + message;  
 }  
}

**Ribbon.java**

package accessory;  
  
public class Ribbon extends Accessory {  
 private String color;  
 private double lengthCm;  
  
 public Ribbon() {  
 super("Стрічка", 10);  
 setLengthCm(15);  
 setColor("Рожевий");  
 }  
  
 public String getColor() {return color;}  
 public double getLengthCm() {return lengthCm;}  
  
 private void setColor(String color) {this.color = color;}  
 private void setLengthCm(double length) {this.lengthCm = length;}  
  
 @Override  
 public String toString() {  
 return super.toString() + ", Колір: " + color + ", Довжина: " + lengthCm + " сантиметрів";  
 }  
}

**Wrapper.java**

package accessory;  
  
public class Wrapper extends Accessory {  
 private String material;  
 private String size;  
  
 public Wrapper() {  
 super("Упаковка", 30);  
 setMaterial("Папір");  
 setSize("Середній");  
 }  
  
 public String getMaterial() {return material;}  
 public String getSize() {return size;}  
  
 private void setMaterial(String material) {this.material = material;}  
 private void setSize(String size) {this.size = size;}  
  
 @Override  
 public String toString() {  
 return super.toString() + ", Матеріал: " + material + ", Розмір: " + size;  
 }  
}

**Bouquet.java**

package bouquet;  
  
import accessory.Accessory;  
import flower.Flower;  
  
import java.util.ArrayList;  
import java.util.Comparator;  
import java.util.Scanner;  
  
  
public class Bouquet {  
 private ArrayList<Flower> flowers;  
 private ArrayList<Accessory> accessories;  
  
 public Bouquet() {  
 this.flowers = new ArrayList<>();  
 this.accessories = new ArrayList<>();  
 }  
  
 public ArrayList<Flower> getFlowers() {return flowers;}  
 public ArrayList<Accessory> getAccessories() {return accessories;}  
  
 public void addFlower(Flower flower) {  
 flowers.add(flower);  
 System.*out*.println("Добавлено квітку");  
 }  
  
 public void removeFlower() {  
 int i = 0;  
 if(!flowers.isEmpty()) {  
 for(Flower flower : flowers) {  
 i++;  
 System.*out*.printf("%d. %s\n", i,flower);  
 }  
 Scanner sc = new Scanner(System.*in*);  
 int choice = sc.nextInt();  
 flowers.remove(flowers.get(choice-1));  
 System.*out*.println("Видалено квітку");  
 }  
 else{  
 System.*out*.println("Не знайдено квітів");  
 }  
 }  
  
 public void addAccessory(Accessory accessory) {  
 accessories.add(accessory);  
 System.*out*.println("Добавлено аксесуар");  
 }  
  
 public void removeAccessory() {  
 int i = 0;  
 if(!accessories.isEmpty()) {  
 for(Accessory accessory : accessories) {  
 i++;  
 System.*out*.printf("%d. %s\n", i,accessory);  
 }  
 Scanner sc = new Scanner(System.*in*);  
 System.*out*.println("Виберіть аксесуар для видалення:");  
 int choice = sc.nextInt();  
 accessories.remove(accessories.get(choice-1));  
 System.*out*.println("Видалено аксесуар");  
 }  
 else{  
 System.*out*.println("Не знайдено аксесуарів");  
 }  
 }  
  
 public void calculatePrice(){  
 double flowerPrice = 0;  
 double accessoriesPrice = 0;  
 double sum = 0;  
 if(!flowers.isEmpty()){  
 for (Flower flower : flowers) {  
 flowerPrice += flower.getPrice();  
 }  
 System.*out*.println("Вартість квітів: " + flowerPrice + " грн.");  
 }  
 if(!accessories.isEmpty()){  
 for (Accessory accessory : accessories) {  
 accessoriesPrice += accessory.getPrice();  
 }  
 System.*out*.println("Вартість аксесуарів: " + accessoriesPrice + " грн.");  
 }  
 sum = flowerPrice + accessoriesPrice;  
 if(sum > 0){  
 System.*out*.println("Загальна вартість букета: " + sum + " грн.");  
 }  
 else {  
 System.*out*.println("Не знайдено букет");  
 }  
  
 }  
  
 public void sortByFreshness(){  
 if(!flowers.isEmpty()){  
 flowers.sort(Comparator.*comparingInt*(Flower::getFreshness).reversed());  
 System.*out*.println("Квіти відсортовано за свіжістю.");  
 }  
 else{  
 System.*out*.println("Не знайдено букет");  
 }  
 }  
  
 public void findFlowerByStemLength(int minLength , int maxLength){  
 ArrayList<Flower> result = new ArrayList<>();  
 for (Flower flower : flowers) {  
 int stemLength = flower.getStemLengthCm();  
 if (stemLength >= minLength && stemLength <= maxLength) {  
 result.add(flower);  
 }  
 }  
 if(result.isEmpty()){  
 System.*out*.println("Не знайдено квітів в діапазоні");  
 }  
 else {  
 System.*out*.println("Знайдені квіти:");  
 for (Flower flower : result) {  
 System.*out*.println(flower);  
 }  
 }  
 }  
  
 public void displayBouquet(){  
 if(!flowers.isEmpty()){  
 System.*out*.println("Квіти у букеті: ");  
 for (Flower flower : flowers) {  
 System.*out*.println(flower);  
 }  
 }  
 else{  
 System.*out*.println("Не знайдено букет");  
 }  
 if(!accessories.isEmpty()){  
 System.*out*.println("Аксесуари у букеті: ");  
 for (Accessory accessory : accessories) {  
 System.*out*.println(accessory);  
 }  
 }  
 }  
}

