­­­­­­­­Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Інститут комп’ютерних наук та інформаційних технологій

Кафедра автоматизованих систем управління



**Звіт**

до лабораторної роботи № 7-8

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 №* *7-8*

**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.

**Лабораторна робота 7**

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

**AccessoryTest**

package accessory;  
  
import org.junit.jupiter.api.Test;  
import static org.junit.jupiter.api.Assertions.\*;  
  
class AccessoryTest {  
  
 @Test  
 void testCardProperties() {  
 Card card = new Card();  
 *assertEquals*("Листівка", card.getName());  
 *assertEquals*(20, card.getPrice());  
 *assertEquals*("Вітаю!", card.getMessage());  
 *assertEquals*("День Народження", card.getOccasion());  
 }  
  
 @Test  
 void testCardToString() {  
 Card card = new Card();  
 String expected = "Товар: Листівка, Вартість: 20.0, Подія: День Народження, Повідомлення: Вітаю!";  
 *assertEquals*(expected, card.toString());  
 }  
  
 @Test  
 void testRibbonProperties() {  
 Ribbon ribbon = new Ribbon();  
 *assertEquals*("Стрічка", ribbon.getName());  
 *assertEquals*(10, ribbon.getPrice());  
 *assertEquals*("Рожевий", ribbon.getColor());  
 *assertEquals*(15, ribbon.getLengthCm());  
 }  
  
 @Test  
 public void testRibbonToString() {  
 Ribbon ribbon = new Ribbon();  
 String expected = "Товар: Стрічка, Вартість: 10.0, Колір: Рожевий, Довжина: 15.0 сантиметрів";  
 *assertEquals*(expected, ribbon.toString());  
 }  
  
 @Test  
 void testWrapperProperties() {  
 Wrapper wrapper = new Wrapper();  
 *assertEquals*("Упаковка", wrapper.getName());  
 *assertEquals*(30, wrapper.getPrice());  
 *assertEquals*("Папір", wrapper.getMaterial());  
 *assertEquals*("Середній", wrapper.getSize());  
 }  
  
 @Test  
 public void testWrapperToString() {  
 Wrapper wrapper = new Wrapper();  
 String expected = "Товар: Упаковка, Вартість: 30.0, Матеріал: Папір, Розмір: Середній";  
 *assertEquals*(expected, wrapper.toString());  
 }  
}

**BouquetTest**

package bouquet;  
  
import accessory.Card;  
import flower.Daisy;  
import flower.Flower;  
import flower.Rose;  
import flower.Tulip;  
import org.junit.jupiter.api.BeforeEach;  
import org.junit.jupiter.api.Test;  
  
import java.util.ArrayList;  
  
import static org.junit.jupiter.api.Assertions.\*;  
  
class BouquetTest {  
 private Bouquet bouquet;  
  
 @BeforeEach  
 void setUp() {  
 bouquet = new Bouquet();  
 }  
  
 @Test  
 void testAddFlower(){  
 Rose rose = new Rose();  
 bouquet.addFlower(rose);  
 *assertTrue*(bouquet.getFlowers().contains(rose));  
 }  
  
 @Test  
 void testRemoveFlower(){  
 Rose rose = new Rose();  
 bouquet.addFlower(rose);  
 bouquet.removeFlower(0);  
 *assertFalse*(bouquet.getFlowers().contains(rose));  
 }  
  
 @Test  
 void testAddAccessory(){  
 Card card = new Card();  
 bouquet.addAccessory(card);  
 *assertTrue*(bouquet.getAccessories().contains(card));  
 }  
  
 @Test  
 void testRemoveAccessory(){  
 Card card = new Card();  
 bouquet.addAccessory(card);  
 bouquet.removeAccessory(0);  
 *assertFalse*(bouquet.getAccessories().contains(card));  
 }  
  
