Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Кафедра ЕОМ



Звіт

Лабораторна робота № 2

З дисципліни «Кросплатформні засоби програмування»

На тему: «Класи та пакети»

Виконав: ст. гр. КІ-304

Комісаров Д.О.

Перевірив:

доцент кафедри ЕОМ

Іванов Ю.С.

Львів 2025

Зміст

[Зміст 2](#_Toc210514861)

[Розділ 1. Мета роботи. 3](#_Toc210514862)

[Розділ 2. Теоретичні відомості. 4](#_Toc210514863)

[Розділ 3. Завдання (варіант № 9). 6](#_Toc210514864)

[Розділ 4. Хід роботи. 7](#_Toc210514865)

[Розділ 5. Відповіді на контрольні запитання. 10](#_Toc210514866)

[Висновки 12](#_Toc210514867)

[Список використаних джерел 13](#_Toc210514868)

[ДОДАТОК А. файл Plant.java 14](#_Toc210514869)

[ДОДАТОК Б. ФАЙЛ leaf.JAVA 16](#_Toc210514870)

[ДОДАТОК В. ФАЙЛ Root.JAVA 17](#_Toc210514871)

[ДОДАТОК Г. ФАЙЛ stem.JAVA 18](#_Toc210514872)

[ДОДАТОК Д. ФАЙЛ planttest.JAVA 19](#_Toc210514873)

[ДОДАТОК Е. вихідний матеріал 21](#_Toc210514874)

[ДОДАТОК Ж. фрагмент згенерованої документації 22](#_Toc210514875)

# Мета роботи.

Ознайомитися з процесом розробки класів та пакетів мовою Java.

# Теоретичні відомості.

Мова Java є повністю об’єктно-орієнтованою мовою програмування, тому вона

дозволяє писати програми лише з використанням об’єктно-орієнтованих парадигм програмування, що базуються на понятті класів. Необов’язковий специфікатор доступу public робить клас загальнодоступним. У кожному файлі з кодом програми може бути лише один загальнодоступний клас, ім’я якого співпадає з назвою файлу, та безліч класів без специфікатора public. Ініціалізація посилання на об’єкт класу здійснюється за допомогою оператора new і вказування конструктора, який має збудувати об’єкт. Одержаний в результаті цих операцій об’єкт розташується у області оперативної пам’яті що зветься ”куча”. Ініціалізація посилання на об’єкт класу за допомогою конструктора за замовчуванням має такий синтаксис:

назваПосилання = new НазваКонструтора();

Приклад ініціалізації посилання на об’єкт класу StartClass:

obj = new StartClass();

При створенні об’єктів дозволяється суміщати оголошення та ініціалізацію об’єктів, а також створювати анонімні об’єкти. Якщо посилання на об’єкт не посилається на жоден об’єкт, то йому слід присвоїти значення null. На відміну від полів-посилань на об’єкти, локальні змінні-посилання на об’єкти не ініціалізуються значенням null при оголошенні. Для них ініціалізацію посилання слід проводити явно. Метод – функція-член класу, яка призначена маніпулювати станом об’єкту класу. Методи можуть бути перевантаженими. Перевантаження методів відбувається шляхом вказування різної кількості параметрів та їх типів методам з однаковими назвами. Синтаксис оголошення методу наступний:

[СпецифікаторДоступу] [static] [final] Тип назваМетоду([параметри]) [throws класи]

{

[Тіло методу]

[return [значення]];

}

Конструктори, методи, та поля класу можуть бути відкритими (public),

закритими (private) та захищеними (protected), що визначається специфікатором

доступу. Специфікатор доступу public робить елемент класу загальнодоступним в межах пакету (набору класів, з яких складається програма).