**AddRemoveAccessoryCommand.java**

package command;  
  
import accessory.Card;  
import accessory.Ribbon;  
import accessory.Wrapper;  
import bouquet.Bouquet;  
  
import java.util.Scanner;  
  
public class AddRemoveAccessoryCommand implements Command {  
 private Bouquet bouquet;  
  
 public AddRemoveAccessoryCommand(Bouquet bouquet) {  
 this.bouquet = bouquet;  
 }  
  
 @Override  
 public void execute() {  
 Scanner sc = new Scanner(System.*in*);  
 System.*out*.println("1. Додати аксесуар");  
 System.*out*.println("2. Видалити аксесуар");  
 System.*out*.print("Оберіть опцію: ");  
 int option = sc.nextInt();  
 sc.nextLine();  
  
 switch (option) {  
 case 1:  
 System.*out*.println("1. Стрічка");  
 System.*out*.println("2. Пакування");  
 System.*out*.println("3. Листівка");  
 System.*out*.print("Оберіть аксесуар: ");  
 int choice = sc.nextInt();  
 sc.nextLine();  
 switch (choice) {  
 case 1:  
 bouquet.addAccessory(new Ribbon());  
 break;  
 case 2:  
 bouquet.addAccessory(new Wrapper());  
 break;  
 case 3:  
 bouquet.addAccessory(new Card());  
 break;  
 default:  
 System.*out*.println("Некоректний ввід");  
 }  
 break;  
 case 2:  
 bouquet.removeAccessory();  
 break;  
 default:  
 System.*out*.println("Некоректний ввід");  
 }  
 }  
}

**AddRemoveFlowerCommand.java**

package command;  
  
import bouquet.Bouquet;  
import flower.Daisy;  
import flower.Rose;  
import flower.Tulip;  
  
import java.util.Scanner;  
  
public class AddRemoveFlowerCommand implements Command {  
 private Bouquet bouquet;  
  
 public AddRemoveFlowerCommand(Bouquet bouquet) {  
 this.bouquet = bouquet;  
 }  
  
 @Override  
 public void execute() {  
 Scanner sc = new Scanner(System.*in*);  
 System.*out*.println("1. Додати квітку");  
 System.*out*.println("2. Видалити квітку");  
 System.*out*.print("Оберіть опцію: ");  
 int option = sc.nextInt();  
 sc.nextLine();  
  
 switch (option) {  
 case 1:  
 System.*out*.println("1. Троянда");  
 System.*out*.println("2. Ромашка");  
 System.*out*.println("3. Тюльпан");  
 System.*out*.print("Оберіть квітку: ");  
 int choice = sc.nextInt();  
 sc.nextLine();  
 switch (choice) {  
 case 1:  
 bouquet.addFlower(new Rose());  
 break;  
 case 2:  
 bouquet.addFlower(new Daisy());  
 break;  
 case 3:  
 bouquet.addFlower(new Tulip());  
 break;  
 default:  
 System.*out*.println("Некоректний ввід");  
 }  
 break;  
 case 2:  
 bouquet.removeFlower();  
 break;  
 default:  
 System.*out*.println("Некоректний ввід");  
 }  
 }  
}

**CalculateBouquetPriceCommand.java**

package command;  
  
import bouquet.Bouquet;  
  
public class CalculateBouquetPriceCommand implements Command {  
 private Bouquet bouquet;  
  
 public CalculateBouquetPriceCommand(Bouquet bouquet) {  
 this.bouquet = bouquet;  
 }  
 @Override  
 public void execute() {  
 bouquet.calculatePrice();  
 }  
}

**DisplayBouquetPriceCommand.java**

package command;  
  
import bouquet.Bouquet;  
  
public class DisplayBouquetPriceCommand implements Command {  
 private Bouquet bouquet;  
  
 public DisplayBouquetPriceCommand(Bouquet bouquet) {  
 this.bouquet = bouquet;  
 }  
  
 @Override  
 public void execute() {  
 bouquet.displayBouquet();  
 }  
}

**FindFLowerByStemLengthCommand.java**

package command;  
  
import bouquet.Bouquet;  
  
import java.util.Scanner;  
  
public class FindFLowerByStemLengthCommand implements Command {  
 private Bouquet bouquet;  
  
 public FindFLowerByStemLengthCommand(Bouquet bouquet) {  
 this.bouquet = bouquet;  
 }  
  
 @Override  
 public void execute() {  
 Scanner scanner = new Scanner(System.*in*);  
 System.*out*.println("Введіть мінімальну довжину стебла:");  
 int minLength = scanner.nextInt();  
 System.*out*.println("Введіть максимальну довжину стебла:");  
 int maxLength = scanner.nextInt();  
  
 bouquet.findFlowerByStemLength(minLength, maxLength);  
 }  
}

**SortByFreshnessCommand.java**

package command;  
  
import bouquet.Bouquet;  
  
public class SortByFreshnessCommand implements Command {  
 private Bouquet bouquet;  
  
 public SortByFreshnessCommand(Bouquet bouquet) {  
 this.bouquet = bouquet;  
 }  
  
 @Override  
 public void execute() {  
 bouquet.sortByFreshness();  
 }  
}

**Git репозиторій:** <https://github.com/wixieee/AP_LPNU/tree/master/LR4-8PP>

**Висновок**

На цій лабораторній роботі я закріпив можливості ООП: класи, успадкування, поліморфізм, інкапсуляцію