 @Test  
 void testCalculatePrice(){  
 bouquet.addFlower(new Rose());  
 bouquet.addAccessory(new Card());  
 double expectedPrice = 45.0;  
 *assertEquals*(expectedPrice, bouquet.calculatePrice());  
 }  
  
 @Test  
 void testFindFlowerByStemLength(){  
 Rose rose = new Rose();  
 Tulip tulip = new Tulip();  
 Daisy daisy = new Daisy();  
 bouquet.addFlower(rose);  
 bouquet.addFlower(tulip);  
 bouquet.addFlower(daisy);  
 ArrayList <Flower> expected = new ArrayList<>();  
 expected.add(tulip);  
 *assertEquals*(expected, bouquet.findFlowerByStemLength(6,7));  
 }  
  
 @Test  
 void testSortByFreshness(){  
 Rose rose = new Rose();  
 Tulip tulip = new Tulip();  
 Daisy daisy = new Daisy();  
 bouquet.addFlower(tulip);  
 bouquet.addFlower(daisy);  
 bouquet.addFlower(rose);  
 ArrayList <Flower> expected = new ArrayList<>();  
 expected.add(rose);  
 expected.add(tulip);  
 expected.add(daisy);  
 *assertEquals*(expected, bouquet.sortByFreshness());  
 }  
  
 @Test  
 void testSortByFreshnessEmptyBouquet() {  
 *assertTrue*(bouquet.sortByFreshness().isEmpty());  
 }  
  
 @Test  
 void testSortByFreshnessOneFlower() {  
 Rose rose = new Rose();  
 bouquet.addFlower(rose);  
 ArrayList<Flower> expected = new ArrayList<>();  
 expected.add(rose);  
 *assertEquals*(expected, bouquet.sortByFreshness());  
 }  
}

**CommandTest**

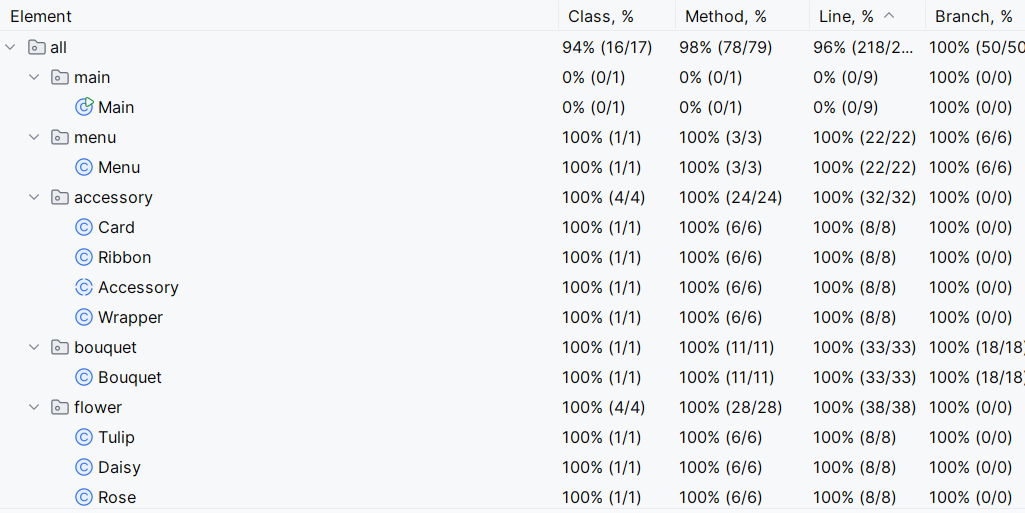
package command;  
  
import accessory.Card;  
import bouquet.Bouquet;  
import flower.Daisy;  
import flower.Rose;  
import org.junit.jupiter.api.BeforeEach;  
import org.junit.jupiter.api.Test;  
  
import java.io.ByteArrayInputStream;  
import java.io.ByteArrayOutputStream;  
import java.io.PrintStream;  
  
import static org.junit.jupiter.api.Assertions.*assertTrue*;  
  
class CommandTest {  
 private Bouquet bouquet;  
 private final ByteArrayOutputStream outputStreamCaptor = new ByteArrayOutputStream();  
  