Специфікатор доступу private робить елемент класу закритим (недоступним) для всіх зовнішніх відносно даного класу елементів програми (включаючи похідні класи). Специфікатор доступу protected робить елемент класу закритим (недоступним) для всіх зовнішніх відносно даного класу елементів програми, проте цей елемент буде загальнодоступним для похідних класів. Якщо будь-який елемент класу не має специфікатора доступу, то цей елемент автоматично стає відкритим та видимим у межах пакету (не плутати з public). Всі елементи класу, що оголошені без використання ключового слова static, належать об’єкту класу. Тобто, кожен об’єкт класу містить власну копію цих елементів класу. Ключове слово static робить поле або метод членом класу, а не об’єкту, тобто вони є спільними для всіх об’єктів класу. Оскільки клас існує завжди, на відміну від об’єктів, які створюються в процесі роботи програми, то статичні елементи класу доступні навіть тоді, коли ще не створено жодного об’єкту класу. Цей підхід використовується при написанні методу main з якого починається виконання консольної програми, бо на момент її запуску ще не існує жодного об’єкту.

# Завдання (варіант № 9).

1. Написати та налагодити програму на мові Java, що реалізує у вигляді класу предметну

область згідно варіанту (Рослина). Програма має задовольняти наступним вимогам:

• програма має розміщуватися в пакеті KI\_304.Komisarov.Lab2;

• клас має містити мінімум 3 поля, що є об’єктами класів, які описують складові

частини предметної області;

• клас має містити кілька конструкторів та мінімум 10 методів;

• для тестування і демонстрації роботи розробленого класу розробити клас-драйвер;

• методи класу мають вести протокол своєї діяльності, що записується у файл;

• розробити механізм коректного завершення роботи з файлом (не надіятися на

метод finalize());

• програма має володіти коментарями, які дозволять автоматично згенерувати

документацію до розробленого пакету.

2. Автоматично згенерувати документацію до розробленої програми.

3. Завантажити код на GitHub згідно методичних вказівок по роботі з GitHub.

4. Скласти звіт про виконану роботу з приведенням тексту програми, результату її

виконання та фрагменту згенерованої документації та завантажити його у ВНС.

5. Дати відповідь на контрольні запитання.

# Хід роботи.

1. Запустити на виконання середовище Eclipse IDE.

2. Дослідити тестову програму, що наведена в методичних вказівках.

3. Почати створення власного проекту. Для цього слід викликати підпункут меню File->New->Project… У вікні, що відкриється слід вибрати Java Project (рис. 4.1) та натиснути кнопку ”Next>”..

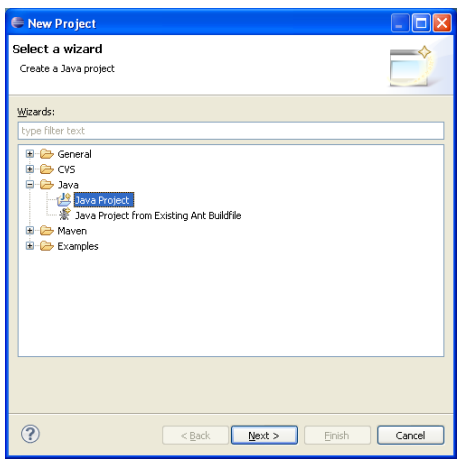


Рис. 4.1. Діалогове вікно вибору типу проекту в Eclipse IDE.

4. У вікні, що відкрилося, необхідно вказати назву нового проекту, місце розташування проекту, версію JRE на яке орієнтована програма, топологію проекту (Project layout) та натиснути кнопку ”Next>” (рис. 4.2).

