Національний університет «Львівська Політехніка»

Інститут комп’ютерних технологій, автоматики та метрології

Кафедра електронних обчислювальних машин



Звіт

Про виконання лабораторної роботи №3

З дисципліни «Кросплатформлені засоби програмування»

**Виконав:**

студент групи КІ-306

Ориняк Ю.І.

**Перевірив:**

Доцент кафедри ЕОМ

Іванов Ю. С.

Львів – 2023

**Тема:** Спадкування та інтерфейси.

**Мета:** Ознайомитися з спадкуванням та інтерфейсами у мові Java.

**Завдання:**

1. Написати та налагодити програму на мові Java, що розширює клас, що реалізований у лабораторній роботі №2, для реалізації предметної області заданої варіантом. Суперклас, що реалізований у лабораторній роботі №2, зробити абстрактним. Розроблений підклас має забезпечувати механізми свого коректного функціонування та реалізовувати мінімум один інтерфейс. Програма має розміщуватися в пакеті Група.Прізвище.Lab3 та володіти коментарями, які дозволять автоматично згенерувати документацію до розробленого пакету.  
2. Автоматично згенерувати документацію до розробленого пакету.  
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**

****

**Виконання:**

**Код програми:**

**Water.java:**

**package** lab3OrinyakKI306;

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

**public** **class** WaterBody {

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

**private** FishingofWater Fishing;

**private** NaturalWorld Natural;

**public** PrintWriter fout;

**private** **double** depth;

/\*\*

\* Default constructor. Initializes objects and field values of the WaterBody class.

\* **@throws** FileNotFoundException thrown if unable to find or create a file for writing.

\*/

**public** WaterBody() **throws** FileNotFoundException{

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

Fishing = **new** FishingofWater();

Natural = **new** NaturalWorld();

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

depth = 0.0;

}

/\*\*

\* Constructor that takes the depth parameter of the water body.

\* Initializes objects and field values of the WaterBody class.

\* **@param** xDepth the depth of the water body.

\* **@throws** FileNotFoundException thrown if unable to find or create a file for writing.

\*/

**public** WaterBody(**double** xDepth) **throws** FileNotFoundException{

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

Fishing = **new** FishingofWater();

Natural = **new** NaturalWorld();

depth = xDepth;

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

}

/\*\*

\* Method to check the water situation based on its depth.

\*

\*/

**public** **void** SetDepthWater (**double** xDepth)

{

depth = xDepth;

}

**public** **void** SituationofWater()

{

**if** (depth < 2.0)

{

System.***out***.println("Водойма на стадії обміління!");

fout.println("\n"+"Водойма на стадії обміління!");

}

**else**

{

System.***out***.println("У водойми хороший водний режим!!! ");

fout.println("\n"+"У водойми хороший водний режим!!! ");

}

}

/\*\*

\* Method to calculate the square area of the water body.

\* **@return** the square area of the water body.

\*/

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

{

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

}

/\*\*

\* Method to set parameters for the water body's square dimensions.

\* **@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);

}

/\*\*

\* Method to get the width of the water body.

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

\*/

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

{

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

}

/\*\*

\* Method to get the length of the water body.

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

\*/

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

{

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

}

/\*\*

\* Sets the amount of fish for fishing.

\* **@param** xAmountoffish the amount of fish to set.

\*/

**public** **void** setParameterforFishing(**int** xAmountoffish)

{

Fishing.setAmountoffish(xAmountoffish);

}

/\*\*

\* Displays the remaining amount of fish after fishing.

\*/

**public** **void** ResultFishing()

{

System.***out***.print("Кількість риби яка лишилася після риболовлі: ");

System.***out***.println(Fishing.Fishing());

fout.print("Кількість риби яка лишилася після риболовлі: ");

fout.println(Fishing.Fishing());

}

/\*\*

\* Runs the process of stocking the pond with fish.

\*/

**public** **void** RunStockapond()

{

System.***out***.print("Зариблення водойми пройшло успішно, кількість риби: ");

System.***out***.println(Fishing.Stockapond());

fout.print("Зариблення водойми пройшло успішно, кількість риби: ");

fout.print(Fishing.Stockapond());

}

/\*\*

\* Sets parameters for the natural world (flora and fauna).

