# Міністерство освіти і науки України

Національний університет „Львівська політехніка”

# Кафедра ЕОМ



Звіт

до лабораторної роботи № 6

з дисципліни: «Кросплатформні засоби програмування»

«Параметризоване програмування» Варіант - 1

Виконав:

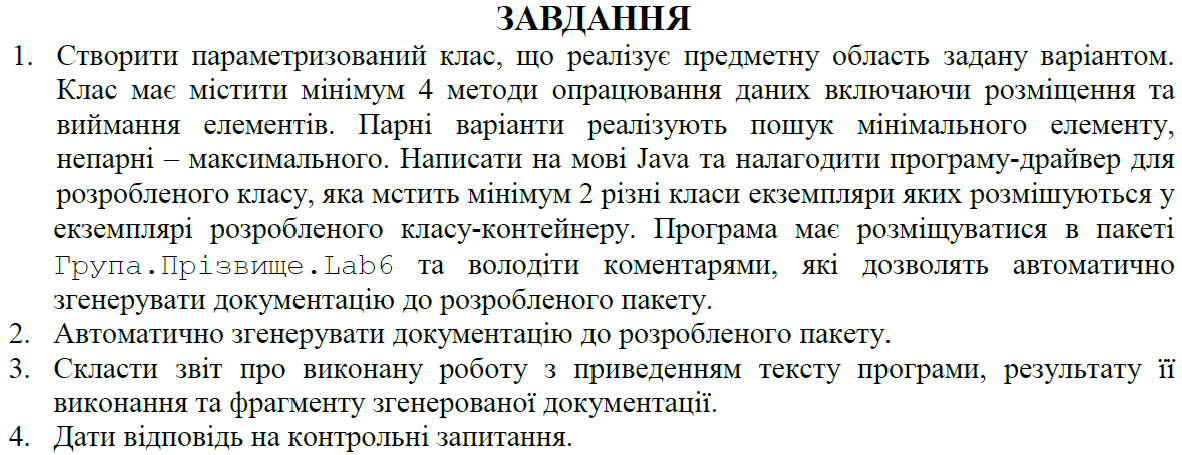
Студент групи КІ-305

Баран В. М.

Прийняв:

Іванов Ю. С.

# Львів 2023



**Варіант завдання:**



# Код програми:

**File App.java**

package KI305.Vozniuk.Lab6;  
  
*/\*\*  
 \* Class Lab6VozniukKI305 realize example program to lab work 6  
 \*  
 \* @author Vozniuk Oleg  
 \* @version 1.1  
 \*/*public class App {  
 public static void main(String[] args) {  
 Array<? super Info> Array = new Array<>();  
 Array.AddInfo(new Vector3D("Vec1", 120, 35));  
 Array.AddInfo(new Vector2D("Vec2", 100));  
 Array.AddInfo(new Vector2D("Vec3", 40));  
 Array.AddInfo(new Vector3D("Vec4", 95, 10));  
 Info res1 = Array.findMax();  
 System.*out*.print("The biggest vector in the Array has: \n");  
 res1.print();  
 }  
}

}

# File Array.java

package KI305.Vozniuk.Lab6;  
  
import java.util.ArrayList;  
*/\*\*  
 \* Parameterized class <code>Array</code> implements Array  
 \*/*class Array<T extends Info> {  
 private ArrayList<T> arr;  
  
 */\*\*  
 \* Constructor  
 \*/* public Array() {  
 arr = new ArrayList<T>();  
 }  
  
  
 */\*\*  
 \* Method find object with the biggest size  
 \*/* 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;  
 }  
  
 */\*\*  
 \* Method add new object to Array  
 \*  
 \* @param* <*code*>*Info</code> The class value  
 \*/* public void AddInfo(T Info) {  
 arr.add(Info);  
 System.*out*.print("Element added: ");  
 Info.print();  
 }  
 */\*\*  
 \* Method delete object from Array  
 \*  
 \* @param* <*code*>*i</code> The index of element of array in Array  
 \*/* public void DeleteInfo(int i) {  
 arr.remove(i);  
 }  
}

# File Vector3D.java

package KI305.Vozniuk.Lab6;  
  
class Vector3D implements Info {  
 private String VectorName;  
 private int height;  
 private int length;  
  
 */\*\*  
 \* Constructor  
 \*  
 \* @param* <*code*>*hName</code> The vector name  
 \* @param* <*code*>*pLive</code> The vector height  
 \* @param* <*code*>*pSize</code> The vector length  
 \*/* public Vector3D(String hName, int hHeight, int hLength) {  
 VectorName = hName;  
 height = hHeight;  
 length = hLength;  
 }  
  
