Міністерство освіти та науки України

Національний університет «Львівська політехніка»

Кафедра «Електронних обчислювальних машин»



**ЗВІТ**

**з лабораторної роботи № 7**

**з дисципліни:** «Кросплатформенні засоби програмування»

**на тему:** «ПАРАМЕТРИЗОВАНЕ ПРОГРАМУВАННЯ»

Виконав: ст. гр. КІ-35

Флик Є.В.

Прийняв:

доцент кафедри ЕОМ

Іванов Ю.С

Львів-2022

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

**ЗАВДАННЯ**

1. Створити параметризований клас, що реалізує предметну область задану варіантом. Клас має містити мінімум 4 методи опрацювання даних включаючи розміщення та виймання елементів. Парні варіанти реалізують пошук мінімального елементу, непарні – максимального. Написати на мові Java та налагодити програму-драйвер для розробленого класу, яка мстить мінімум 2 різні класи екземпляри яких розмішуються у 9 екземплярі розробленого класу-контейнеру. Програма має розміщуватися в пакеті Група.Прізвище.Lab6 та володіти коментарями, які дозволять автоматично згенерувати документацію до розробленого пакету.

2. Автоматично згенерувати документацію до розробленого пакету.

3. Скласти звіт про виконану роботу з приведенням тексту програми, результату її виконання та фрагменту згенерованої документації.

4. Дати відповідь на контрольні запитання.

**Варіант 22:**

****

**Текст програми:**

|  |
| --- |
| ***Лістинг Thing***  */\*\*  \* lab7 package  \*/* package KI35.Flyk.lab7;  */\*\*  \* Class Thing  \** ***@version*** *1.0  \*/* public abstract class Thing {  protected int weight;  protected String name;  */\*\*  \* Constructor  \** ***@param*** *weight Thing weight  \** ***@param*** *name Thing name  \*/* public Thing(int weight, String name)  {  this.name = name;  this.weight = weight;  }  */\*\*  \* Methods returns Thing weight  \*/* public int getWeight() {  return weight;  }  */\*\*  \* Methods returns Thing name  \*/* public String getName() { return name; }  public void printData()  {  System.*out*.println("Name:" + name);  System.*out*.println("Weight:" + weight);  } }  ***Лістинг Suitcase***  */\*\*  \* lab7 package  \*/* package KI35.Flyk.lab7;  import java.util.ArrayList; */\*\*  \* Class Suitcase  \** ***@version*** *1.0  \*/* public class Suitcase<T extends Thing> {  private ArrayList<T> storage = new ArrayList();  */\*\*  \* Method returns an item by the index  \** ***@param*** *index Element index  \** ***@throws*** *Exception  \*/* public T get(int index)  {  try{  return storage.get(index);  }  catch (Exception e)  {  throw e;  }  }  */\*\*  \* Method adds an item  \** ***@param*** *item Item to be added  \*/* public void add(T item)  {  try {  storage.add(item);  }  catch (Exception e)  {  throw e;  }  }  */\*\*  \* Method shows storage content  \*/* public void showAll()  {  for(int i = 0; i < storage.size(); i++)  {  try{  storage.get(i).printData();  }  catch (Exception e)  {  System.*out*.println(e);  }  }  }  */\*\*  \* Method returns total weight  \*/* public int getTotalWeight()  {  int result = 0;  for(int i = 0; i < storage.size(); i++)  {  result += storage.get(i).getWeight();  }  return result;  }  */\*\*  \* Method returns fruit with max weight  \*/* public T getFruitWithMaxWeight()  {  int maxIdx = 0;   for(int i = 0; i < storage.size(); i++)  {  if(i == 0)  {  continue;  }  if(storage.get(i).getWeight() < storage.get(maxIdx).getWeight())  {  maxIdx = i;  }  }  return storage.get(maxIdx);  }  }  ***Лістинг Clothes***  */\*\*  \* lab7 package  \*/* package KI35.Flyk.lab7;  */\*\*  \* Class Clothes  \** ***@version*** *1.0  \*/* public class Clothes extends Thing {  */\*\*  \* Constructor  \** ***@param*** *weight Thing weight  \** ***@param*** *name Thing name  \*/* public static int *countCl*;  public Clothes(int weight, String name)  {  super(weight, name);  ++*countCl*;  } }  ***Лістинг Laptops***  */\*\*  \* lab7 package  \*/* package KI35.Flyk.lab7;  */\*\*  \* Class Laptops  \** ***@version*** *1.0  \*/* public class Laptops extends Thing {  */\*\*  \* Constructor  \** ***@param*** *weight Thing weight  \** ***@param*** *name Thing name  \*/* public static int *countLap*;  public Laptops(int weight, String name)  {  super(weight, name);   ++*countLap*;  } }  ***Лістинг Towels***  */\*\*  \* lab7 package  \*/* package KI35.Flyk.lab7;  */\*\*  \* Class Towels  \** ***@version*** *1.0  \*/* public class Towels extends Thing {  public static int *countTow*;  */\*\*  \* Constructor  \** ***@param*** *weight Thing weight  \** ***@param*** *name Thing name  \*/* public Towels(int weight, String name)  {  super(weight, name);  ++*countTow*;  } }  ***Лістинг App***  */\*\*  \* lab7 package  \*/* package KI35.Flyk.lab7; */\*\*  \* Class App  \** ***@author*** *Flyk Yevhenii  \** ***@version*** *1.0  \*/* public class App {  */\*\*  \** ***@param*** *args  \*/* public static void main(String[] args)  {  Suitcase suitcase = new Suitcase();   suitcase.add(new Clothes(50, "Pants"));  //suitcase.add(new Laptops(500, "LaptopHP"));  suitcase.add(new Laptops(550, "LaptopLenovo"));  suitcase.add(new Towels(100, "Towel"));  suitcase.add(new Clothes(100, "T-Shirt"));  suitcase.add(new Clothes(75, "Hat"));   System.*out*.println(suitcase.get(0).getName());  System.*out*.println(suitcase.get(1).getName());  System.*out*.println(suitcase.get(2).getName());  System.*out*.println(suitcase.get(3).getName());   System.*out*.println(suitcase.getFruitWithMaxWeight().getName());  System.*out*.println(suitcase.getTotalWeight());  System.*out*.println("\nNumber of Clothes: "+Clothes.*countCl* + "\nNumber of Laptops: "+ Laptops.*countLap*);  }     } |

