Національний університет «Львівська Політехніка»

Інститут комп’ютерних технологій, автоматики та метрології

Кафедра електронних обчислювальних машин



Звіт

Про виконання лабораторної роботи №2

З дисципліни «Кросплатформлені засоби програмування»

**Виконав:**

студент групи КІ-305

Ключко Д.С.

**Перевірив:**

Доцент кафедри ЕОМ

Іванов Ю. С.

Львів – 2023

**Тема:** Класи та пакети.

**Мета:** Ознайомитися з процесом розробки класів та пакетів мовою Java.

**Завдання:**

1. Написати та налагодити програму на мові Java, що реалізує у вигляді класу предметну  
область згідно варіанту. Програма має задовольняти наступним вимогам:  
 • програма має розміщуватися в пакеті Група.Прізвище.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.

**Завдання згідно варіанту:**

**Варіант: 12**



**Код програми:**

**Water.java:**

**package** lab2Klyuchko305;

**import** java.io.\*;

**public** **class** Water {

**private** **static** Square *Squareofwater*;

**private** EcologyWater Ecologysytuation;

**private** **double** temperature;

**private** Fullinformation Information;

**private** PrintWriter fout;

/\*\*

\* The Water class represents properties and functionalities related to a body of water.

\*/

**public** Water() **throws** FileNotFoundException{

// Class attributes including water square, ecological situation, temperature, information, and file output

*Squareofwater* = **new** Square();

Ecologysytuation = **new** EcologyWater();

temperature =0;

Information = **new** Fullinformation();

fout = **new** PrintWriter ("Log.txt");

}

/\*\*

\* Constructor with a temperature parameter initializing water properties with given temperature.

\* **@param** xtemperature the temperature of the water

\* **@throws** FileNotFoundException if Log.txt file is not found

\*/

**public** Water(**double** xtemperature) **throws** FileNotFoundException{

*Squareofwater* = **new** Square();

Ecologysytuation = **new** EcologyWater();

temperature = xtemperature;

Information = **new** Fullinformation();

fout = **new** PrintWriter ("Log.txt");

}

/\*\*

\* Retrieves the calculated square of the water body.

\* **@return** the calculated square of the water body

\*/

**public** **double** GetCalcSquare()

{

**return** *Squareofwater*.calcSquare();

}

/\*\*

\* Determines and displays the current depth of the water body based on temperature conditions.

\*/

**public** **void** CurrentDepth()

{

**double** curentdepth;

**if**(temperature >= 25 && temperature <=45)

{

curentdepth = *Squareofwater*.getDepth()-0.75;

}

**else** {

curentdepth = *Squareofwater*.getDepth();

}

System.***out***.print("Поточна глибина: ");

System.***out***.print(curentdepth);

System.***out***.println(" m ");

fout.print("Поточна глибина: ");

fout.print(curentdepth);

fout.println(" m");

}

/\*\*

\* Sets the width and length parameters for the water body's square.

\* **@param** widthSquareofwater the width of the water body

\* **@param** lengthSquareofwater the length of the water body

\*/

**public** **void** setParametersforSquare(**double** widthSquareofwater, **double** lengthSquareofwater){

*Squareofwater*.setLengts(lengthSquareofwater);

*Squareofwater*.setWidth(widthSquareofwater);

}

/\*\*

\* Retrieves the width of the water body.

\* **@return** the width of the water body

\*/

**public** **double** getWidthforWater()

{

**return** *Squareofwater*.getWidth();

}

/\*\*

\* Retrieves the length of the water body.

\* **@return** the length of the water body

\*/

**public** **double** getLengthforWater()

{

**return** *Squareofwater*.getLength();

}

/\*\*

\* Sets parameters for the ecological situation such as color and transparency of the water.

