# **Національний технічний університет України**

# **«Київський політехнічний інститут»**

# **Факультет інформатики та обчислювальної техніки**

# **Кафедра обчислювальної техніки**

# **Лабораторна робота №4**

### **З програмування**

**виконав студент першого курсу**

**група ІО-91**

**Нечитайло Олег Андрійович**

**Тема:** Створення власного класу в мові програмування Java.

**Мета:** Здобуття навичок у створенні та використанні власного класу в мові програмування Java. Розробка засобів для сортування масиву об’єктів класу, використовуючи для цього стандартні методи.

**Варіант 12**

**Завдання:**

Інформація про море.

Створити клас, який описує задані варіантом дані (п.2), та складається не менше ніж з 7 полів та методів доступу до цих полів. Також цей клас повинен містити не менше 3 зв’язаних конструкторів. В класі необхідно створити, використовуючи стандартні засоби мови програмування, не менше 4 методів, за допомогою яких можна було б сортувати масив об’єктів такого класу за одним із полів. Для перевірки необхідно створити клас, що складається з виконавчого методу. Всі початкові дані задаються у виконавчому методі. Всі класи повинні бути задокументовані з використанням javadoc.

Документація классів:

Class CSea

**CSea**  
Клас данних про море  
  
**Поля:**   
name - Назва моря   
ocean - Назва басейну океану   
countries - Країни, що мають кордони з морем   
square - Площа моря   
salt - Соленість води   
volume - Об'єм води в морі   
temperature -Середня температура води в морі

### CSea

**CSea**(java.lang.String Name)

**Parameters:**

Name - значення поля name

### CSea

**CSea**(java.lang.String Name,

java.lang.String Ocean,

double Square,

double Salt)

**Parameters:**

Name - значення поля name

Ocean - значення поля ocean

Square - значення поля square

Salt - значення поля salt

### CSea

**CSea**(java.lang.String Name,

java.lang.String Ocean,

java.lang.String Countries,

double Square,

double Salt,

double Volume,

double Temperature)

**Parameters:**

Name - значення поля name

Ocean - значення поля ocean

Countries - значення поля countries

Square - значення поля square

Salt - значення поля salt

Volume - значення поля volume

Temperature - значення поля temperature

### GetName

public java.lang.String **GetName**()

**GetName()**

**Returns:**

значення поля String name

### GetOcean

public java.lang.String **GetOcean**()

**GetOcean()**

**Returns:**

значення поля String ocean

### GetCountries

public java.lang.String **GetCountries**()

**GetCountries()**

**Returns:**

значення поля String countries

### GetSalt

public double **GetSalt**()

**GetSalt()**

**Returns:**

значення поля String salt

### GetSquare

public double **GetSquare**()

**GetSquare()**

**Returns:**

значення поля String square

### GetVolume

public double **GetVolume**()

**GetVolume()**

**Returns:**

значення поля String volume

### GetTemperature

public double **GetTemperature**()

**GetTemperature()**

**Returns:**

значення поля String temperature

### SetName

public void **SetName**(java.lang.String Name)

**SetName(String Name)**

**Parameters:**

Name - нове значення поля name

### SetOcean

public void **SetOcean**(java.lang.String Ocean)

**SetOcean(String Ocean)**

**Parameters:**

Ocean - нове значення поля ocean

### SetCountries

public void **SetCountries**(java.lang.String Countries)

**SetCountries(String Countries)**

**Parameters:**

Countries - нове значення поля countries

### SetSalt

public void **SetSalt**(double Salt)

**SetSalt(String Salt)**

**Parameters:**

Salt - нове значення поля salt

### SetSquare

public void **SetSquare**(double Square)

**SetSquare(String Square)**

**Parameters:**

Square - нове значення поля square

### SetVolume

public void **SetVolume**(double Volume)

**SetVolume(String Volume)**

**Parameters:**

Volume - нове значення поля volume

### SetTemperature

public void **SetTemperature**(double Temperature)

**SetTemperature(String Temperature)**

**Parameters:**

Temperature - нове значення поля temperature

### SortByName

public static void **SortByName**([CSea](file:///F:\Alik\Java\workspace\S2Lab4\doc\CSea.html)[] array)

**void SortByName(CSea array[])**

**Parameters:**

array - массив з елементами класу CSea для сортування   
**Опис дії**  
Сортування массиву за зростанням за ім'ям.  
використовується клас SortByNameComp як компаратор і метод java.util.Arrays.sort для сортування.