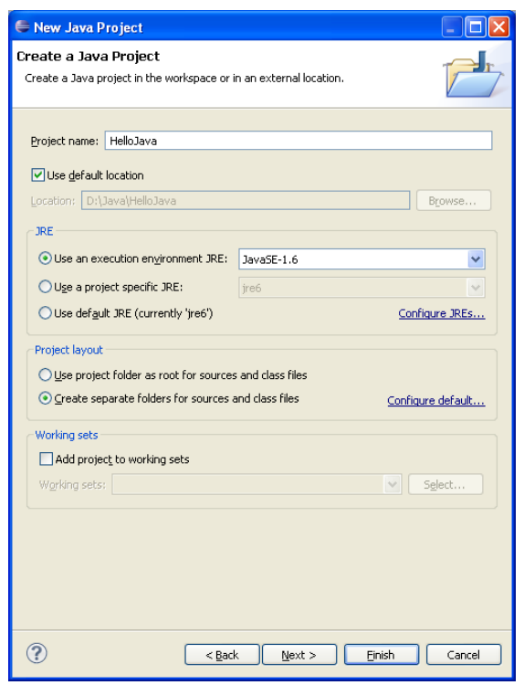


Рис. 4.2. Діалогове вікно створення нового проекту в Eclipse IDE.

5. У вікні, що відкрилося, натиснути кнопку ”Finish” (рис. 4.3).

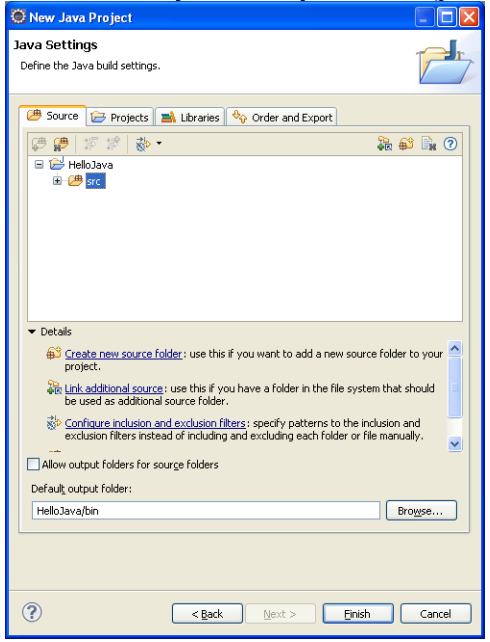


Рис. 4.3. Діалогове вікно налаштування властивостей побудови нового проекту в Eclipse IDE.

6. У вікні, що з’явилося (рис. 4.4), викликати підпункт меню File->New->File. У вікні, що відкрилося слід задати каталог, де розташовуватиметься новий файл з кодом, назву файлу з розширенням .java та натиснути кнопку ”Finish” (рис. 4.5).