\* **@param** colorwaterEcologysytuation the color of the water

\* **@param** transparencyEcologysytuation the transparency level of the water

\*/

**public** **void** setParametersforEcologysituation(String colorwaterEcologysytuation, **int** transparencyEcologysytuation)

{

Ecologysytuation.setColorwater(colorwaterEcologysytuation);

Ecologysytuation.setTransparency(transparencyEcologysytuation);

}

/\*\*

\* Displays the current ecological situation of the water body.

\*/

**public** **void** CurentEcologysituation()

{

Ecologysytuation.Ecologysituation();

}

/\*\*

\* Displays the maximum depth of the water body.

\*/

**public** **void** Depth()

{

fout.print("Максимальна глибина: ");

fout.print(*Squareofwater*.getDepth());

fout.println(" m");

System.***out***.print("Максимальна глибина: ");

System.***out***.print (*Squareofwater*.getDepth());

System.***out***.println(" m ");

}

/\*\*

\* Sets parameters for the geographical information of the water body.

\* **@param** countryInformation the country of the water body

\* **@param** regionInformation the region of the water body

\* **@param** typeInformation the type of the water body

\*/

**public** **void** setParametersforInformation(String countryInformation,String regionInformation, String typeInformation)

{

Information.setCountry(countryInformation);

Information.setRegions(regionInformation);

Information.setType(typeInformation);

}

/\*\*

\* Displays the geographical information of the water body.

\*/

**public** **void** getInformation()

{

System.***out***.print("Країна: ");

System.***out***.println(Information.getCountry());

System.***out***.print("Регіон: ");

System.***out***.println(Information.getRegion());

System.***out***.print("Тип водойми: ");

System.***out***.println(Information.getType());

fout.print ("Країна: ");

fout.println(Information.getRegion());

fout.print("Регіон: ");

fout.println(Information.getCountry());

fout.print("Тип водойми: ");

fout.println(Information.getType());

fout.flush();

}

//class for calcSquare

/\*\*

\* The Square class represents the dimensions and calculations related to the square of a water body.

\*/

**class** Square{

**private** **double** width;

**private** **double** length;

/\*\*

\* Default constructor initializing width and length to zero.

\*/

**public** Square(){

width = 0.0;

length =0.0;

}

/\*\*

\* Constructor initializing width and length with provided values.

\* **@param** widthSquareofwater the width of the water body

\* **@param** lengthSquareofwater the length of the water body

\*/

**public** Square(**double** widthSquareofwater, **double** lengthSquareofwater) {

width = widthSquareofwater;

length = lengthSquareofwater;

}

// Methods get

/\*\*

\* Gets the width of the water body.

\* **@return** the width of the water body

\*/

**public** **double** getWidth()

{

**return** width;

}

/\*\*

\* Gets the length of the water body.

\* **@return** the length of the water body

\*/

**public** **double** getLength()

{

**return** length;

}

//Methods set

/\*\*

\* Sets the width and length of another Square object.

\* **@param** obj the Square object to set width and length

\*/

**public** **void** getSquare(Square obj) {

obj.width = width;

obj.length = length;

}

/\*\*

\* Sets the width of the water body.

\* **@param** widthSquareofwater the width to set

\*/

**public** **void** setWidth(**double** widthSquareofwater)

{

width = widthSquareofwater;

}

/\*\*

\* Sets the length of the water body.

\* **@param** lengthSquareofwater the length to set

\*/

**public** **void** setLengts(**double** lengthSquareofwater)

{

length = lengthSquareofwater;

}

/\*\*

\* Calculates the depth of the water body based on width and length.

\* **@return** the depth of the water body

\*/

**public** **double** getDepth() {

**double** depth;

depth = 4\*(width)/(length);

**return** depth;

}

/\*\*

\* Calculates the square of the water body and displays it.

\* **@return** the calculated square of the water body

\*/

**public** **double** calcSquare() {

**double** square;

square = *Squareofwater*.getLength() \* *Squareofwater*.getWidth();

fout.print("Площа водойми: ");

fout.print(square);

fout.println(" m^2");

System.***out***.print("Площа водойми: ");

System.***out***.print(square);

System.***out***.println(" m^2");

**return** square;

}

}

/\*\*

\* The EcologyWater class manages ecological parameters and checks for a water body.