 @BeforeEach  
 public void setUp() {  
 bouquet = new Bouquet();  
 System.*setOut*(new PrintStream(outputStreamCaptor));  
 }  
  
 @Test  
 public void testAddRemoveFlowerCommandIncorrectOption() {  
 AddRemoveFlowerCommand addRemoveFlowerCommand = new AddRemoveFlowerCommand(bouquet);  
  
 System.*setIn*(new ByteArrayInputStream("99\n".getBytes()));  
 addRemoveFlowerCommand.execute();  
  
 *assertTrue*(outputStreamCaptor.toString().contains("Некоректний ввід"));  
 }  
  
 @Test  
 public void testAddRoseCommand() {  
 AddRemoveFlowerCommand addRemoveFlowerCommand = new AddRemoveFlowerCommand(bouquet);  
  
 System.*setIn*(new ByteArrayInputStream("1\n1\n".getBytes()));  
 addRemoveFlowerCommand.execute();  
  
 *assertTrue*(outputStreamCaptor.toString().contains("Добавлено квітку"));  
 }  
  
 @Test  
 public void testAddDaisyCommand() {  
 AddRemoveFlowerCommand addRemoveFlowerCommand = new AddRemoveFlowerCommand(bouquet);  
  
 System.*setIn*(new ByteArrayInputStream("1\n2\n".getBytes()));  
 addRemoveFlowerCommand.execute();  
  
 *assertTrue*(outputStreamCaptor.toString().contains("Добавлено квітку"));  
 }  
  
 @Test  
 public void testAddTulipCommand() {  
 AddRemoveFlowerCommand addRemoveFlowerCommand = new AddRemoveFlowerCommand(bouquet);  
  
 System.*setIn*(new ByteArrayInputStream("1\n3\n".getBytes()));  
 addRemoveFlowerCommand.execute();  
  
 *assertTrue*(outputStreamCaptor.toString().contains("Добавлено квітку"));  
 }  
  
 @Test  
 public void testAddFlowerCommandIncorrectFlower() {  
 AddRemoveFlowerCommand addRemoveFlowerCommand = new AddRemoveFlowerCommand(bouquet);  
  
 System.*setIn*(new ByteArrayInputStream("1\n99\n".getBytes()));  
 addRemoveFlowerCommand.execute();  
  
 *assertTrue*(outputStreamCaptor.toString().contains("Некоректний ввід"));  
 }  
  
 @Test  
 public void testRemoveFlowerCommand() {  
 AddRemoveFlowerCommand addRemoveFlowerCommand = new AddRemoveFlowerCommand(bouquet);  
  
 bouquet.addFlower(new Rose());  
 System.*setIn*(new ByteArrayInputStream("2\n1\n".getBytes()));  
 addRemoveFlowerCommand.execute();  
  
 *assertTrue*(outputStreamCaptor.toString().contains("Видалено квітку"));  
 }  
  
 @Test  
 public void testRemoveFlowerCommandEmpty() {  
 AddRemoveFlowerCommand addRemoveFlowerCommand = new AddRemoveFlowerCommand(bouquet);  
  
 System.*setIn*(new ByteArrayInputStream("2\n".getBytes()));  
 addRemoveFlowerCommand.execute();  
  
 *assertTrue*(outputStreamCaptor.toString().contains("Не знайдено квітів"));  
 }  
  
  
  
 @Test  
 public void testAddRemoveAccessoryCommandIncorrectOption() {  
 AddRemoveAccessoryCommand addRemoveAccessoryCommand = new AddRemoveAccessoryCommand(bouquet);  
  
 System.*setIn*(new ByteArrayInputStream("99\n".getBytes()));  
 addRemoveAccessoryCommand.execute();  
  
