Міністерство освіти і науки України

Національний університет “Львівська політехніка”

Кафедра ЕОМ



**Звіт**

З лабораторної роботи №6

Варіант – 9

З дисципліни: «Кросплатформні засоби програмування»

На тему: «ПАРАМЕТРИЗОВАНЕ ПРОГРАМУВАННЯ »

Виконав: ст. гр. КІ-305

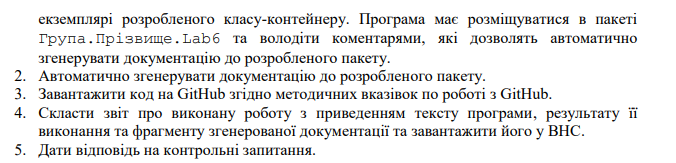
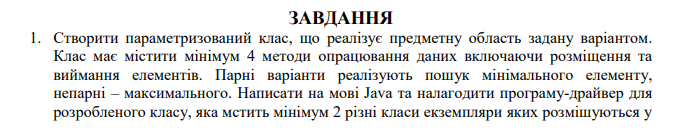
Заставний Р.А.

Прийняв:

доцент кафедри ЕОМ

Іванов Ю. С.

Львів-2023

**Мета роботи:** оволодіти навиками параметризованого програмування мовою Java 

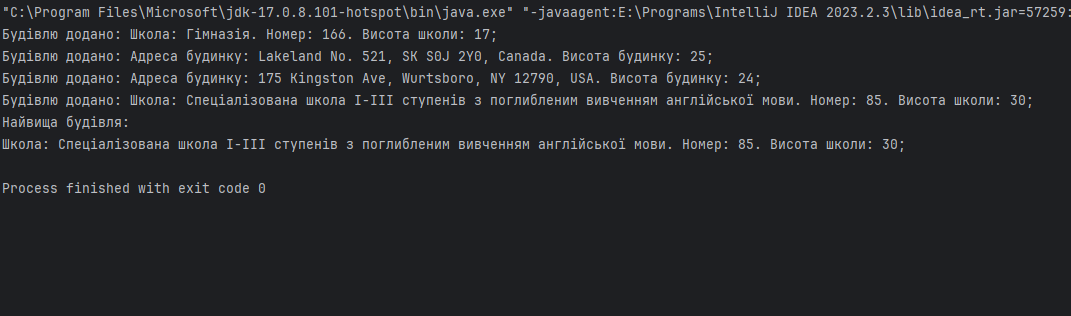
**Варіант завдання:**

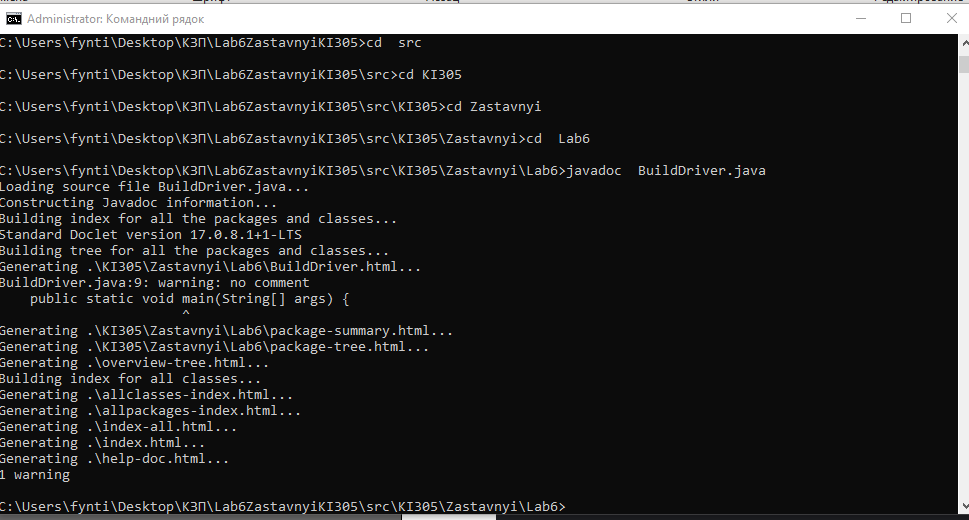


**Код програми:**

package KI305.Zastavnyi.Lab6;  
  
import java.util.ArrayList;  
  
*/\*\*  
 \* Клас BuildDriver демонструє використання класу Building з різними типами будівель.  
 \*/*public class BuildDriver {  
 public static void main(String[] args) {  
 Building<? super Data> build = new Building<>();  
 build.AddBuild(new School("Гімназія", 166, 17));  
 build.AddBuild(new House("Lakeland No. 521, SK S0J 2Y0, Canada", 25));  
 build.AddBuild(new House("175 Kingston Ave, Wurtsboro, NY 12790, USA", 24));  
 build.AddBuild(new School("Спеціалізована школа I-III ступенів з поглибленим вивченням англійської мови", 85, 30));  
 Data res = build.findMax();  
 System.*out*.print("Найвища будівля: \n");  
 res.print();  
 }  
}  
  