 */\*\*  
 \* Method return Vector3D name  
 \*  
 \* @return VectorName  
 \*/* public String getVectorName() {  
 return VectorName;  
 }  
  
 */\*\*  
 \* Method set Vector name for Vector3D  
 \*  
 \* @param* <*code*>*name</code> The Vector name  
 \*/* public void setVectorName(String name) {  
 VectorName = name;  
 }  
  
 */\*\*  
 \* Method return length in the Vector3D  
 \*  
 \* @return length  
 \*/* public int getLength() {  
 return length;  
 }  
  
 */\*\*  
 \* Method set length in the Vector3D  
 \*  
 \* @param* <*code*>*n</code> The length  
 \*/* public void setLength(int n) {  
 length = n;  
 }  
  
 */\*\*  
 \* Method return height of Vector3D  
 \*  
 \* @return height  
 \*/* public int getHeight() {  
 return height;  
 }  
  
 */\*\*  
 \* Method compare objects type Info  
 \*  
 \* @param* <*code*>*p</code> The weight of object  
 \* @return num (0, 1, -1)  
 \*/* public int compareTo(Info p) {  
 Integer s = height;  
 return s.compareTo(p.getHeight());  
 }  
  
 */\*\*  
 \* Method print information about Vector3D  
 \*/* public void print() {  
 System.*out*.print("Vector3D: " + VectorName + ", Length: " + length +  
 ", Vector height: " + height + ";\n");  
 }  
  
  
}

# File Vector2D.java

package KI305.Vozniuk.Lab6;  
  
class Vector2D implements Info {  
 private String Vector2DName;  
 private int Vector2DHeight;  
  
 */\*\*  
 \* Constructor  
 \*  
 \* @param* <*code*>*sName</code> The name of Vector2D  
 \* @param* <*code*>*sWeight</code> The weight of Vector2D  
 \*/* public Vector2D(String sName, int sHeight) {  
 Vector2DName = sName;  
 Vector2DHeight = sHeight;  
 }  
  
 */\*\*  
 \* Method return name of Vector2D  
 \*  
 \* @return Vector2DName  
 \*/* public String getName() {  
 return Vector2DName;  
 }  
  
 */\*\*  
 \* Method set name of Vector2D  
 \*  
 \* @param* <*code*>*n</code> The Vector2DName  
 \*/* public void setName(String name) {  
 Vector2DName = name;  
 }  
  
 */\*\*  
 \* Method return height of Vector2D  
 \*  
 \* @return height  
 \*/* public int getHeight() {  
 return Vector2DHeight;  
 }  
  
 */\*\*  
 \* Method set height of Vector2D  
 \*  
 \* @param* <*code*>*n</code> The Vector2DHeight  
 \*/* public void setHeight(int n) {  
 Vector2DHeight = n;  
 }  
  
 */\*\*  
 \* Method compare objects type Info  
 \*  
 \* @param* <*code*>*p</code> The weight of object  
 \* @return num (0, 1, -1)  
 \*/* public int compareTo(Info p) {  
 Integer s = Vector2DHeight;  
 return s.compareTo(p.getHeight());  
 }  
  
 */\*\*  
 \* Method print information about Vector2D  
 \*/* public void print() {  
 System.*out*.print("Vector2D Name: " + Vector2DName + ",Vector2D Height: " + Vector2DHeight + ";\n");  
 }  
}