\* **@param** xFlora the flora of the water body.

\* **@param** xFauna the fauna of the water body.

\*/

**public** **void** setParametersforNaturalWord(String xFlora, String xFauna)

{

Natural.setFauna(xFauna);

Natural.setFlora(xFlora);

}

/\*\*

\* Retrieves and displays information about the flora and fauna of the water body.

\*/

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

{

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

System.***out***.println(Natural.getFlora());

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

System.***out***.println(Natural.getFauna());

fout.print ("Рослини водойми: ");

fout.println(Natural.getFlora());

fout.print("Тварини водойми: ");

fout.println(Natural.getFauna());

}

/\*\*

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

\*/

**class** Square{

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

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

/\*\*

\* Default constructor for Square, initializes width and length to 0.0.

\*/

**public** Square(){

width = 0.0;

length =0.0;

}

/\*\*

\* Parameterized constructor for Square with specified width and length 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;

}

/\*\*

\* Returns the width of the water body.

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

\*/

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

{

**return** width;

}

/\*\*

\* Returns the length of the water body.

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

\*/

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

{

**return** length;

}

/\*\*

\* Copies the width and length of the current Square object to another Square object.

\* **@param** obj the Square object to which the dimensions are copied.

\*/

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

obj.width = width;

obj.length = length;

}

/\*\*

\* Sets the width of the water body.

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

\*/

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

{

width = widthSquareofwater;

}

/\*\*

\* Sets the length of the water body.

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

\*/

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

{

length = lengthSquareofwater;

}

/\*\*

\* Calculates and returns the square area of the water body.

\* **@return** the square area 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 FishingofWater class manages the fishing activities and fish population in the water body.

\*/

**class** FishingofWater{

**private** **int** amountoffish;

/\*\*

\* Default constructor for FishingofWater, initializes the fish count to 0.

\*/

**public** FishingofWater(){

amountoffish =0;

}

/\*\*

\* Constructor for FishingofWater with a specified initial fish count.

\* **@param** xAmountoffish the initial fish count.

\*/

**public** FishingofWater (**int** xAmountoffish) {

amountoffish = xAmountoffish;

}

/\*\*

\* Gets the current amount of fish.

\* **@return** the current amount of fish.

\*/

**public** **int** getAmountofFish()

{

**return** amountoffish ;

}

/\*\*

\* Copies the fish count to another FishingofWater object.

\* **@param** obj the FishingofWater object to which the fish count is copied.

\*/

**public** **void** getFishingParameterObj(FishingofWater obj) {

obj.amountoffish = amountoffish;

}

/\*\*

\* Sets the amount of fish to a specified value.

\* **@param** xAmountoffish the amount of fish to set.

\*/

**public** **void** setAmountoffish(**int** xAmountoffish )

{

amountoffish = xAmountoffish;

}

/\*\*

\* Simulates the remaining amount of fish after fishing, reducing the count by 5.

\* **@return** the remaining amount of fish after fishing.

\*/

**public** **int** Fishing()

{

**return** amountoffish - 5;

}

/\*\*

\* Simulates the fish population after stocking the pond with a fixed count of 10000 fish.

\* **@return** the fish count after stocking the pond.

\*/

**public** **int** Stockapond()

{

**return** amountoffish = 10000;

}

}

/\*\*

\* The NaturalWorld class manages information about the flora and fauna in the water body.

\*/

**class** NaturalWorld{

**private** String flora;

**private** String fauna;

/\*\*

\* Default constructor for NaturalWorld, initializes flora and fauna to default values.

\*/

**public** NaturalWorld(){

flora = "default";

fauna = "default";

}

/\*\*

\* Constructor for NaturalWorld with specified flora and fauna.

\* **@param** xFlora the flora of the water body.

\* **@param** xFauna the fauna of the water body.

\*/

**public** NaturalWorld (String xFlora,String xFauna) {

flora = xFlora;

fauna =xFauna;

}

/\*\*

\* Gets the flora of the water body.