 *assertTrue*(outputStreamCaptor.toString().contains("Некоректний ввід"));  
 }  
  
 @Test  
 public void testAddRibbonCommand() {  
 AddRemoveAccessoryCommand addRemoveAccessoryCommand = new AddRemoveAccessoryCommand(bouquet);  
  
 System.*setIn*(new ByteArrayInputStream("1\n1\n".getBytes()));  
 addRemoveAccessoryCommand.execute();  
  
 *assertTrue*(outputStreamCaptor.toString().contains("Добавлено аксесуар"));  
 }  
  
 @Test  
 public void testAddWrapperCommand() {  
 AddRemoveAccessoryCommand addRemoveAccessoryCommand = new AddRemoveAccessoryCommand(bouquet);  
  
 System.*setIn*(new ByteArrayInputStream("1\n2\n".getBytes()));  
 addRemoveAccessoryCommand.execute();  
  
 *assertTrue*(outputStreamCaptor.toString().contains("Добавлено аксесуар"));  
 }  
  
 @Test  
 public void testAddCardCommand() {  
 AddRemoveAccessoryCommand addRemoveAccessoryCommand = new AddRemoveAccessoryCommand(bouquet);  
  
 System.*setIn*(new ByteArrayInputStream("1\n3\n".getBytes()));  
 addRemoveAccessoryCommand.execute();  
  
 *assertTrue*(outputStreamCaptor.toString().contains("Добавлено аксесуар"));  
 }  
  
 @Test  
 public void testAddAccessoryCommandIncorrectAccessory() {  
 AddRemoveAccessoryCommand addRemoveAccessoryCommand = new AddRemoveAccessoryCommand(bouquet);  
  
 System.*setIn*(new ByteArrayInputStream("1\n99\n".getBytes()));  
 addRemoveAccessoryCommand.execute();  
  
 *assertTrue*(outputStreamCaptor.toString().contains("Некоректний ввід"));  
 }  
  
 @Test  
 public void testRemoveAccessoryCommand() {  
 AddRemoveAccessoryCommand addRemoveAccessoryCommand = new AddRemoveAccessoryCommand(bouquet);  
  
 bouquet.addAccessory(new Card());  
 System.*setIn*(new ByteArrayInputStream("2\n1\n".getBytes()));  
 addRemoveAccessoryCommand.execute();  
  
 *assertTrue*(outputStreamCaptor.toString().contains("Видалено аксесуар"));  
 }  
  
 @Test  
 public void testRemoveAccessoryCommandEmpty() {  
 AddRemoveAccessoryCommand addRemoveAccessoryCommand = new AddRemoveAccessoryCommand(bouquet);  
  
 System.*setIn*(new ByteArrayInputStream("2\n".getBytes()));  
 addRemoveAccessoryCommand.execute();  
  
 *assertTrue*(outputStreamCaptor.toString().contains("Не знайдено аксесуарів"));  
 }  
  
 @Test  
 public void testCalculateBouquetPriceCommandFilled() {  
 CalculateBouquetCommand calculateBouquetPriceCommand = new CalculateBouquetCommand(bouquet);  
 bouquet.addFlower(new Rose());  
 bouquet.addAccessory(new Card());  
 calculateBouquetPriceCommand.execute();  
  
 *assertTrue*(outputStreamCaptor.toString().contains("Загальна вартість букета: 45.0 грн."));  
 }  
  
 @Test  
 public void testCalculateBouquetPriceCommandEmpty() {  
 CalculateBouquetCommand calculateBouquetPriceCommand = new CalculateBouquetCommand(bouquet);  
 calculateBouquetPriceCommand.execute();  
  
 *assertTrue*(outputStreamCaptor.toString().contains("Не знайдено букет"));  
 }  
  
 @Test  
 public void testSortByFreshnessCommandEmpty() {  
 SortByFreshnessCommand sortByFreshnessCommand = new SortByFreshnessCommand(bouquet);  
  
 sortByFreshnessCommand.execute();  
  