### SortByOcean

public static void **SortByOcean**([CSea](file:///F:\Alik\Java\workspace\S2Lab4\doc\CSea.html)[] array)

**void SortByOcean(CSea array[])**

**Parameters:**

array - массив з елементами класу CSea для сортування   
**Опис дії**  
Сортування массиву за зростанням за назвою океану.  
використовується клас SortByNameOcean як компаратор і метод java.util.Arrays.sort для сортування.

### SortBySquare

public static void **SortBySquare**([CSea](file:///F:\Alik\Java\workspace\S2Lab4\doc\CSea.html)[] array)

**void SortBySquare(CSea array[])**

**Parameters:**

array - массив з елементами класу CSea для сортування   
**Опис дії**  
Сортування массиву за зростанням за площею  
використовується клас SortSquareComp як компаратор і метод java.util.Arrays.sort для сортування.

### SortByTemperature

public static void **SortByTemperature**([CSea](file:///F:\Alik\Java\workspace\S2Lab4\doc\CSea.html)[] array)

**void SortByTemperature(CSea array[])**

**Parameters:**

array - массив з елементами класу CSea для сортування   
**Опис дії**  
Сортування массиву за зростанням за температурою.  
використовується клас SortByTemperatureComp як компаратор і метод java.util.Arrays.sort для сортування.

class SortByNameComp

extends java.lang.Object

implements java.util.Comparator<[CSea](file:///F:\Alik\Java\workspace\S2Lab4\doc\CSea.html)>

**Method Detail**

public int **compare**([CSea](file:///F:\Alik\Java\workspace\S2Lab4\doc\CSea.html) o1,

[CSea](file:///F:\Alik\Java\workspace\S2Lab4\doc\CSea.html) o2)

**Specified by:**

compare in interface java.util.Comparator<[CSea](file:///F:\Alik\Java\workspace\S2Lab4\doc\CSea.html)>

class SortByOceanComp

extends java.lang.Object

implements java.util.Comparator<[CSea](file:///F:\Alik\Java\workspace\S2Lab4\doc\CSea.html)>

**Method Detail**

public int **compare**([CSea](file:///F:\Alik\Java\workspace\S2Lab4\doc\CSea.html) o1,

[CSea](file:///F:\Alik\Java\workspace\S2Lab4\doc\CSea.html) o2)

**Specified by:**

compare in interface java.util.Comparator<[CSea](file:///F:\Alik\Java\workspace\S2Lab4\doc\CSea.html)>

class SortBySquareComp

extends java.lang.Object

implements java.util.Comparator<[CSea](file:///F:\Alik\Java\workspace\S2Lab4\doc\CSea.html)>

**Method Detail**

public int **compare**([CSea](file:///F:\Alik\Java\workspace\S2Lab4\doc\CSea.html) o1,

[CSea](file:///F:\Alik\Java\workspace\S2Lab4\doc\CSea.html) o2)

**Specified by:**

compare in interface java.util.Comparator<[CSea](file:///F:\Alik\Java\workspace\S2Lab4\doc\CSea.html)>

class SortByTemperatureComp

extends java.lang.Object

implements java.util.Comparator<[CSea](file:///F:\Alik\Java\workspace\S2Lab4\doc\CSea.html)>

**Method Detail**

public int **compare**([CSea](file:///F:\Alik\Java\workspace\S2Lab4\doc\CSea.html) o1,

[CSea](file:///F:\Alik\Java\workspace\S2Lab4\doc\CSea.html) o2)

**Specified by:**

compare in interface java.util.Comparator<[CSea](file:///F:\Alik\Java\workspace\S2Lab4\doc\CSea.html)>

//CMain.java

**public** **class** CMain {

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

CSea arr[] = {**new** CSea("a", "b", "c", 5.3, 12d, 4d, 2d),

**new** CSea("f", "a", "d", 2.3, 6d, 0.01, 2.3),

**new** CSea("c", "d", "f", 3.4, 6.5, 0.5, 8.3)};

CSea.*SortByName*(arr);

**for**(**int** i=0; i<arr.length; i++){

System.*out*.print(arr[i]);

}

System.*out*.print("\n");

CSea.*CSortBySquare*(arr);

**for**(**int** i=0; i<arr.length; i++){

System.*out*.print(arr[i]);

}

System.*out*.print("\n");

CSea.*SortByOcean*(arr);

**for**(**int** i=0; i<arr.length; i++){

System.*out*.print(arr[i]);

}

System.*out*.print("\n");

CSea.*SortByTemperature*(arr);

**for**(**int** i=0; i<arr.length; i++){

System.*out*.print(arr[i]);