\*/

**class** EcologyWater{

**private** String colorwater;

**private** **int** transparency;

/\*\*

\* Default constructor initializing default color and transparency values.

\*/

**public** EcologyWater(){

colorwater = "Бірюзовий";

transparency = 70;

}

/\*\*

\* Constructor initializing color and transparency with provided values.

\* **@param** colorwaterEcologysytuation the color of the water

\* **@param** transparencyEcologysytuation the transparency level of the water

\*/

**public** EcologyWater (String colorwaterEcologysytuation, **int** transparencyEcologysytuation) {

colorwater = colorwaterEcologysytuation;

transparency = transparencyEcologysytuation;

}

/\*\*

\* Retrieves the color of the water.

\* **@return** the color of the water

\*/

**public** String getColorwater()

{

**return** colorwater;

}

/\*\*

\* Retrieves the transparency level of the water.

\* **@return** the transparency level of the water

\*/

**public** **int** getTransparency()

{

**return** transparency;

}

/\*\*

\* Sets the color and transparency of another EcologyWater object.

\* **@param** obj the EcologyWater object to set color and transparency

\*/

**public** **void** getEcologyParameters(EcologyWater obj) {

obj.colorwater = colorwater;

obj.transparency = transparency;

}

/\*\*

\* Sets the color of the water.

\* **@param** colorwaterEcologysytuation the color to set

\*/

**public** **void** setColorwater(String colorwaterEcologysytuation)

{

colorwater = colorwaterEcologysytuation;

}

/\*\*

\* Sets the transparency level of the water.

\* **@param** transparencyEcologysytuation the transparency level to set

\*/

**public** **void** setTransparency(**int** transparencyEcologysytuation)

{

transparency = transparencyEcologysytuation;

}

/\*\*

\* Checks the ecological situation of the water body based on color and transparency.

\* Prints information about water cleanliness based on defined criteria.

\*/

**public** **void** Ecologysituation()

{

**if**(colorwater=="Зелений"||colorwater=="Жовтий"||colorwater=="Коричневий")

{

System.***out***.println("Колір води є незадовільним! В даній водоймі брудна вода!!!");

fout.println("Колір води є незадовільним! В даній водоймі брудна вода!!!");

}

**else**

{

System.***out***.println("За результатами перевірки кольору води, водойма є чистою!");

fout.println("За результатами перевірки кольору води, водойма є чистою!");

}

**if**(transparency<60)

{

System.***out***.println("Вода не є прозорою! В даній водоймі брудна вода!!!");

fout.println("Вода не є прозорою! В даній водоймі брудна вода!!!");

}

**else**

{

System.***out***.println("Вода прозора,отже водойма є чистою!");

fout.println("Вода прозора,отже водойма є чистою!");

}

}

}

/\*\*

\* The Fullinformation class manages geographical information about a water body.

\*/

**class** Fullinformation{

**private** String country;

**private** String region;

**private** String type;

/\*\*

\* Default constructor initializing default country, region, and type values.

\*/

**public** Fullinformation(){

country = "default";

region = "default";

type = "default";

}

/\*\*

\* Constructor initializing country, region, and type with provided values.

\* **@param** countryInformation the country of the water body

\* **@param** regionInformation the region of the water body

\* **@param** typeInformation the type of the water body

\*/

**public** Fullinformation (String countryInformation,String regionInformation, String typeInformation) {

country = countryInformation;

region =regionInformation;

type = typeInformation;

}

/\*\*

\* Retrieves the country of the water body.

\* **@return** the country of the water body

\*/

**public** String getCountry()

{

**return** country;

}

/\*\*

\* Retrieves the region of the water body.

