Міністерство освіти і науки України

Національний університет „Львівська політехніка”

Кафедра ЕОМ



**Звіт**

з лабораторної роботи №7

з дисципліни: “Кросплатформні засоби програмування”

на тему: “Параметризоване програмування”

Виконав: ст. гр. КІ-34

Зубалій І.А.

Прийняв:

Іванов Ю.С.

Львів – 2022

**Мета:** оволодіти навиками параметризованого програмування мовоюJava.

**Індивідуальне завдання:** створити параметризований клас, що реалізує предметну область задану варіантом:

**5. Двозв’язний список**

**Хід роботи:**

1. Запустив середовище Eclipse та написав програму згідно індивідуального завдання:

*Driver.java*

/\*\*

\* lab 7 package

\*/

**package** lab7;

/\*\*

\* Driver class implements main method for MyDoubleLinkedList class possibilities demonstration

\*

\* **@version** 1.0

\*/

**public** **class** Driver {

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

MyDoubleLinkedList<? **super** Data<?>> list = **new** MyDoubleLinkedList<Data<?>>();

list.addFirst(**new** StringData("A"));

list.addFirst(**new** NumberData(1));

list.addLast(**new** StringData("B"));

list.addLast(**new** StringData("X"));

list.removeLast();

list.addLast(**new** StringData("C"));

list.addLast(**new** NumberData(2));

list.addLast(**new** FloatData(6.0f));

list.addFirst(**new** NumberData(3));

list.printList();

Data<?> data = list.findMax();

System.***out***.println("The greatest data is: ");

data.print();

}

}

*MyDoubleLinkedList.java*

/\*\*

\* lab 7 package

\*/

**package** lab7;

**import** java.util.LinkedList;

/\*\*

\* Class MyDoubleLinkedList implements double-linked list

\*

\* **@version** 1.0

\*/

**public** **class** MyDoubleLinkedList<T **extends** Data<?>> {

**private** LinkedList<T> list;

/\*\*

\* Constructor

\*/

MyDoubleLinkedList() {

list = **new** LinkedList<T>();

}

/\*\*

\* Method returns the greatest data

\*

\* **@return** Data

\*/

**public** T findMax() {

T max = **null**;

**for** (T t : list) {

**if** (t **instanceof** NumberData || t **instanceof** FloatData)

max = t;

}

**if** (!list.isEmpty()) {

**for** (T t : list)

**if** (t.compareTo(max) > 0)

max = t;

**return** max;

}

**return** **null**;

}

/\*\*

\* Method prints all items of list

\*/

**public** **void** printList() {

**for** (T t : list)

System.***out***.print(t.getValue() + " ");

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

}

/\*\*

\* Method prints the item of list

\*

\* **@param** index

\*/

**public** **void** printItem(**int** index) {

System.***out***.println(list.get(index));

}

/\*\*

\* Method gets the item of list

\*

\* **@param** index

\* **@return** Data

\*/

**public** T get(**int** index) {

**return** list.get(index);

}

/\*\*

\* Method adds item to the beginning of list

\*

\* **@param** t - item

\*/

**public** **void** addFirst(T t) {

list.addFirst(t);

}

/\*\*

\* Method adds item to the end of list

\*

\* **@param** t - item

\*/

**public** **void** addLast(T t) {

list.addLast(t);

}

/\*\*

\* Method adds item to the list

\*

\* **@param** index

\* **@param** t - item

\*/

**public** **void** add(**int** index, T t) {

list.add(index, t);

}

/\*\*

\* Method removes item from the beginning of list

\*/

**public** **void** removeFirst() {

list.removeFirst();

}

/\*\*

\* Method removes item from the end of list

\*/

**public** **void** removeLast() {

list.removeLast();

}

/\*\*

\* Method removes item from the list

\*

\* **@param** index

\*/

**public** **void** remove(**int** index) {

list.remove(index);