}

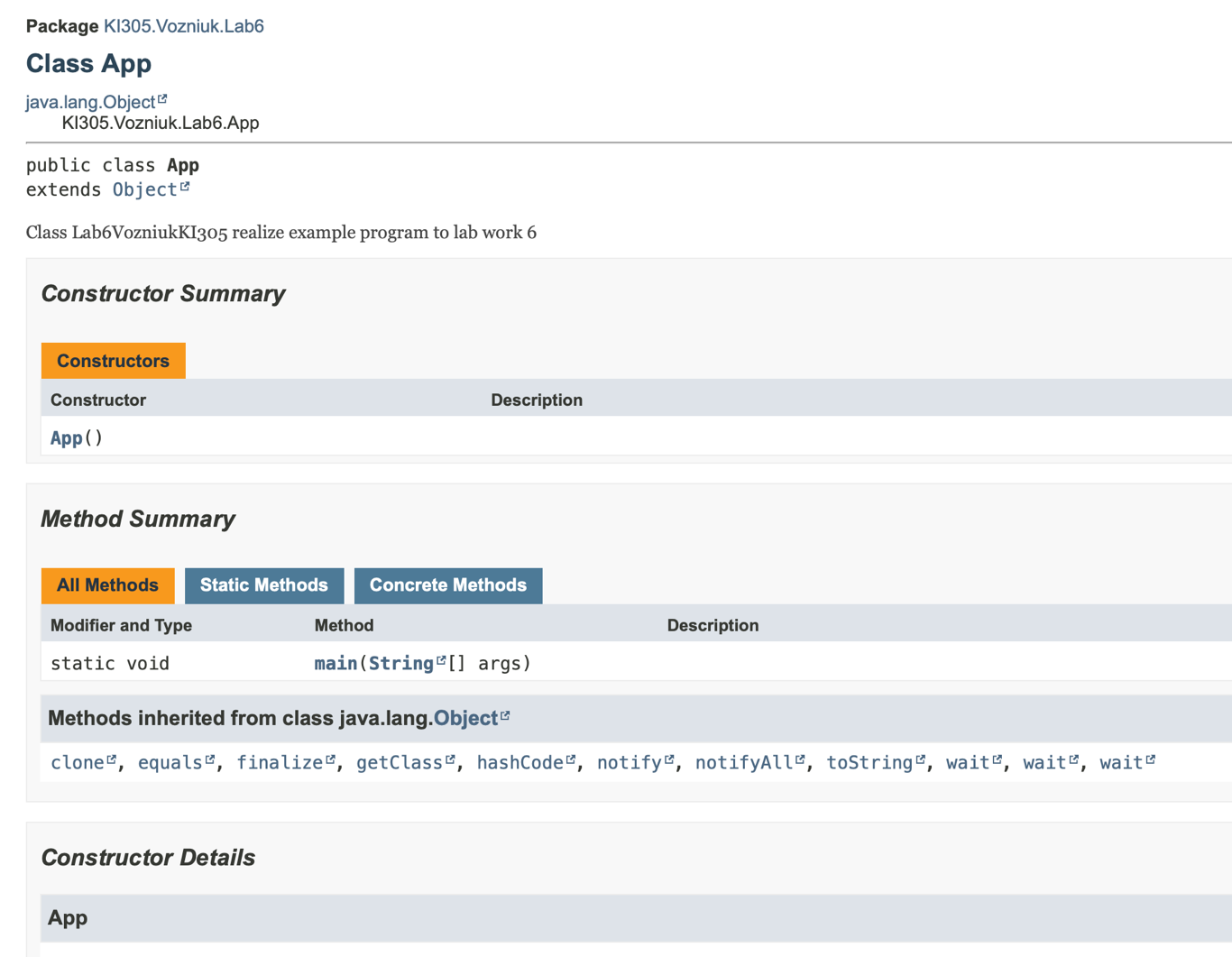
# File Info.java

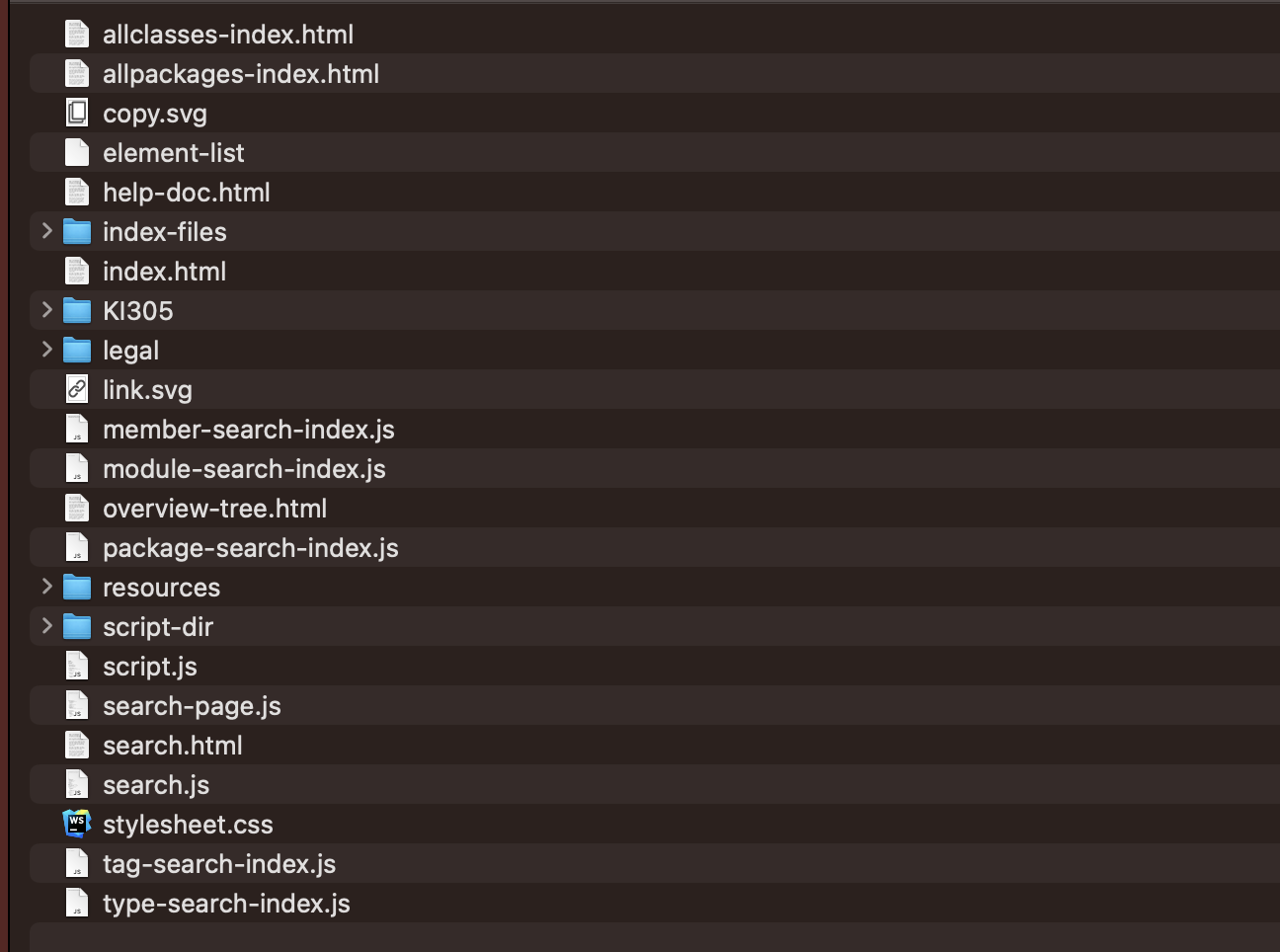
package KI305.Vozniuk.Lab6;  
  