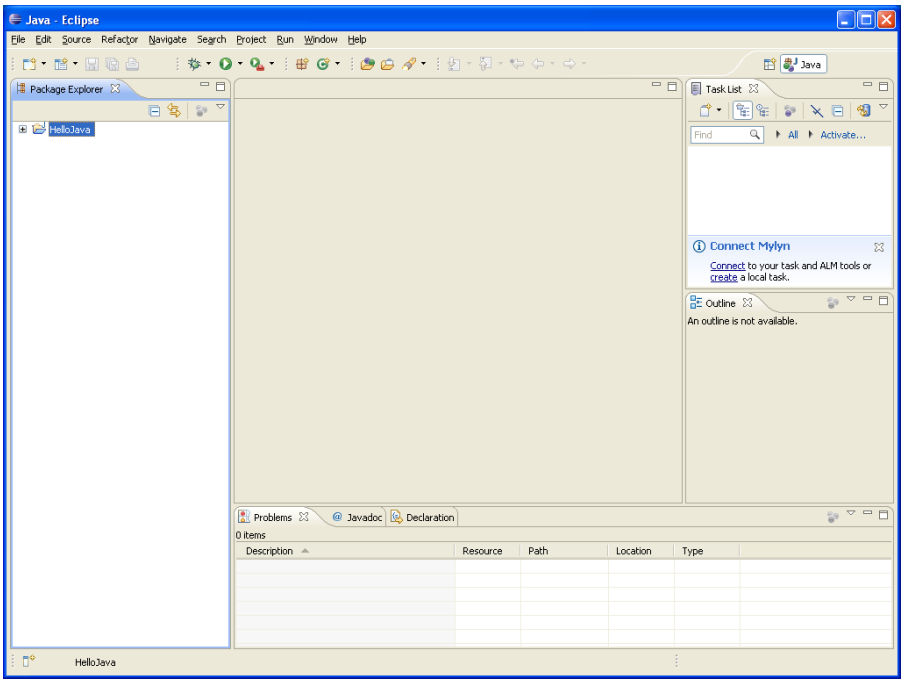


Рис. 4.4. Середовище Eclipse IDE з відкритим проектом Java.

7. У створеному файлі написати програму згідно завдання. Програма має обов’язково містити public клас, назва якого співпадає з назвою файлу.

8. Запустити програму на виконання. Для цього слід вибрати підпункт меню Run->Run.

9. Встановити точки переривання, запустити налагоджувач (Run->Debug) та покроково дослідити процес виконання програми (рис. 4.6).

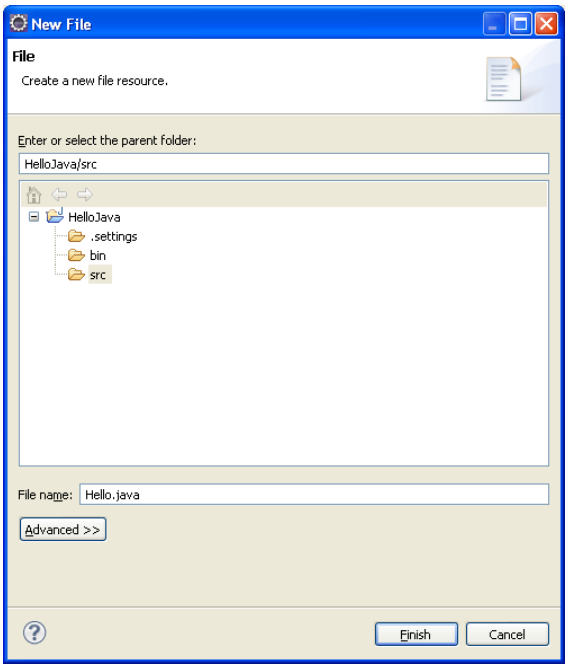


Рис. 4.5. Створення нового файлу.

