Міністерство освіти і 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:**

студент групи КІ-35

Сабадаш Ю.А.

**Прийняв:**

доцент кафедри ЕОМ

Іванов Ю. С.

Львів – 2022

Мета: 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.



Текст програми

Lab7.java

package KI35.Sabadash;  
import static java.lang.System.*out*;  
*/\*\*  
 \* Class <code>BoxDriver</code> implements interaction with class <code>Box</code>  
 \** ***@author*** *Sabadash Yurii KI-35  
 \** ***@version*** *1.0  
 \*/*public class BoxDriver  
{  
 */\*\*  
 \* Main method the entry point to start program  
 \** ***@param*** *args  
 \*/* public static void main(String[] args)  
 {  
 Box<? super Item> bigBox = new Box<Item>();  
  
 Sweet oreoCrumbs = new Sweet("Oreo Crushed Cookie Crumbs (No Creme)", 410);  
  
 bigBox.addItem(new Tools("Calipers", 210, "200мм"));  
 bigBox.addItem(new Tools("Roulette", 100, "7.5 м \* 25 мм"));  
 bigBox.addItem(new Tools("Сhisel", 180, "CR-V 22 мм"));  
 bigBox.addItem(new Book("English Grammar in Use Intermediate", 100, " Antoine de Saint-Exupery", "Art" ));  
 bigBox.addItem(new Sweet("Oreo Small Crushed Cookie", 400));  
 bigBox.addItem(new Sweet("Toblerone White Chocolate Large Bar", 360));  
 bigBox.addItem(oreoCrumbs);  
 *out*.println("-----------------------");  
  
 var min = bigBox.findMin();  
 *out*.println("The element with smallest weight is : ");  
 min.printInfo();  
  
 *out*.println("\n====Printing=all=item=in=the==box====\n");  
 bigBox.printAllArray();  
  
 bigBox.deleteDataByObject(oreoCrumbs);  
 *out*.println("\n====Printing=all=item=in=the==box====\n");  
 bigBox.printAllArray();  
  
  
 bigBox.deleteDataByIndex(1);  
 *out*.println("\n====Printing=all=item=in=the==box====\n");  
 bigBox.printAllArray();  
  
 }  
}

Book.java

package KI35.Sabadash;  
  
import static java.lang.System.*out*;  
  
*/\*\*  
 \* Class <code>Book</code> implements book  
 \*  
 \** ***@author*** *Sabadash Yurii KI-35  
 \** ***@version*** *1.0  
 \*/*class Book implements Item {  
 private String bookName;  
 private String genre;  
 private String author;  
 private int bookWeight;  
  
 public String getBookName() {  
 return bookName;  
 }  
  
 public void setBookName(String bookName) {  
 this.bookName = bookName;  
 }  
  
 public String getGenre() {  
 return genre;  
 }  
  
 public void setGenre(String genre) {  
 this.genre = genre;  
 }  
  
 public String getAuthor() {  
 return author;  
 }  
  
 public void setAuthor(String author) {  
 this.author = author;  
 }  
  
 public int getBookWeight() {  
 return bookWeight;  
 }  
  
 public void setBookWeight(int bookWeight) {  
 this.bookWeight = bookWeight;  
 }  
  
 */\*\*  
 \* Сonstrucnor for initialize item  
 \*  
 \** ***@param*** *bookName The book name  
 \** ***@param*** *bookWeight The book weight  
 \** ***@param*** *author The book author  
 \** ***@param*** *genre The book genre  
 \*/* Book(String bookName, int bookWeight, String author, String genre) {  
 this.bookName = bookName;  
 this.bookWeight = bookWeight;  
 this.author = author;  
 this.genre = genre;  
 }  
  
 @Override  
 public int getWeight() {  
 return bookWeight;  
 }  
  
 @Override  
 public void printInfo() {  
 *out*.println  
 ("Name: " + bookName +  
 "\nAuthor: " + author +  
 "\nGenre: " + genre +  
 "\nWeigh: " + bookWeight  
 );  
 }  
  
 @Override  
 public int compareTo(Item item) {  
 Integer cmp = bookWeight;  
 return cmp.compareTo(item.getWeight());  
 }  
}

Box.java

package KI35.Sabadash;  
  
import java.util.ArrayList;  
import java.util.Objects;  
  
import static java.lang.System.*out*;  
  
*/\*\*  
 \* Class <code>Box</code> implements Box of some items  
 \*  
 \** ***@param*** <*T*> *The Type that implements interface Item  
 \** ***@author*** *Sabadash Yurii KI-35  
 \** ***@version*** *1.0  
 \*/*public class Box<T extends Item> {  
 private ArrayList<T> arr;  
  
 */\*\*  
 \* Constructor default  
 \*/* public Box() {  
 arr = new ArrayList<>();  
 }  
  
 */\*\*  
 \* Method to find items with minimal weight  
 \*  
 \** ***@return*** *Object with minimal elements  
 \*/* public T findMin() {  
 if (!arr.isEmpty()) {  
 T min = arr.get(0);  
 for (var i : arr) {  
 if (i.compareTo(min) < 0)  
 min = i;  
 }  
 return min;  
 }  
 return null;  
 }  
  
 */\*\*  
 \* Method to add item to box  
 \*  
 \** ***@param*** *item Item that will be added  
 \*/* public void addItem(T item) {  
 *out*.println("-----------------------");  
 arr.add(item);  
 *out*.println("Item added # " + arr.size() + " :");  
 item.printInfo();  
  
