Міністерство освіти і науки України

Національний Університет «Львівська Політехніка»

Кафедра АСУ

Звіт до лабораторної роботи №2

з курсу: **«Інноваційні інформаційні технології»**

на тему:

**«Перевантаження методів класу і операцій»**

**Виконав:**

студент групи КНМ-12

Новосад Б. І.

**Перевірив:**

старший викладач

Ковальчук А.М.

Львів 2015

Мета : Ознайомлення зі змістом поняття **'перевантаження'** в

С++ та використання **'friend-функцій'.**

Методи класів, так як і звичайні функції С++, можна пере­вантажувати. Перевантаження функцій означає, що в поточній об­ласті дії одне і теж ім'я можуть використовувати декілька функцій. Компілятор вибирає потрібну функцію, враховуючи кількість і тип аргументів, використаних при її викликові.

В приведеному прикладі значення кута передається методам класу в двох форматах: як числа типу double або у вигляді строки. За допомогою перевантаження методів можна опрацьовувати обидва ці типи.

Вибрати два завдання з номерами: ***n*1** = порядковому номеру студента в групі, ***n*2**= **⏐*N* + 2 - 2*n*1⏐**, де ***N*** **-** число студентів в групі.

***n*1 =** 18

***n*2 = ⏐*29* + 2 – 2\**18*⏐** = 5

**Завдання 5 та 18.**

* Задана послідовність з ***n*** нерівних між собою чисел. Додавання визначається як сума чисел цієї послідовності, розміщених між максимальним і мінімальним числами ( в суму не включати ці два числа).
* У наступних завданнях, здійснивши табуляцію деякої функції, виконати наступне:Вивести номер передостаннього додатного елемента.

Хід роботи

**Клас Tabulation**

**package** com.company;  
  
  
**import** java.util.ArrayList;  
**import** java.util.Arrays;  
  
*/\*\*  
 \* Created by Bohdan Novosad on 21.09.2016.  
 \*/***public class** Tabulation {  
 **private double a**,**b**,**h**;  
 **private int k** = 0;  
 **private** Function **function**;  
 **private** ArrayList<Point> **data** = **new** ArrayList<>();  
  
 **public** Tabulation(**double** a, **double** b, **double** h, Function function){  
 **this**.**a**=a;  
 **this**.**b**=b;  
 **this**.**h**=h;  
 **this**.**function** = function;  
 **this**.**k** = (**int**) (((b-a)/h)+1);  
 }  
  
 **public void** doIt(){  
 **int** i=0;  
 **for**(**double** a = **this**.**a**;a<**this**.**b**;a+=**h**){  
 Point point = **new** Point(a, **function**.compute(a));  
 **data**.add(point);  
 System.***out***.println(point);  
 }  
 }  
  
 **public** Point getPreLastPoint(){  
 **int** lich = 0;  
 **int** pos = -1;  
 **for**(**int** i=**data**.size()-1;i>=0;i--){  
  
 **if**(**data**.get(i).getY()>0)  
 lich++;  
 **if**(lich == 2)  
 pos = i;  
 **if**(lich==2)  
 **break**;  
 }  
  
 **if**(pos==-1){  
 **return null**;  
 }**else** {  
 **return data**.get(pos);  
 }  
 }  
  
 @Override  
 **public** String toString() {  
 **return "Tabulation{"** +  
 **"data="** + **data** +  
 **'}'**;  
 }  
}

**Клас Point**

**package** com.company;  
  
*/\*\*  
 \* Created by Bohdan Novosad on 21.09.2016.  
 \*/***public class** Point {  
 **private double x**,**y**;  
  
 **public double** getX() {  
 **return x**;  
 }  
  
 **public void** setX(**double** x) {  
 **this**.**x** = x;  
 }  
  
 **public double** getY() {  
 **return y**;  
 }  
  
 **public void** setY(**double** y) {  
 **this**.**y** = y;  
 }  
  
 **public** Point(**double** x, **double** y) {  
 **this**.**x** = x;  
 **this**.**y** = y;  
 }  
  
 @Override  
 **public** String toString() {  
 **return "Point{"** +  
 **"x="** + **x** +  
 **", y="** + **y** +  
 **'}'**;  
 }  
  
 @Override  
 **public boolean** equals(Object o) {  
 **if** (**this** == o) **return true**;  
 **if** (o == **null** || getClass() != o.getClass()) **return false**;  
  
 Point point = (Point) o;  
  
 **if** (Double.*compare*(point.**x**, **x**) != 0) **return false**;  
 **return** Double.*compare*(point.**y**, **y**) == 0;  
  