\* **@return** the region of the water body

\*/

**public** String getRegion()

{

**return** region;

}

/\*\*

\* Retrieves the type of the water body.

\* **@return** the type of the water body

\*/

**public** String getType()

{

**return** type;

}

/\*\*

\* Sets the country of the water body.

\* **@param** countryInformation the country to set

\*/

**public** **void** setCountry(String countryInformation)

{

country = countryInformation;

}

/\*\*

\* Sets the region of the water body.

\* **@param** regionInformation the region to set

\*/

**public** **void** setRegions(String regionInformation)

{

region =regionInformation;

}

/\*\*

\* Sets the type of the water body.

\* **@param** typeInformation the type to set

\*/

**public** **void** setType(String typeInformation)

{

type =typeInformation;

}

/\*\*

\* Sets the country, region, and type of another Fullinformation object.

\* **@param** obj the Fullinformation object to set country, region, and type

\*/

**public** **void** getEcologyParameters(Fullinformation obj) {

obj.country = country;

obj.region = region;

obj.type = type;

}

}

}

**WaterApp.java:**

package lab2Klyuchko305;

/\*\*

\* Class App realize program to lab work 2

\*

\* @author Klyuchko Dmytro

\* @version 1.1

\* @since version 1.0

\*/

import java.io.\*;

public class WaterApp {

/\*\*

\* @param args

\* @throws FileNotFoundException

\*/

public static void main(String[] args) throws FileNotFoundException {

Water lake = new Water(27);

lake.setParametersforSquare(24, 32);

lake.GetCalcSquare();

lake.setParametersforEcologysituation("Голубий", 70);

lake.CurentEcologysituation();

lake.Depth();

lake.CurrentDepth();

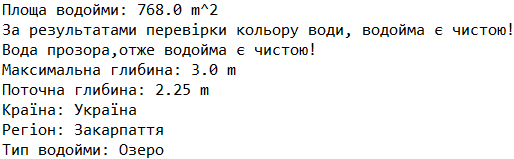
lake.setParametersforInformation("Україна", "Закарпаття", "Озеро");

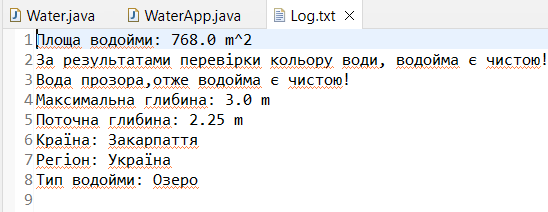
lake.getInformation();