\* **@return** the flora of the water body.

\*/

**public** String getFlora()

{

**return** flora;

}

/\*\*

\* Gets the fauna of the water body.

\* **@return** the fauna of the water body.

\*/

**public** String getFauna()

{

**return** fauna;

}

/\*\*

\* Sets the flora of the water body.

\* **@param** xFlora the flora of the water body.

\*/

**public** **void** setFlora(String xFlora)

{

flora = xFlora;

}

/\*\*

\* Sets the fauna of the water body.

\* **@param** xFauna the fauna of the water body.

\*/

**public** **void** setFauna(String xFauna)

{

fauna =xFauna;

}

/\*\*

\* Copies the flora and fauna of the current NaturalWorld object to another object.

\* **@param** obj the NaturalWorld object to which flora and fauna are copied.

\*/

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

obj.fauna = fauna;

obj.flora = flora;

}

}

}

**Sea.java:**

**package** lab3OrinyakKI306;

**import** java.io.FileNotFoundException;

/\*\*

\* The Volume interface represents a contract for classes that can calculate volume.

\* Classes implementing this interface must provide an implementation for the calculateVolume method.

\*/

**interface** Volume {

**double** calculateVolume();

}

/\*\*

\* The Sea class represents a body of water in the program, extending the WaterBody class

\* and implementing the Volume interface. It encapsulates information about the salinity

\* and depth of the sea.

\*/

**public** **class** Sea **extends** WaterBody **implements** Volume {

**private** **double** salinity;

**private** **int** depth;

/\*\*

\* Constructs a Sea object with specified depth and salinity.

\*

\* **@param** Xdepth The depth of the sea.

\* **@param** Xsalinity The salinity level of the sea.

\*/

**public** Sea(**int** Xdepth, **double** Xsalinity) **throws** FileNotFoundException {

depth = Xdepth;

salinity =Xsalinity;

}

/\*\*

\* Gets the salinity of the sea.

\*

\* **@return** The salinity value.

\*/

**public** **double** getSalinity() {

**return** salinity;

}

/\*\*

\* Sets the salinity level of the sea.

\*

\* **@param** Xsalinity The salinity value to set.

\*/

**public** **void** setSalinity(**double** Xsalinity) {

salinity = Xsalinity;

}

/\*\*

\* Gets the depth of the sea.

\*

\* **@return** The depth value.

\*/

**public** **int** getDepth() {

**return** depth;

}

/\*\*

\* Sets the depth of the sea.

\*

\* **@param** Xdepth The depth value to set.

\*/

**public** **void** setDepth(**int** Xdepth) {

depth = Xdepth;

}

/\*\*

\* Checks the salinity level of the sea and prints a message accordingly.

\*/

**public** **void** checkSalinity() {

**if** (salinity < 1.0) {

System.***out***.println("У морі недостатня солоність води!!!");

fout.println("У морі недостатня солоність води!!!");

} **else** {

System.***out***.println("Солоність води в межах норми!.");

fout.println("Солоність води в межах норми!.");

}

}

/\*\*

\* Calculates the volume of the sea based on width, length, and depth.

\*

\* **@return** The calculated volume.

\*/

**public** **double** calculateVolume() {

**double** volume = getWidthforWater()\*getLengthforWater() \* depth;

**return** volume;

}

/\*\*

\* Prints the provided line.

\*

\* **@param** line The line to be printed.

\*/

**public** **void** Print(**double** line) {

System.***out***.println(line);

}

/\*\*

\* Writes the provided line to a file.

\*

\* **@param** line The line to be written to the file.

\*/

**public** **void** WritetoFile(**double** line) {

**if** (line == 0) {

fout.close();

}

fout.println(line);

fout.flush();

}

/\*\*

\* Writes the provided string to a file.

\*

\* **@param** line The string to be written to the file.

\*/

**public** **void** WritetoFile(String line) {

**if** (line == **null**) {

fout.close();

}

fout.print(line);

fout.flush();

}

}

**Seaapp.java**

**package** lab3OrinyakKI306;

**import** java.io.FileNotFoundException;

/\*\*

\* The Seaapp class contains the main method to demonstrate various operations

\* and functionalities related to the Sea class.