}

System.*out*.print("\n");

}

}

//CSea.java

**public** **class** CSea{

**private** String name;

**private** String ocean;

**private** String countries;

**private** **double** square, salt, volume, temperature;

**public** String GetName() {**return** name;}

**public** String GetOcean() {**return** ocean;}

**public** String GetCountries() {**return** countries;}

**public** **double** GetSalt() {**return** salt;}

**public** **double** GetSquare() {**return** square;}

**public** **double** GetVolume() {**return** volume;}

**public** **double** GetTemperature(){**return** temperature;}

**public** **void** SetName(String Name) {name=Name;}

**public** **void** SetOcean(String Ocean) {ocean=Ocean;}

**public** **void** SetCountries(String Countries) {countries=Countries;}

**public** **void** SetSalt(**double** Salt) {salt=Salt;}

**public** **void** SetSquare(**double** Square) {square=Square;}

**public** **void** SetVolume(**double** Volume) {volume=Volume;}

**public** **void** SetTemperature(**double** Temperature){temperature=Temperature;}

CSea(String Name){

name=Name;

}

CSea(String Name, String Ocean,**double** Square, **double** Salt){

**this**(Name);

ocean=Ocean;

square=Square;

salt=Salt;

}

CSea(String Name,String Ocean,String Countries,**double** Square,**double** Salt,**double** Volume,**double** Temperature){

**this**(Name, Ocean, Square, Salt);

countries=Countries;

volume=Volume;

temperature=Temperature;

}

**public** **static** **void** SortByName(CSea array[]){

CSortByNameComp comp = **new** CSortByNameComp();

java.util.Arrays.*sort*(array, comp);

}

**public** **static** **void** SortByOcean(CSea array[]){

CSortByOceanComp comp = **new** CSortByOceanComp();

java.util.Arrays.*sort*(array, comp);

}

**public** **static** **void** CSortBySquare(CSea array[]){

CSortBySquareComp comp = **new** CSortBySquareComp();

java.util.Arrays.*sort*(array, comp);

}

**public** **static** **void** SortByTemperature(CSea array[]){

CSortByTemperatureComp comp = **new** CSortByTemperatureComp();

java.util.Arrays.*sort*(array, comp);

}

**public** String toString(){

String s=String.*format*("%s %s %s %s %.2f %.4f %.1f\n", name, ocean, countries, square, salt, volume, temperature);

**return** s;

}

}

// CSortByNameComp.java

**import** java.util.Comparator;

**class** CSortByNameComp **implements** Comparator<CSea>{

**public** **int** compare(CSea o1, CSea o2){

**return** o1.GetName().compareTo(o2.GetName());

}

}

// CSortBySquareComp.java

**import** java.util.Comparator;

**class** CSortBySquareComp **implements** Comparator<CSea>{

**public** **int** compare(CSea o1, CSea o2){

**double** r=o1.GetSquare()- o2.GetSquare();

**if** (r>0) **return** 1;

**if** (r<0) **return** -1;

**return** 0;

}

}

// CSortByOceanComp.java

**import** java.util.Comparator;

**class** CSortByOceanComp **implements** Comparator<CSea>{

**public** **int** compare(CSea o1, CSea o2){

**return** o1.GetOcean().compareTo(o2.GetOcean());

}

}

// CSortByTemperatureComp.java

**import** java.util.Comparator;

**class** CSortByTemperatureComp **implements** Comparator<CSea>{

**public** **int** compare(CSea o1, CSea o2){

**double** r=o1.GetTemperature()- o2.GetTemperature();

**if** (r>0) **return** 1;

**if** (r<0) **return** -1;

**return** 0;

}

}

Приклад роботи:

a b c 5.3 12,00 4,0000 2,0

c d f 3.4 6,50 0,5000 8,3

f a d 2.3 6,00 0,0100 2,3

f a d 2.3 6,00 0,0100 2,3

c d f 3.4 6,50 0,5000 8,3

a b c 5.3 12,00 4,0000 2,0

f a d 2.3 6,00 0,0100 2,3

a b c 5.3 12,00 4,0000 2,0

c d f 3.4 6,50 0,5000 8,3

a b c 5.3 12,00 4,0000 2,0

f a d 2.3 6,00 0,0100 2,3

c d f 3.4 6,50 0,5000 8,3

**Висновок:** під час виконання цієї роботи я дізнався про стандартні методи сортування масивів об’єктів власного класу за допомогою стандартного методу та інтерфейсу компаратора. Також покращив навички документування коду за допомогою джавадок