}

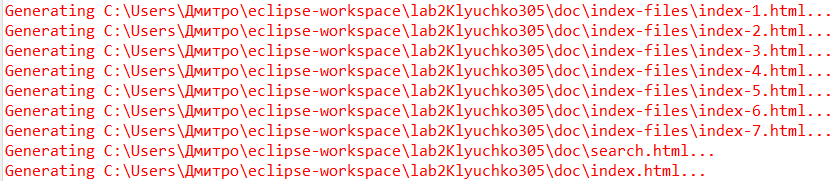
}

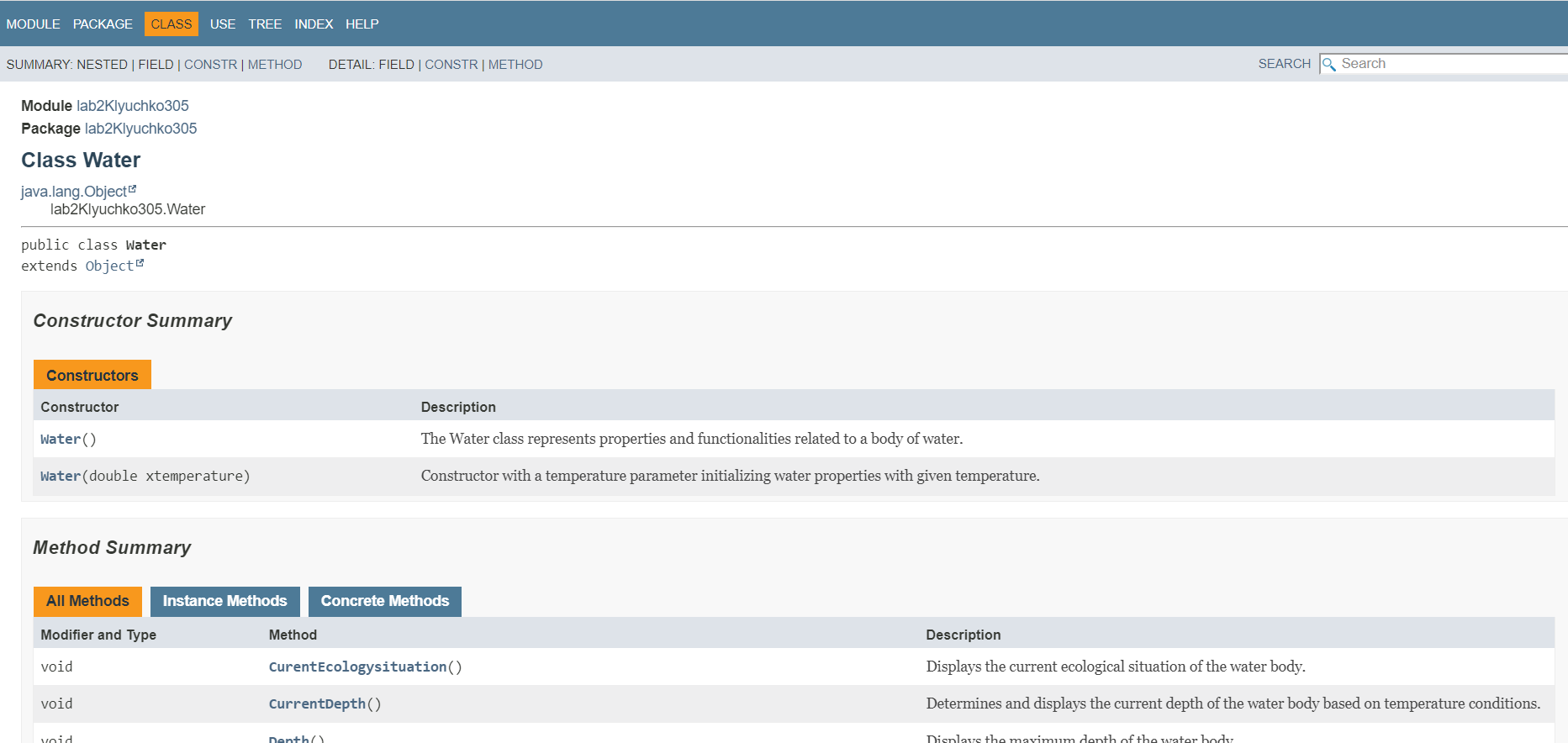
**Результат роботи програми у консолі та файлі:**