\*/

**public** **class** Seaapp {

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

// Create a Sea object named blacksea with specified depth and salinity

Sea blacksea = **new** Sea(13, 4.75);

// Set parameters for the square of the water body

blacksea.setParametersforSquare(45.1, 64.5);

blacksea.GetCalcSquare(); // Calculate the square of the water body

blacksea.setParameterforFishing(1000); // Set parameters for fishing

blacksea.ResultFishing(); // Get the result of fishing

blacksea.RunStockapond(); // Run stocking a pond

blacksea.setParametersforNaturalWord("ламінарія, зелена водорость", "cудак, дельфін, камбала, бичок");

// Set parameters for natural world

System.***out***.println("Ширина водойми: " + blacksea.getWidthforWater()); // Print width of water body

System.***out***.println("Довжина водойми: " + blacksea.getLengthforWater()); // Print length of water body

blacksea.SetDepthWater(1000.75); // Set depth of water body

blacksea.SituationofWater(); // Determine the situation of water

blacksea.getInformation(); // Get information

blacksea.checkSalinity(); // Check salinity of water

System.***out***.print("Об'єм води становить: ");

blacksea.Print(blacksea.calculateVolume()); // Calculate and print the volume of water

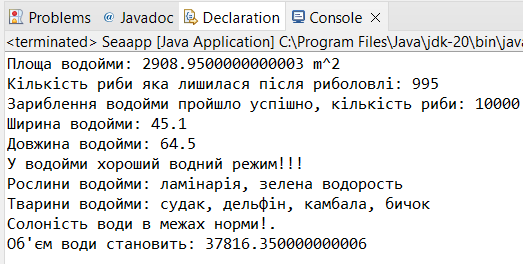
blacksea.WritetoFile("Об'єм води становить: "); // Write information to a file

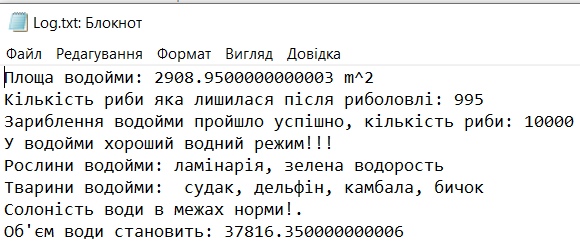
blacksea.WritetoFile(blacksea.calculateVolume()); // Write volume information to a file