 }  
  
 @Override  
 **public int** hashCode() {  
 **int** result;  
 **long** temp;  
 temp = Double.*doubleToLongBits*(**x**);  
 result = (**int**) (temp ^ (temp >>> 32));  
 temp = Double.*doubleToLongBits*(**y**);  
 result = 31 \* result + (**int**) (temp ^ (temp >>> 32));  
 **return** result;  
 }  
}

**Використання класів.**

**package** com.company;  
  
**public class** Main {  
  
 **public static void** main(String[] args) {  
 *// write your code here* Tabulation tabulation = **new** Tabulation(-100.0,100.0,7.5, **new** MySin());  
 tabulation.doIt();  
 Point point = tabulation.getPreLastPoint();  
  
 *print*(tabulation);  
  
 *print*(point);  
  
 }  
 **static void** print(Point point){  
 System.***out***.println(point);  
 }  
  
 **static void** print(Tabulation tabulation){  
 System.***out***.println(tabulation);  
 }  
}

**Результат роботи програми**

**….**

**Point{x=50.0, y = -0.26237485370392877}**

**Point{x=57.5, y = 0.8141897215084345}**

**Point{x=65.0, y = 0.8268286794901034}**

**Point{x=72.5, y = -0.2409736772881011}**

**Point{x=80.0, y = -0.9938886539233752}**

**Point{x=87.5, y = -0.44806014160260915}**

**Point{x=95.0, y = 0.683261714736121}**

**Tabulation{data=[…]}**

**Point{x=65.0, y=0.8268286794901034}**

**Завдання №2**

**Клас Sequence**

**import** java.util.ArrayList;  
**import** java.util.List;

*/\*\*  
 \* Created by Bohdan Novosad on 21.09.2016.  
 \*/***public class** Sequence {  
 ArrayList<Integer> **sequenceValue** = **new** ArrayList<>();  
  
 **public** Sequence(List<Integer> integers){  
 **sequenceValue**.addAll(integers);  
 }  
  
 **public** Integer getSum(){  
 **int** rez = 0;  
 **int** min = getMin(**sequenceValue**);  
 **int** max = getMax(**sequenceValue**);  
 **int** firs\_index, last\_index;  
 **if**(min<max){  
 firs\_index = min;  
 last\_index = max;  
 }**else**{  
 firs\_index = max;  
 last\_index = min;  
 }  
  
 **for** (**int** i=firs\_index+1;i<last\_index;i++){  
 rez+=**sequenceValue**.get(i);  
 }  
 **return** rez;  
 }  
  
 **private int** getMax(ArrayList<Integer> sequenceValue) {  
 Integer max = sequenceValue.get(0);  
 **int** pos = 0;  
 **for**(**int** i=1;i<sequenceValue.size();i++){  
 **if**(max>sequenceValue.get(i)){  
 max = sequenceValue.get(i);  
 pos = i;  
 }  
 }  
 **return** pos;  
 }  
  
 **private int** getMin(ArrayList<Integer> sequenceValue) {  
 Integer min = sequenceValue.get(0);  
 **int** pos = 0;  
 **for**(**int** i=1;i<sequenceValue.size();i++){  
 **if**(min<sequenceValue.get(i)){  
 min = sequenceValue.get(i);  
 pos = i;  
 }  
 }  
 **return** pos;  
 }  
  
 @Override  
 **public** String  
 toString() {  
 **return "Sequence{"** +  
 **"sequenceValue="** + **sequenceValue** +  
 **'}'**;  
 }

**Використання класу**

**package** com.company;  
  
**import** java.util.ArrayList;  
**import** java.util.Random;  
  
**public class** Main {  
  
 **public static void** main(String[] args) {  
 *// write your code here* ArrayList<Integer> data = **new** ArrayList<>();  
 Random random = **new** Random();  
  
 **for**(**int** i=0;i<10;i++){  
 data.add(random.nextInt()%100);  
 }  
  
 Sequence sequence = **new** Sequence(data);  
  
 System.***out***.println(sequence);  
  
 System.***out***.println(sequence.getSum());  
  
 }  
}

**Результат роботи програми.**

Sequence{sequenceValue=[-86, -36, 61, -55, 60, -80, 8, -45, -75, 88]}

-162

**Висновок**: перевантаження методів це дуже корисний інструментарій в руках досвідченого програміста. Це дає змогу використовувати одне ім’я для різних цілей, і це покращує розуміння коду. На лабораторній роботі перед мною стояло два завдання, які я успішно виконав, і використав набуті знання на прктиці.