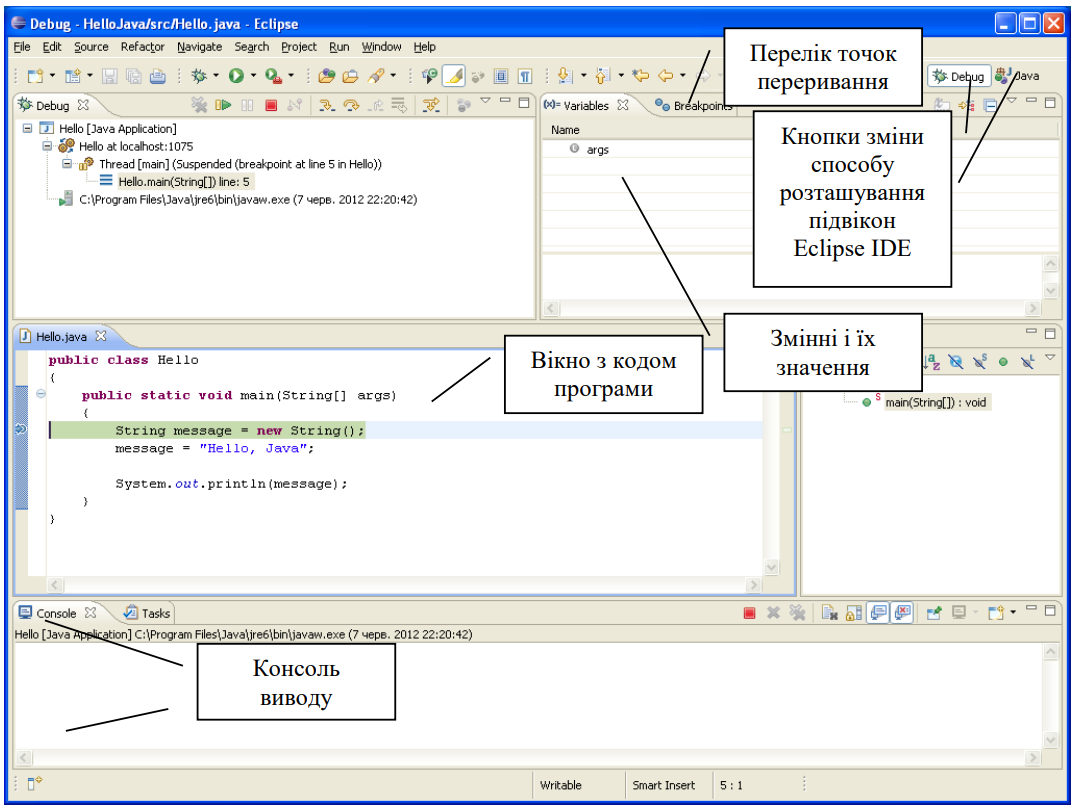


Рис. 4.6. Процес налагодження програми.

10. Автоматично згенерувати документацію у каталог \doc вашого проекту. Проаналізувати автоматично згенеровану документацію.

# Відповіді на контрольні запитання.

1. Синтаксис визначення класу

class ClassName {

// поля

// методи

}

2. Синтаксис визначення методу

returnType methodName(parameters) {

// тіло методу

}

3. Синтаксис оголошення поля

типНазви fieldName;

4. Як оголосити та ініціалізувати константне поле?

final типНазви CONST\_NAME = значення;

5. Які є способи ініціалізації полів?

 безпосередньо при оголошенні;

 у конструкторі;

 у блоці ініціалізації.

6. Синтаксис визначення конструктора.

ClassName(parameters) {

// ініціалізація

}

7. Синтаксис оголошення пакету

package packageName;

8. Як підключити до програми класи, що визначені в зовнішніх пакетах?

import packageName.ClassName;

// або

import packageName.\*;

9. В чому суть статичного імпорту пакетів?

Дозволяє використовувати статичні поля і методи класів без зазначення імені класу.

import static java.lang.Math.\*;

10. Які вимоги ставляться до файлів і каталогів при використанні пакетів?

 назва пакету має відповідати структурі каталогів;

 вихідний файл повинен зберігатися у папці, що відповідає пакету;

 у файлі може бути лише один публічний клас, ім’я файлу збігається з його ім’ям.

Висновки

У ході виконання лабораторної роботи було створено програму, яка містить в собі клас відповідно до індивідуального завдання. Для перевірки написаний ще один файл, у якому відбувається перевірка роботи класу. У програмі також передбачено коментарі, що дають змогу автоматично генерувати документацію.

Список використаних джерел

1. Schildt H. Java: The Complete Reference, 12th Edition. / Herbert Schildt. – McGraw Hill, 2021. – 1280 p.
2. Java SE Documentation at a Glance [електронний ресурс]. – Режим доступу до документації: <http://www.oracle.com/technetwork/java/javase/> documentation/index.html
3. файл Plant.java

package KI\_304.Komisarov.LAB\_02;  
  
import java.io.FileWriter;  
import java.io.IOException;  
import java.time.LocalDateTime;  
  
*/\*\*  
 \* Клас Plant описує рослину та її складові.  
 \*/*public class Plant implements AutoCloseable {  
 private String name;  
 private Root root;  
 private Stem stem;  
 private Leaf leaf;  
  
 private FileWriter logWriter;  
  
