**Міністерство освіти і науки, молоді та спорту України**

**Національний технічний університет України**

**«Київський політехнічний інститут»**

**Факультет інформатики та обчислювальної техніки**

**Кафедра обчислювальної техніки**

**Лабораторна робота №4**

**Тема: «Класи в мові програмування Java.»**

**Виконав: Студент групи ІО-33**

**Заколенко Р.К.**

**Перевірив: Викладач**

**Невдащенко М.В.**

**Київ 2014**

1. **Варіант завдання**

Номер залікової книжки – 3308

С11 = 8 (Визначити клас меблі, який складається як мінімум з 5-и полів.)

Створити клас із виконавчим методом, в якому створити масив з об’єктів класу визначеному варіантом. Відсортувати масив, за одним з полів, за зростання, а за іншим, за спаданням використавши при цьому стандартні засоби сортування.

1. **Опис програми**

Програма містить п’ять класів:

1. Клас Furniture (Меблі);
2. Клас SortByLength (сортування об’єктів класу Furniture за полем «length»);
3. Клас SortByHeight (сортування об’єктів класу Furniture за полем «height») ;
4. Клас SortByType (сортування об’єктів класу Furniture за полем «type») ;
5. Клас із виконавчим методом.

У виконавчому методі ми створюємо об’єкти класу Furniture, з яких формує масив.

Далі сортуємо елементи масиву за трьома полями «length», «height», «type» та після кожного сортування виводимо на екран значення відсортованих полів. Сортування відбувається за допомогою стандартних засобів сортування, класу Arrays. Класи SortByLength, SortByHeight, SortByType реалізують інтерфейс Comparator, у кожному з цих класів реалізований метод compare таким чином, щоб відбувалося відповідне сортування.

1. **Код програми**

**package** ua.kpi.io33.zakolenko.lab4;

/\*\*

\* This class describe data about object Furniture.

\*

\* **@author** Roman Zakolenko

\* **@version** 1,6 07 Apr 2014

\*/

**public** **class** Furniture {

**public** String type;

**public** String color;

**public** **int** length;

**public** **int** height;

**public** **int** width;

/\*\*

\* **@param** t

\* this variable is assigned to the field "type".

\* **@param** c

\* this variable is assigned to the field "color".

\* **@param** l

\* this variable is assigned to the field "height".

\* **@param** h

\* this variable is assigned to the field "height".

\* **@param** w

\* this variable is assigned to the field "width".

\*/

**public** Furniture(String t, String c, **int** l, **int** h, **int** w) {

type = t;

color = c;

length = l;

height = h;

width = w;

}

}

**package** ua.kpi.io33.zakolenko.lab4;

**import** java.util.Comparator;

/\*\*

\* This class uses an interface Comparator and overrides the method compare,

\* that is used to sort the Object class Furniture by field "length".

\*

\* **@author** Roman Zakolenko

\* **@version** 1,6 07 Apr 2014

\*/

**public** **class** SortByLength **implements** Comparator<Furniture> {

/\*\*

\***@return** type of Integer.

\*Overrides the method compare for sort massive of object

\***@param** obj1,obj2 are objects of class Furniture

\*/

**public** **int** compare(Furniture obj1, Furniture obj2) {

**return** obj2.length - obj1.length;

}

}

**package** ua.kpi.io33.zakolenko.lab4;

**import** java.util.Comparator;

/\*\*

\* This class uses an interface Comparator and overrides the method compare,

\* that is used to sort the Object class Furniture by field "type".

\*

\* **@author** Roman Zakolenko

\* **@version** 1,6 07 Apr 2014

\*/

**public** **class** SortByType **implements** Comparator<Furniture> {

/\*\*

\***@return** type of Integer.

\*Overrides the method compare for sort massive of object.

\***@param** obj1,obj2 are objects of class Furniture.

\*/

**public** **int** compare(Furniture obj1, Furniture obj2) {

**return** obj1.type.toLowerCase().compareTo(obj2.type.toLowerCase());

}

}

**package** ua.kpi.io33.zakolenko.lab4;

**import** java.util.Comparator;

/\*\*

\* This class uses an interface Comparator and overrides the method compare,

\* that is used to sort the Object class Furniture by field "height".

\*

\* **@author** Roman Zakolenko

\* **@version** 1,6 07 Apr 2014

\*/

**public** **class** SortByHeight **implements** Comparator<Furniture> {

/\*\*

\***@return** type of Integer.

\*Overrides the method compare for sort massive of object.

\***@param** obj1,obj2 are objects of class Furniture.

\*/

**public** **int** compare(Furniture obj1, Furniture obj2) {

**return** obj1.height - obj2.height;

}

}

**package** ua.kpi.io33.zakolenko.lab4;

**import** java.util.Arrays;

/\*\*

\* Is used to create objects of class Furniture, as well as for sorting objects

\* of class Furniture for their fields.

\*

\* **@author** Roman Zakolenko

\* **@version** 1,7 07 Apr 2014

\* **@see** uk.kpi.io33.zakolenko.lab4.Furniture

\* uk.kpi.io33.zakolenko.lab4.SortByLength

\* uk.kpi.io33.zakolenko.lab4.SortByType

\* uk.kpi.io33.zakolenko.lab4.SortByHeight

\*/

**public** **class** Lab4 {

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

Furniture table1 = **new** Furniture("table", "brown", 15, 42, 65);

Furniture table2 = **new** Furniture("table", "green", 25, 44, 66);

Furniture table3 = **new** Furniture("table", "red", 34, 43, 25);

Furniture chair1 = **new** Furniture("chair", "brown", 10, 21, 11);

Furniture chair2 = **new** Furniture("chair", "white", 12, 4, 34);

Furniture Array[] = { table1, table2, table3, chair1, chair2 };

Arrays.*sort*(Array, **new** SortByType());

System.*out*.println("Lexicographical sorting:");

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

System.*out*.println(Array[i].type.toLowerCase());

}

Arrays.*sort*(Array, **new** SortByLength());

System.*out*.println("Sorting by length:");

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

System.*out*.println(Array[i].length);

}

Arrays.*sort*(Array, **new** SortByHeight());

System.*out*.println("Sorting by height:");

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

System.*out*.println(Array[i].height);

}

}

}

**4.Висновок**

Під час виконання лабораторної роботи я ознайомився із класами в мові програмування Java, а саме навчився їх створювати та створювати конструктори даного класу. Було створено конструктор класу Furniture, параметри якого зв’язані із полями даного класу, тобто при створенні об’єкту класу Furniture за допомогою параметрів цього конструктора ми задаємо значення полів об’єкта.

Я ознайомився із інтерфейсами на прикладі інтерфейсу Comparator та реалізував метод compare відповідно до вимог сортування.