}

}

*Data.java*

/\*\*

\* lab 7 package

\*/

**package** lab7;

/\*\*

\* Interface Datar describes rules for data type

\*

\* **@version** 1.0

\*/

**interface** Data<T> {

/\*\*

\* Method prints data

\*/

**public** **void** print();

/\*\*

\* Method returns data

\*/

**public** T getValue();

/\*\*

\* Method compares current Data with another Data

\*

\* **@param** Data

\*/

**public** **int** compareTo(Data<?> d);

}

*NumberData.java*

/\*\*

\* lab 7 package

\*/

**package** lab7;

/\*\*

\* Class NumberData implements numeric type of data

\*

\* **@version** 1.0

\*/

**public** **class** NumberData **implements** Data<Integer> {

**private** Integer number;

/\*\*

\* Constructor

\*/

**public** NumberData() {

number = 0;

}

/\*\*

\* Constructor

\*

\* **@param** number

\*/

**public** NumberData(Integer number) {

**this**.number = number;

}

/\*\*

\* **@return** the number

\*/

**public** Integer getValue() {

**return** number;

}

/\*\*

\* **@param** number the number to set

\*/

**public** **void** set(Integer number) {

**this**.number = number;

}

/\*\*

\* Method compares this NumberData with another Data

\*

\* **@param** o - data

\*/

@Override

**public** **int** compareTo(Data<?> o) {

**if** (Float.*parseFloat*(**this**.getValue().toString()) > Float.*parseFloat*(o.getValue().toString()))

**return** 1;

**if** (**this**.getValue().equals(o.getValue()))

**return** 0;

**return** -1;

}

/\*\*

\* Method prints number

\*/

@Override

**public** **void** print() {

System.***out***.println("Number: " + number);

}

}

*StringData.java*

/\*\*

\* lab 7 package

\*/

**package** lab7;

/\*\*

\* Class StringData implements string type of data

\*

\* **@version** 1.0

\*/

**public** **class** StringData **implements** Data<String> {

**private** String string;

/\*\*

\* Constructor

\*/

StringData() {

string = "";

}

/\*\*

\* Constructor

\*

\* **@param** string

\*/

**public** StringData(String string) {

**this**.string = string;

}

/\*\*

\* **@return** the string

\*/

**public** String getValue() {

**return** string;

}

/\*\*

\* **@param** string the string to set

\*/

**public** **void** set(String string) {

**this**.string = string;

}

/\*\*

\* Method compares this NumberData with another Data

\*

\* **@param** o - data

\*/

@Override

**public** **int** compareTo(Data<?> o) {

**return** -1;

}

/\*\*

\* Method prints string

\*/

@Override

**public** **void** print() {

System.***out***.println("String: " + string);

}

}

*FloatData.java*

/\*\*

\* lab 7 package

\*/

**package** lab7;

/\*\*

\* Class FloatData implements numeric type of data with floating point

\*

\* **@version** 1.0

\*/

**public** **class** FloatData **implements** Data<Float> {

**private** Float fnumber;

/\*\*

\* Constructor

\*/

**public** FloatData() {

}

/\*\*

\* **@return** the fnumber

\*/

**public** Float getValue() {

**return** fnumber;

}

/\*\*

\* **@param** fnumber the fnumber to set

\*/

**public** **void** set(Float fnumber) {

**this**.fnumber = fnumber;

}

/\*\*

\* Constructor

\*

\* **@param** fnumber

\*/

**public** FloatData(Float fnumber) {

**this**.fnumber = fnumber;

}

/\*\*

\* Method compares this NumberData with another Data

\*

\* **@param** o - data

\*/

@Override

**public** **int** compareTo(Data<?> o) {

**if** (Float.*parseFloat*(**this**.getValue().toString()) > Float.*parseFloat*(o.getValue().toString()))

**return** 1;

**if** (**this**.getValue().equals(o.getValue()))

**return** 0;

**return** -1;

}

/\*\*

\* Method prints float

\*/

@Override

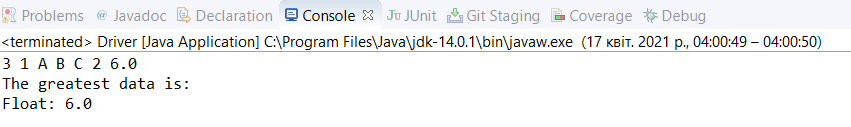
**public** **void** print() {

System.***out***.println("Float: " + fnumber);

}

}

**Результат виконання програми (в консолі середовища розробки):**



*Рис.1. Результат виконання програми*

**Висновок:**

На даній лабораторній роботі оволодів навиками параметризованого програмування мовоюJava.