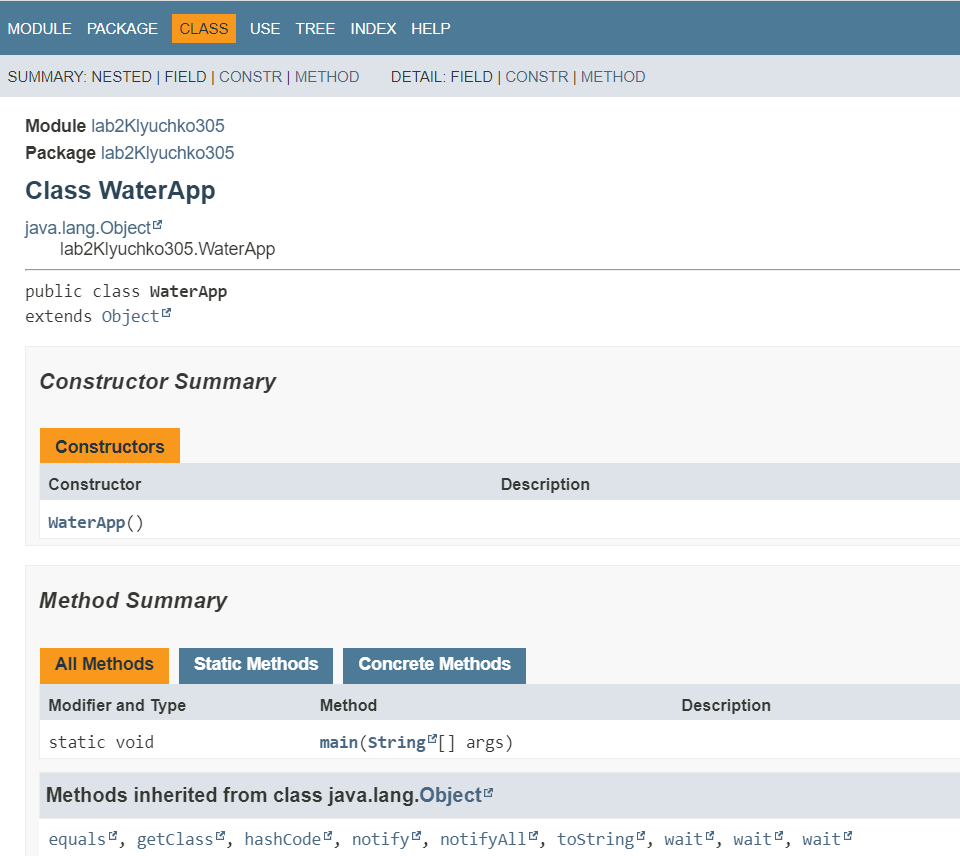
****

****

**Фрагмент генерації документації:**

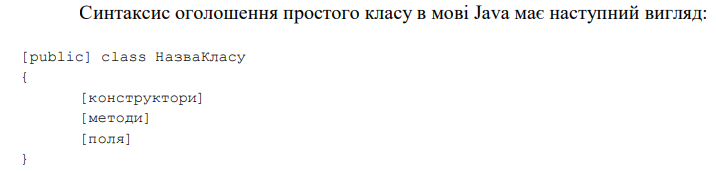
****

****

****

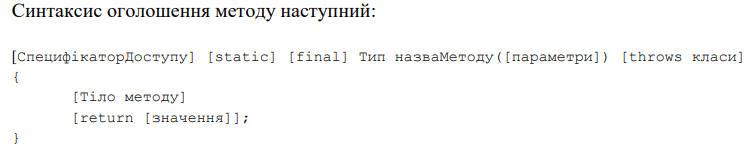
**КОНТРОЛЬНІ ПИТАННЯ**

1. Синтаксис визначення класу.

**Відповідь: **

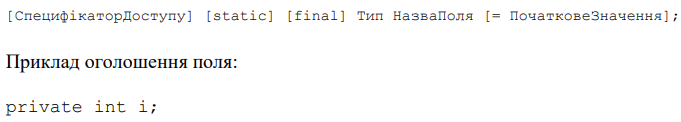
1. Синтаксис визначення методу.

**Відповідь:**

****

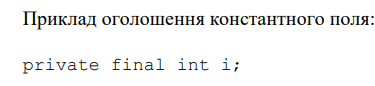
1. Синтаксис оголошення поля.

**Відповідь:**



1. Як оголосити та ініціалізувати константне поле?