*/\*\*  
 \* Клас Building представляє колекцію будівель типу T.  
 \*  
 \* @param* <*T*> *Тип будівель, які можуть бути збережені в колекції.  
 \*/*class Building<T extends Data> {  
 private ArrayList<T> arr;  
  
 */\*\*  
 \* Конструктор для ініціалізації ArrayList.  
 \*/* public Building() {  
 arr = new ArrayList<>();  
 }  
  
 */\*\*  
 \* Знаходить і повертає будівлю з максимальною висотою.  
 \*  
 \* @return Будівля з максимальною висотою.  
 \*/* public T findMax() {  
 if (!arr.isEmpty()) {  
 T max = arr.get(0);  
 for (int i = 1; i < arr.size(); i++) {  
 if (arr.get(i).compareTo(max) > 0)  
 max = arr.get(i);  
 }  
 return max;  
 }  
 return null;  
 }  
  
 */\*\*  
 \* Додає будівлю типу T до колекції.  
 \*  
 \* @param data Будівля для додавання.  
 \*/* public void AddBuild(T data) {  
 arr.add(data);  
 System.*out*.print("Будівлю додано: ");  
 data.print();  
 }  
  
 */\*\*  
 \* Видаляє будівлю за вказаним індексом з колекції.  
 \*  
 \* @param i Індекс будівлі, яку слід видалити.  
 \*/* public void DeleteData(int i) {  
 arr.remove(i);  
 }  
}  
  
*/\*\*  
 \* Інтерфейс Data представляє загальні функції для об'єктів даних з висотою.  
 \*/*interface Data extends Comparable<Data> {  
 */\*\*  
 \* Отримує висоту об'єкта даних.  
 \*  
 \* @return Висота об'єкта даних.  
 \*/* int getHeight();  
  
 */\*\*  
 \* Виводить інформацію про об'єкт даних.  
 \*/* void print();  
}  
  
*/\*\*  
 \* Клас School представляє школу з ім'ям, номером і висотою.  
 \*/*class School implements Data {  
 private String schoolName;  
 private int schoolNumber;  
 private int height;  
  
 */\*\*  
 \* Конструктор для ініціалізації об'єкта School.  
 \*  
 \* @param sName Назва школи.  
 \* @param sNumber Номер школи.  
 \* @param sHeight Висота школи.  
 \*/* public School(String sName, int sNumber, int sHeight) {  
 schoolName = sName;  
 schoolNumber = sNumber;  
 height = sHeight;  
 } public int getHeight()  
 {  
 return height;  
 }  
  
 // Гетери та сетери для властивостей School...  
  
 */\*\*  
 \* Порівнює висоту школи з іншим об'єктом Data.  
 \*  
 \* @param p Об'єкт Data для порівняння.  
 \* @return Від'ємне ціле число, нуль або додатнє ціле число, якщо цей об'єкт менший, рівний або більший за вказаний об'єкт.  
 \*/* public int compareTo(Data p) {  
 return Integer.*compare*(height, p.getHeight());  
 }  
  
 */\*\*  
 \* Виводить інформацію про школу.  
 \*/* public void print() {  
 System.*out*.print("Школа: " + schoolName + ". Номер: " + schoolNumber +  
 ". Висота школи: " + height + ";\n");  
 }  
}  
  
*/\*\*  
 \* Клас House представляє будинок з адресою і висотою.  
 \*/*class House implements Data {  
 private String address;  
 private int height;  
  
 */\*\*  
 \* Конструктор для ініціалізації об'єкта House.  
 \*  
 \* @param hAddress Адреса будинку.  
 \* @param hHeight Висота будинку.  
 \*/* public House(String hAddress, int hHeight) {  
 address = hAddress;  
 height = hHeight;  
 }  
  
 // Гетери та сетери для властивостей House...  
  
 */\*\*  
 \* Порівнює висоту будинку з іншим об'єктом Data.  
 \*  
 \* @param p Об'єкт Data для порівняння.  
 \* @return Від'ємне ціле число, нуль або додатнє ціле число, якщо цей об'єкт менший, рівний або більший за вказаний об'єкт.  
 \*/* public int compareTo(Data p) {  
 return Integer.*compare*(height, p.getHeight());  
 }  
 public int getHeight()  
 {  
 return height;  
 }  
 */\*\*  
 \* Виводить інформацію про будинок.  
 \*/* public void print() {  
 System.*out*.print("Адреса будинку: " + address + ". Висота будинку: " + height + ";\n");  
 }  
}

**Результата роботи програми:**



**Фрагмент згенерованої документації**

**Висновок:** на цій лабораторній роботі я оволодів навиками параметризованого програмування мовою Java