 *assertTrue*(outputStreamCaptor.toString().contains("Не знайдено букет"));  
 }  
  
 @Test  
 public void testSortByFreshnessCommandFilled() {  
 SortByFreshnessCommand sortByFreshnessCommand = new SortByFreshnessCommand(bouquet);  
 bouquet.addFlower(new Rose());  
 bouquet.addFlower(new Daisy());  
 sortByFreshnessCommand.execute();  
  
 *assertTrue*(outputStreamCaptor.toString().contains("Квіти відсортовано за свіжістю."));  
 }  
  
 @Test  
 public void testFindFlowerByStemLengthCommandFalse() {  
 FindFlowerByStemLengthCommand findFlowerByStemLengthCommand = new FindFlowerByStemLengthCommand(bouquet);  
  
 System.*setIn*(new ByteArrayInputStream("6\n7\n".getBytes()));  
 findFlowerByStemLengthCommand.execute();  
  
 *assertTrue*(outputStreamCaptor.toString().contains("Не знайдено квітів в діапазоні"));  
 }  
  
 @Test  
 public void testFindFlowerByStemLengthCommandTrue() {  
 FindFlowerByStemLengthCommand findFlowerByStemLengthCommand = new FindFlowerByStemLengthCommand(bouquet);  
  
 bouquet.addFlower(new Rose());  
 System.*setIn*(new ByteArrayInputStream("10\n30\n".getBytes()));  
 findFlowerByStemLengthCommand.execute();  
  
 *assertTrue*(outputStreamCaptor.toString().contains("Знайдені квіти:"));  
 }  
  
 @Test  
 public void testDisplayBouquetCommandEmpty() {  
 DisplayBouquetCommand displayBouquetCommand = new DisplayBouquetCommand(bouquet);  
  
 displayBouquetCommand.execute();  
  
 String output = outputStreamCaptor.toString();  
 *assertTrue*(output.contains("Не знайдено букет"));  
 }  
  
 @Test  
 public void testDisplayBouquetCommandWithFlower() {  
 DisplayBouquetCommand displayBouquetCommand = new DisplayBouquetCommand(bouquet);  
 bouquet.addFlower(new Rose());  
 displayBouquetCommand.execute();  
  
 String output = outputStreamCaptor.toString();  
 *assertTrue*(output.contains("Квіти у букеті"));  
 }  
  
 @Test  
 public void testDisplayBouquetCommandWithAccessory() {  
 DisplayBouquetCommand displayBouquetCommand = new DisplayBouquetCommand(bouquet);  
 bouquet.addAccessory(new Card());  
 displayBouquetCommand.execute();  
  
 String output = outputStreamCaptor.toString();  
 *assertTrue*(output.contains("Аксесуари у букеті"));  
 }  
}

**FlowerTest**

package flower;  
  
import org.junit.jupiter.api.Test;  
  
import static org.junit.jupiter.api.Assertions.\*;  
  
class FlowerTest {  
 @Test  
 void testDaisyProperties() {  
 Daisy daisy = new Daisy();  
 *assertEquals*("Ромашка", daisy.getName());  
 *assertEquals*(7, daisy.getFreshness());  
 *assertEquals*(5, daisy.getStemLengthCm());  
 *assertEquals*(2.0, daisy.getPrice());  
 *assertEquals*("Жовтий", daisy.getCenterColor());  
 *assertEquals*("Овал", daisy.getLeafShape());  
 }  
  
 @Test  
 void testDaisyToString() {  
 Daisy daisy = new Daisy();  
 String expected = "Назва: Ромашка, Cвіжість: 7/10, Довжина Стебла: 5 сантиметрів, Ціна: 2.0 грн., Колір середини: Жовтий, Форма пелюсток: Овал";  
 *assertEquals*(expected, daisy.toString());  
 }  
  