**Результат роботи :**

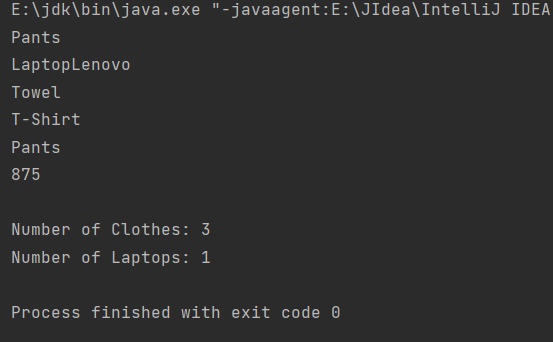
****

Рис.1 Результат роботи програми

**Фрагмент згенерованої документації**

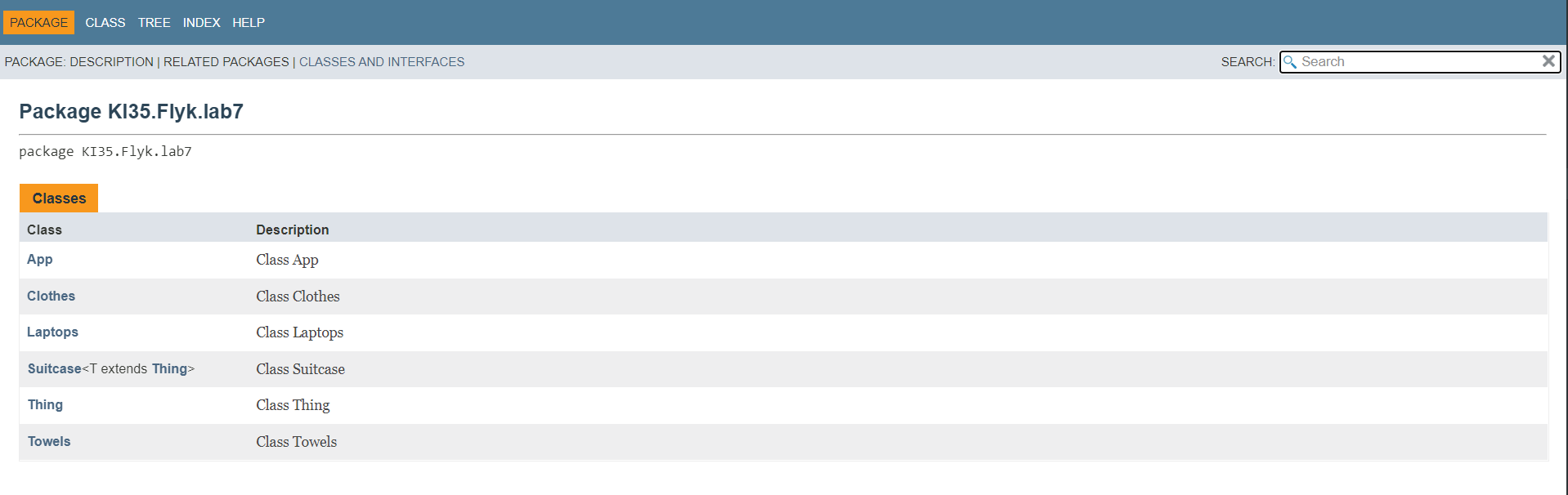


Рис.2.1 Фрагмент згенерованої документації

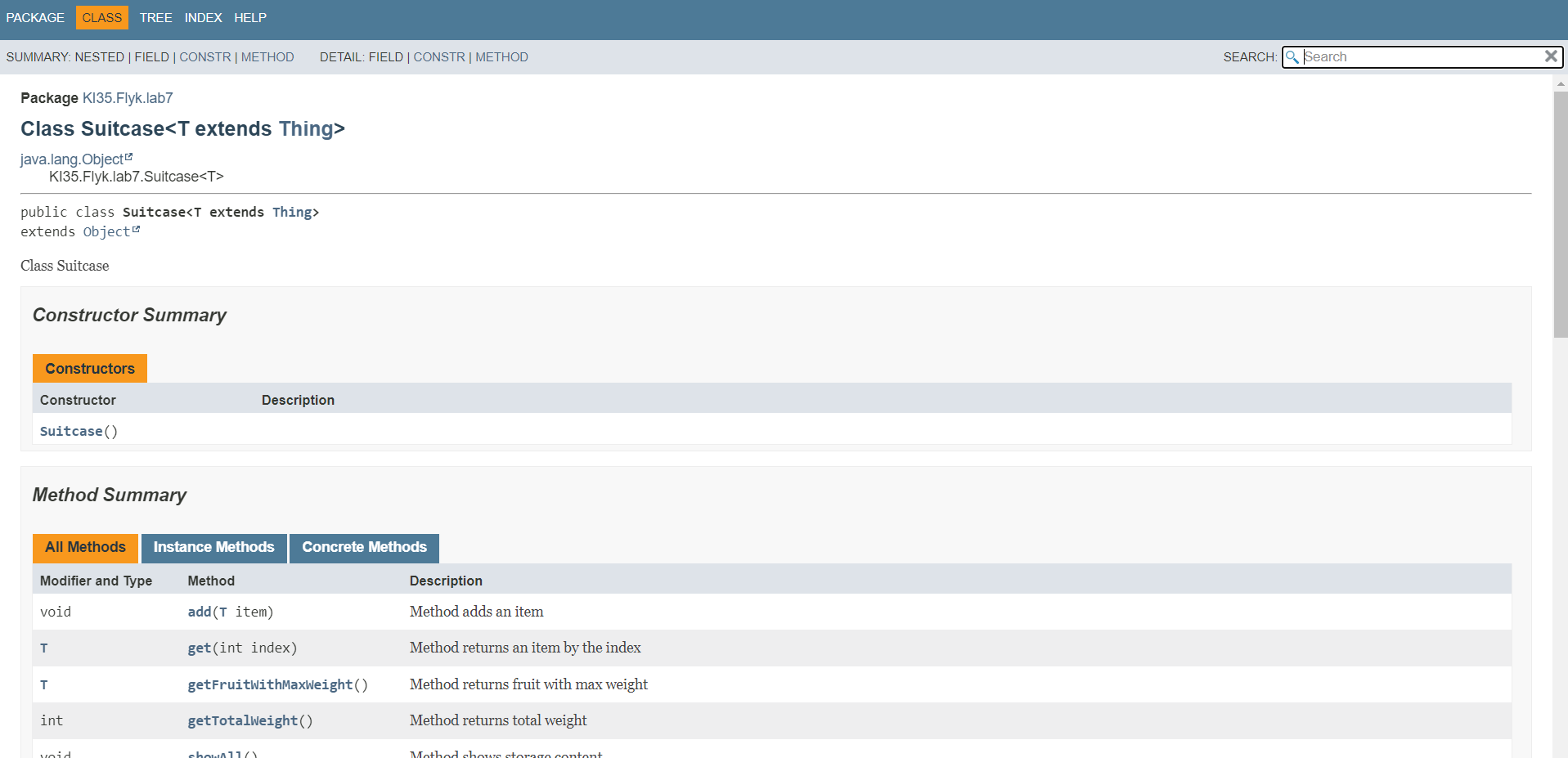


Рис.2.2 Фрагмент згенерованої документації класу Thing

**Відповіді на контрольні запитання:**

1. Дайте визначення терміну «параметризоване програмування».

Параметризоване програмування є аналогом шаблонів у С++. Воно полягає у написанні коду, що можна багаторазово застосовувати з об’єктами різних класів. Користувачів параметризованого програмування можна поділити на 3 рівні кваліфікації:

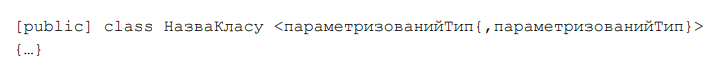
1. ті, що користуються готовими класами;

2. ті, що користуються готовими класами і вміють виправляти помилки, що виникають при їх використанні;

3. ті, що пишуть власні параметризовані класи. Для успішного застосування параметризованого програмування слід навчитися розуміти помилки, що генерує середовище при компіляції програми, що можуть стосуватися, наприклад, неоднозначності визначення спільного суперкласу для всіх переданих об’єктів. З іншої сторони необхідно передбачити захист від присвоєння об’єктів параметризованого класу, що містять об’єкти підкласу об’єктам параметризованого класу, що містять об’єкти суперкласу і дозволити зворотні присвоєння. Для вирішення цієї проблеми у мові Java введено так звані підстановочні типи. Це далеко не всі «підводні камені», що виникають при застосуванні параметризованого програмування.

1. Розкрийте синтаксис визначення простого параметризованого класу.

Параметризований клас – це клас з однією або більше змінними типу. Синтаксис оголошення параметризованого класу:



**Висновок:**

Під час виконання роботи я оволодів навиками параметризованого програмування мовою Java.