*/\*\*  
 \* Interface <code>Info</code> implements Info  
 \*/*interface Info extends Comparable<Info> {  
 int getHeight();  
  
 void print();  
}

# Результат виконання роботи:

# 

# Фрагмент згенерованої документації:

****

****

**Відповіді на контрольні запитання**

**1. Дайте визначення терміну «параметризоване програмування».**

- це підхід до програмування, що дозволяє створювати класи і методи, які можна використовувати з різними типами даних, надаючи більшу гнучкість і безпеку типів у програмах.

**2. Розкрийте синтаксис визначення простого параметризованого класу.** public class НазваКласу<параметризованийТип> {

// Тіло класу }

**3. Розкрийте синтаксис створення об’єкту параметризованого класу. -**

НазваКласу<перелікТипів> зміннаКласу = new

НазваКласу<перелікТипів>(параметри);

**4. Розкрийте синтаксис визначення параметризованого методу.**

- public <параметризованийТип> типПовернення назваМетоду(параметри) {

// Тіло методу }

**5. Розкрийте синтаксис виклику параметризованого методу.**

- (НазваКласу|НазваОб’єкту).<перелікТипів>назваМетоду(параметри);

**6. Яку роль відіграє встановлення обмежень для змінних типів?**

- дозволяє заборонити використання деяких типів або вимагати, щоб тип

підставлений за замовчуванням був підкласом або реалізував певний інтерфейс.

**7. Як встановити обмеження для змінних типів?**

- за допомогою ключового слова extends для суперкласу або інтерфейсу, від яких має походити реальний тип.

8. Розкрийте правила спадкування параметризованих типів.

o - Всі класи, створені з параметризованого класу, незалежні один від

одного.

o - Зазвичай немає залежності між класами, створеними з різними

параметрами типів.

**9. Яке призначення підстановочних типів?**

- використовуються для забезпечення безпеки типів при використанні

параметризованих класів та методів. Вони дозволяють визначити, які типи

можна використовувати замість параметризованих типів.

**10. Застосування підставних типів.**

- <?> (unbounded wildcard) дозволяє читати об'єкти з колекції без змінення її.

• - <? extends Тип> (bounded wildcard) дозволяє читати об'єкти з колекції,

але забороняє додавання в неї нових об'єктів.

• - <? super Тип> (lower bounded wildcard) дозволяє додавати об'єкти в

колекцію, але забороняє їх читання.

**Висновок**

У ході виконання даної лабораторної роботи, я отримала важливі навички

параметризованого програмування мовою Java. Ознайомилась з різними

аспектами мови, такими як використання параметрів у методах, створення та

використання класів та інтерфейсів.