}

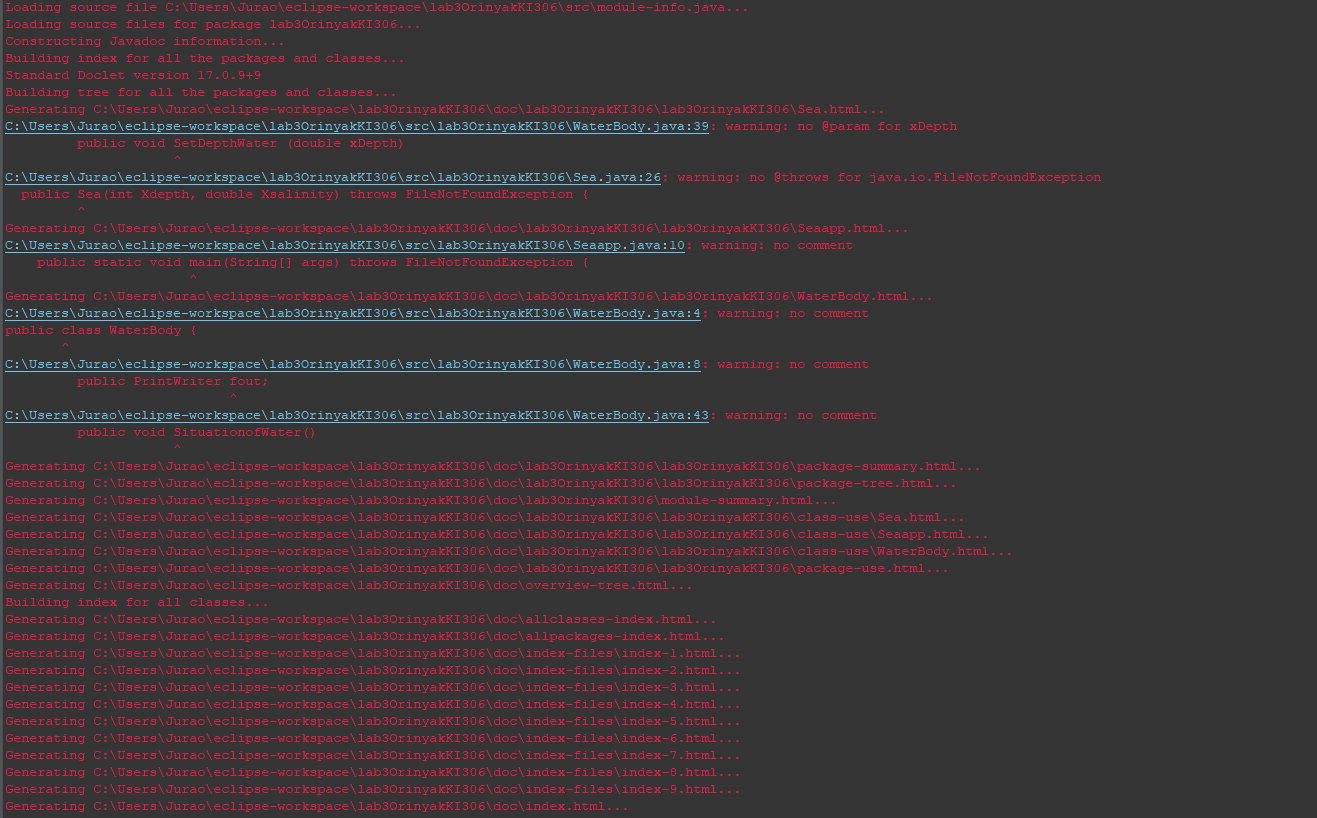
}

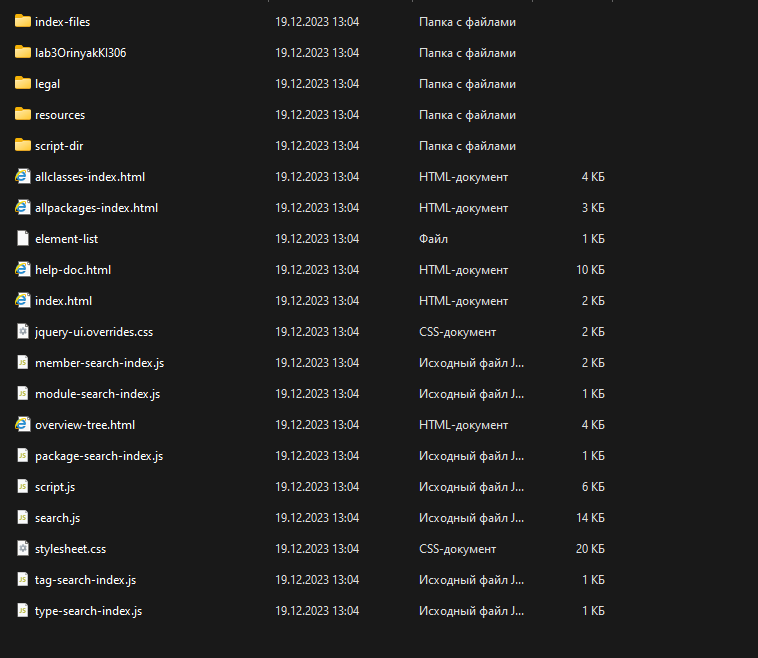
**Результат роботи програми у консолі та файлі:**





**Фрагмент генерації документації:**

****

****

**Висновок:** на цій лабораторній роботі я ознайомився з спадкуванням та інтерфейсами у мові Java. Перетворив попередній клас водойма у абстрактний і реалізував деякі його методи у новоствореному класі море. Також створив інтерфейс «Приріст кораблів» і реалізував його метод в класі море також.