 @Test  
 void testRoseProperties() {  
 Rose rose = new Rose();  
 *assertEquals*("Троянда", rose.getName());  
 *assertEquals*(10, rose.getFreshness());  
 *assertEquals*(30, rose.getStemLengthCm());  
 *assertEquals*(25.0, rose.getPrice());  
 *assertEquals*(1, rose.getThornLengthCm());  
 *assertEquals*(7, rose.getFragranceLevel());  
 }  
  
 @Test  
 void testRoseToString() {  
 Rose rose = new Rose();  
 String expected = "Назва: Троянда, Cвіжість: 10/10, Довжина Стебла: 30 сантиметрів, Ціна: 25.0 грн., Довжина шипів: 1 сантиметрів, Аромат: 7/10";  
 *assertEquals*(expected, rose.toString());  
 }  
  
 @Test  
 void testTulipProperties() {  
 Tulip tulip = new Tulip();  
 *assertEquals*("Тюльпан", tulip.getName());  
 *assertEquals*(9, tulip.getFreshness());  
 *assertEquals*(7, tulip.getStemLengthCm());  
 *assertEquals*(15.0, tulip.getPrice());  
 *assertEquals*("Велика", tulip.getBulbSize());  
 *assertEquals*("Загострена", tulip.getPetalShape());  
 }  
  
 @Test  
 void testTulipToString() {  
 Tulip tulip = new Tulip();  
 String expected = "Назва: Тюльпан, Cвіжість: 9/10, Довжина Стебла: 7 сантиметрів, Ціна: 15.0 грн., Розмір цибулини: Велика, Форма пелюсток: Загострена";  
 *assertEquals*(expected, tulip.toString());  
 }  
}

**MenuTest**

package menu;  
  
import bouquet.Bouquet;  
import command.DisplayBouquetCommand;  
import command.Command;  
import org.junit.jupiter.api.BeforeEach;  
import org.junit.jupiter.api.Test;  
  
import java.io.ByteArrayInputStream;  
import java.io.ByteArrayOutputStream;  
import java.io.PrintStream;  
  
import static org.junit.jupiter.api.Assertions.\*;  
  
class MenuTest {  
 private Menu menu;  
 private Command command;  
 private final ByteArrayOutputStream outputStreamCaptor = new ByteArrayOutputStream();  
  
 @BeforeEach  
 public void setUp() {  
 Bouquet bouquet = new Bouquet();  
 menu = new Menu();  
 command = new DisplayBouquetCommand(bouquet);  
 System.*setOut*(new PrintStream(outputStreamCaptor));  
 }  
  
 @Test  
 public void testAddCommand() {  
 menu.addCommand(6, command);  
 *assertEquals*(command, menu.getCommands().get(6));  
 }  
  
 @Test  
 public void testExecuteCommand\_DisplayBouquet() {  
 menu.addCommand(6, command);  
  
 System.*setIn*(new ByteArrayInputStream("6\n7\n".getBytes()));  
 menu.showMenu();  
  
 *assertTrue*(outputStreamCaptor.toString().contains("Не знайдено букет"));  
 }  
  
 @Test  
 public void testInvalidOptionHandling() {  
 System.*setIn*(new ByteArrayInputStream("99\n7\n".getBytes()));  
 menu.showMenu();  
  
 *assertTrue*(outputStreamCaptor.toString().contains("Невірний вибір. Спробуйте ще раз."));  
 }  
}

****

**Лабораторна робота 8**

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

**Файл конфігурації логера**

*<?*xml version="1.0" encoding="UTF-8"*?>*<Configuration status="WARN">  
<Appenders>  
 <Console name="ConsoleAppender" target="SYSTEM\_OUT">  
 <PatternLayout pattern="%msg%n"/>  
 </Console>  
  
 <File name="FileAppender" fileName="logs/app.log" append="true">  
 <PatternLayout pattern="%d{yyyy-MM-dd HH:mm:ss} [%t] %-5level %logger{36} - %msg%n"/>  
 </File>  
  