**Відповідь:**

****

Ініціалізація при оголошенні:

При оголошенні поля, ви можете встановити його значення. Наприклад:

java

Copy code

public class MyClass {

// Константне поле ініціалізується при оголошенні

private final int MY\_CONSTANT = 10;

}

Ініціалізація в конструкторі:

Якщо значення константи вам потрібно встановити під час створення об'єкта класу, ви можете ініціалізувати його у конструкторі. Однак, після першої ініціалізації константне поле не можна змінювати.

java

Copy code

public class MyClass {

private final int MY\_CONSTANT;

public MyClass(int value) {

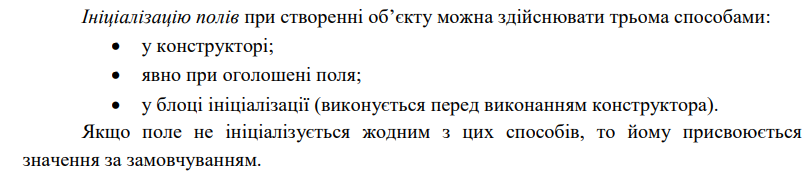
MY\_CONSTANT = value;

}

}

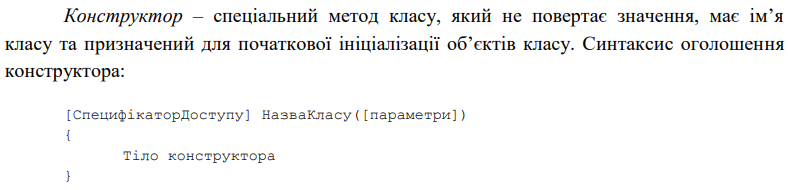
1. Які є способи ініціалізації полів?

**Відповідь:**



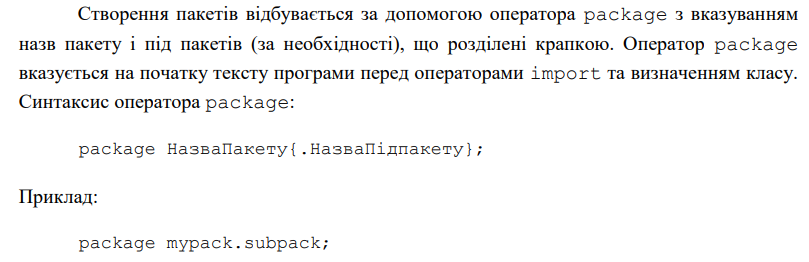
1. Синтаксис визначення конструктора.

**Відповідь:**



1. Синтаксис оголошення пакету.

**Відповідь:**



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

**Відповідь:**

Щоб використовувати класи, що належать зовнішнім пакетам у вашій програмі, вам потрібно вказати їх імпорт.

1. В чому суть статичного імпорту пакетів?

**Відповідь:**

Статичний імпорт в Java - це механізм, який дозволяє коротше викликати статичні методи і статичні поля з класу, не вказуючи повністю кваліфіковане ім'я класу.

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

**Відповідь:**

При використанні пакетів у Java ставляються певні вимоги до організації файлів і каталогів:

1. Ієрархія пакетів: Файли повинні бути розміщені у відповідних пакетах згідно з ієрархією пакетів. Це означає, що розташування файлів відображає структуру пакетів.
2. Директорія пакету: Файли Java повинні розташовуватися у відповідних папках пакетів. Наприклад, якщо ви маєте пакет com.example, ваш файл Java повинен розташовуватися у відповідній структурі каталогів, такій як com/example/YourClass.java.
3. Ім'я файлу: Ім'я файлу Java повинно відповідати імені класу, яке він містить. Наприклад, якщо ваш клас називається YourClass, то файл повинен мати ім'я YourClass.java.
4. Імпорти і пакети: У файлі Java можуть бути імпорти і заявки package. Це використовується для підключення класів з інших пакетів або для визначення поточного пакету, від якого відповідні класи його визначають.
5. Відносні шляхи: При використанні імпортів вам необхідно вказувати повний шлях до класу відносно його розташування в пакеті. Наприклад: import com.example.packageName.ClassName;.

**Висновок:** на цій лабораторній роботі я ознайомитися з процесом розробки класів та пакетів мовою Java.