 // Конструктор 1  
 public Plant(String name, Root root, Stem stem, Leaf leaf) throws IOException {  
 this.name = name;  
 this.root = root;  
 this.stem = stem;  
 this.leaf = leaf;  
 this.logWriter = new FileWriter("plant\_log.txt", true);  
 log("Створено рослину: " + name);  
 }  
  
 // Конструктор 2 (за замовчуванням)  
 public Plant(String name) throws IOException {  
 this(name, new Root(5, true), new Stem(10, 2), new Leaf(3, "зелений"));  
 }  
  
 // Методи  
 public String getName() { return name; }  
 public Root getRoot() { return root; }  
 public Stem getStem() { return stem; }  
 public Leaf getLeaf() { return leaf; }  
  
 public void growStem(double cm) throws IOException {  
 stem.grow(cm);  
 log("Стебло виросло на " + cm + " см");  
 }  
  
 public void growRoot(double cm) throws IOException {  
 root.grow(cm);  
 log("Корінь виріс на " + cm + " см");  
 }  
  
 public void growLeaves(int count) throws IOException {  
 leaf.grow(count);  
 log("З’явилося " + count + " нових листків");  
 }  
  
 public void changeLeafColor(String color) throws IOException {  
 leaf.changeColor(color);  
 log("Змінено колір листя на " + color);  
 }  
  
 public void printInfo() throws IOException {  
 String info = "Рослина: " + name + "\n" + root + "\n" + stem + "\n" + leaf;  
 System.*out*.println(info);  
 log("Виведено інформацію про рослину");  
 }  
  
 private void log(String action) throws IOException {  
 logWriter.write(LocalDateTime.*now*() + " - " + action + "\n");  
 logWriter.flush();  
 }  
  
 @Override  
 public void close() throws IOException {  
 if (logWriter != null) {  
 log("Закриття лог-файлу");  
 logWriter.close();  
 }  
 }  
}

1. ФАЙЛ leaf.JAVA

package KI\_304.Komisarov.LAB\_02;  
  
*/\*\*  
 \* Клас Leaf описує листя рослини.  
 \*/*public class Leaf {  
 private int count; // кількість листків  
 private String color;  
  
 public Leaf(int count, String color) {  
 this.count = count;  
 this.color = color;  
 }  
  
 public int getCount() { return count; }  
 public String getColor() { return color; }  
  
 public void grow(int number) {  
 count += number;  
 }  
  
 public void changeColor(String newColor) {  
 color = newColor;  
 }  
  
 @Override  
 public String toString() {  
 return "Листя: " + count + " шт., колір " + color;  
 }  
}

1. ФАЙЛ Root.JAVA

package KI\_304.Komisarov.LAB\_02;  
  
*/\*\*  
 \* Клас Root описує корінь рослини.  
 \*/*public class Root {  
 private double length; // довжина кореня в см  
 private boolean healthy;  
  
 public Root(double length, boolean healthy) {  
 this.length = length;  
 this.healthy = healthy;  
 }  
  
 public double getLength() { return length; }  
 public boolean isHealthy() { return healthy; }  
  
 public void grow(double cm) {  
 length += cm;  
 }  
  
 @Override  
 public String toString() {  
 return "Корінь: довжина " + length + " см, стан " + (healthy ? "здоровий" : "хворий");  
 }  
}