 <SMTP name="EmailAppender" subject="Critical Error in Application"  
 to="example@gmail.com" from="example@gmail.com"  
 smtpHost="smtp.gmail.com" smtpPort="465"  
 smtpUsername="example@gmail.com" smtpPassword="example"  
 smtpProtocol="smtps" smtpDebug="false">  
 <PatternLayout pattern="%d{yyyy-MM-dd HH:mm:ss} [%t] %-5level %logger{36} - %msg%n"/>  
 <Filters>  
 <ThresholdFilter level="ERROR" onMatch="ACCEPT" onMismatch="DENY"/>  
 </Filters>  
 </SMTP>  
</Appenders>  
  
<Loggers>  
 <Root level="info">  
 <AppenderRef ref="ConsoleAppender"/>  
 <AppenderRef ref="FileAppender"/>  
 <AppenderRef ref="EmailAppender"/>  
 </Root>  
</Loggers>  
</Configuration>

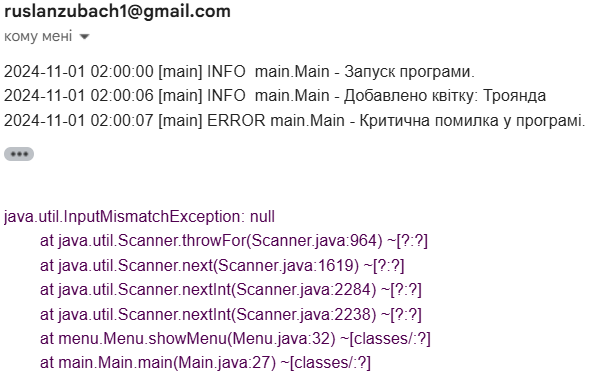
**Використання логера**

package main;  
  
import bouquet.Bouquet;  
import command.\*;  
import menu.Menu;  
import org.apache.logging.log4j.LogManager;  
import org.apache.logging.log4j.Logger;  
  
public class Main {  
 public static final Logger *logger* = LogManager.*getLogger*(Main.class);  
  
 public static void main(String[] args) {  
 *logger*.info("Запуск програми.");  
  
 try {  
 Bouquet bouquet = new Bouquet();  
  
 Menu menu = new Menu();  
  
 menu.addCommand(1, new AddRemoveFlowerCommand(bouquet));  
 menu.addCommand(2, new AddRemoveAccessoryCommand(bouquet));  
 menu.addCommand(3, new CalculateBouquetCommand(bouquet));  
 menu.addCommand(4, new SortByFreshnessCommand(bouquet));  
 menu.addCommand(5, new FindFlowerByStemLengthCommand(bouquet));  
 menu.addCommand(6, new DisplayBouquetCommand(bouquet));  
  
 menu.showMenu();  
  
 } catch (Exception e) {  
 *logger*.error("Критична помилка у програмі.", e);  
 }  
  
 *logger*.info("Програма завершена.");  
 }  
}

**Приклад нормальної роботи логера**

*2024-11-01 01:58:15* [main] INFO *main.Main* - Запуск програми.  
*2024-11-01 01:58:22* [main] INFO *main.Main* - Добавлено квітку: Троянда  
*2024-11-01 01:58:26* [main] INFO *main.Main* - Добавлено квітку: Ромашка  
*2024-11-01 01:58:32* [main] INFO *main.Main* - Видалено квітку: Троянда  
*2024-11-01 01:58:38* [main] INFO *main.Main* - Добавлено аксесуар: Пакування  
*2024-11-01 01:58:41* [main] INFO *main.Main* - Загальна вартість букета: *32*.*0* грн.  
*2024-11-01 01:58:55* [main] INFO *main.Main* - Програма завершена.

**Приклад критичної помилки**

****

**Git репозиторій:** [**https://github.com/wixieee/AP\_LPNU/tree/master/LR7-8PP**](https://github.com/wixieee/AP_LPNU/tree/master/LR7-8PP)

**Висновок**

На цій лабораторній роботі я опанував створення юніт-тестів та логування