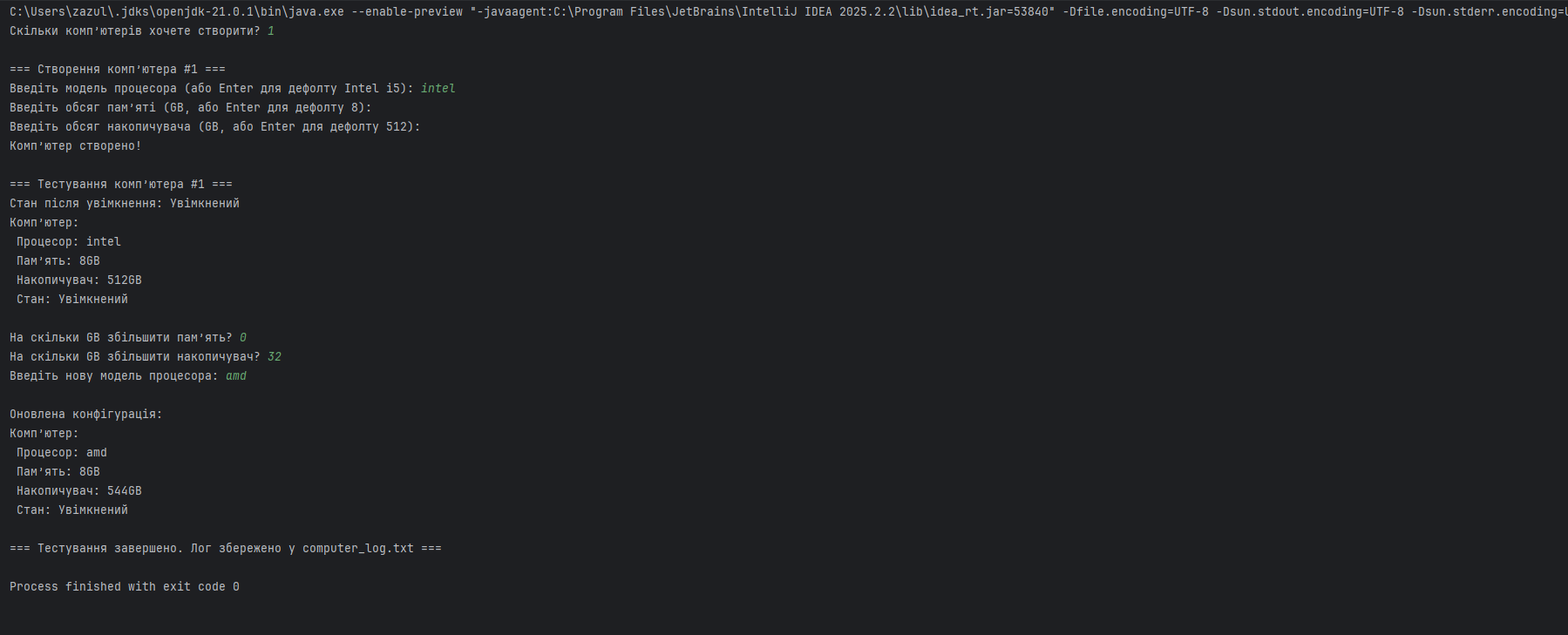
1. ФАЙЛ stem.JAVA

package KI\_304.Komisarov.LAB\_02;  
  
*/\*\*  
 \* Клас Stem описує стебло рослини.  
 \*/*public class Stem {  
 private double height; // висота в см  
 private double thickness; // товщина в мм  
  
 public Stem(double height, double thickness) {  
 this.height = height;  
 this.thickness = thickness;  
 }  
  
 public double getHeight() { return height; }  
 public double getThickness() { return thickness; }  
  
 public void grow(double cm) {  
 height += cm;  
 }  
  
 public void thicken(double mm) {  
 thickness += mm;  
 }  
  
 @Override  
 public String toString() {  
 return "Стебло: висота " + height + " см, товщина " + thickness + " мм";  
 }  
}

1. ФАЙЛ planttest.JAVA

package KI\_304.Komisarov.LAB\_02;  
  
  
*/\*\*  
 \* Клас для тестування Plant.  
 \*/*public class PlantTest {  
 public static void main(String[] args) {  
 try (Plant plant = new Plant("Троянда")) {  
 plant.printInfo();  
 plant.growStem(15);  
 plant.growRoot(7);  
 plant.growLeaves(5);  
 plant.changeLeafColor("жовтий");  
 plant.printInfo();  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 }  
}

1. вихідний матеріал



1. фрагмент згенерованої документації