 }  
  
 */\*\*  
 \* Method to pop item from box by object  
 \*  
 \** ***@param*** *item Item that will be deleted  
 \*/* public void deleteDataByObject(T item) {  
 if (arr.removeIf(arr -> Objects.*equals*(item, arr))) {  
 *out*.println("The following item deleted: ");  
 item.printInfo();  
 } else {  
 *out*.println("Item cannot be deleted: ");  
 item.printInfo();  
 }  
  
 }  
  
 */\*\*  
 \* Method to pop item from box by index  
 \*  
 \** ***@param*** *index Index of item that will be deleted  
 \*/* public void deleteDataByIndex(int index) {  
 if (index > 0 && index <= arr.size()) {  
 *out*.println("The following item deleted ");  
 arr.remove(index - 1).printInfo();  
 } else {  
 *out*.println("Item cannot be deleted: ");  
 arr.get(index).printInfo();  
 }  
 }  
  
 */\*\*  
 \* Method to view all item in the box  
 \*/* public void printAllArray() {  
 int cnt = 0;  
 for (var a : arr) {  
 if (!arr.isEmpty()) {  
 *out*.println("Item # " + (++cnt));  
 *out*.println("------------------------");  
 a.printInfo();  
 *out*.println("------------------------");  
 } else {  
 *out*.println("Box is empty");  
 }  
 }  
 }  
}

Sweet.java

import static java.lang.System.*out*;  
  
*/\*\*  
 \* Class <code> Sweets</code> implements sweets  
 \*  
 \** ***@author*** *Sabadash Yurii KI-35  
 \** ***@version*** *1.0  
 \*/*public class Sweet implements Item {  
 String nameSweets;  
  
 int weightSweet;  
  
 */\*\*  
 \* Constructor to initialize Sweets  
 \*  
 \** ***@param*** *nameSweets The sweet name  
 \** ***@param*** *weightSweet The sweet weight  
 \*/* public Sweet(String nameSweets, int weightSweet) {  
 this.nameSweets = nameSweets;  
 this.weightSweet = weightSweet;  
 }  
  
 @Override  
 public int getWeight() {  
 return weightSweet;  
 }  
  
 @Override  
 public void printInfo() {  
 *out*.println  
 ("Name: " + nameSweets +  
 "\nWeigh: " + weightSweet  
 );  
 }  
  
 @Override  
 public int compareTo(Item item) {  
 Integer cmp = weightSweet;  
 return cmp.compareTo(item.getWeight());  
 }  
}

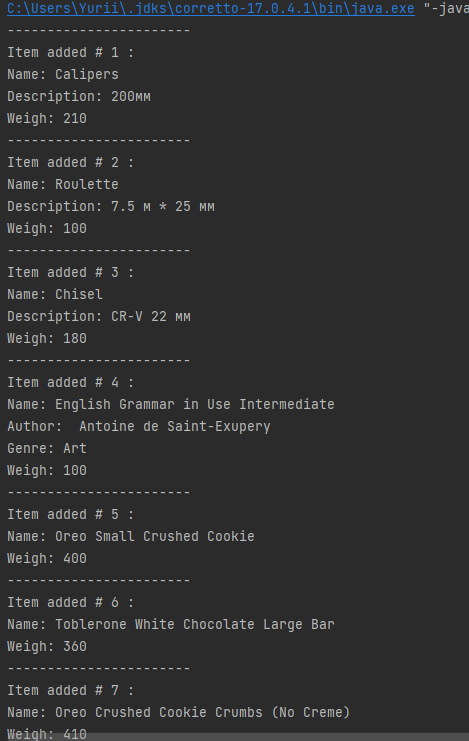
Tools.java

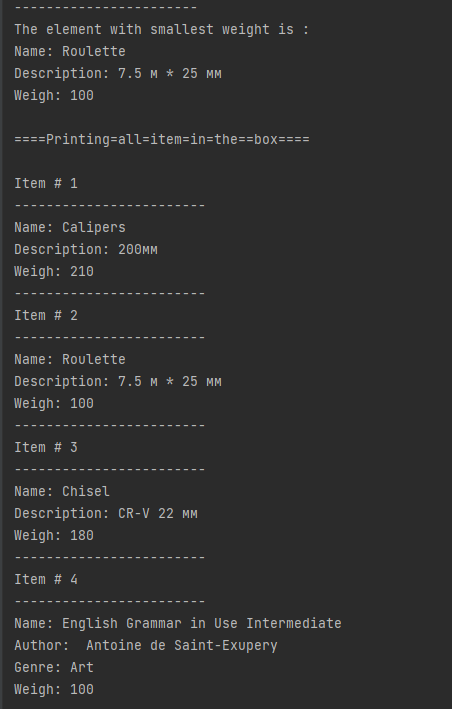
*\** ***@author*** *Sabadash Yurii KI-35  
 \** ***@version*** *1.0  
 \*/*public class Tools implements Item {  
 String nameTools;  
  
 String description;  
 int weightTools;  
  
 */\*\*  
 \* Constructor to initialize Tools  
 \*  
 \** ***@param*** *nameTools The tools name  
 \** ***@param*** *weightTools The weight tools  
 \** ***@param*** *description The tools description  
 \*/* public Tools(String nameTools, int weightTools, String description) {  
 this.nameTools = nameTools;  
 this.weightTools = weightTools;  
 this.description = description;  
 }  
  
 @Override  
 public int getWeight() {  
 return weightTools;  
 }  
  
 @Override  
 public void printInfo() {  
 *out*.println  
 (  
 "Name: " + nameTools +  
 "\nDescription: " + description +  
 "\nWeigh: " + weightTools  
 );  
 }  
  
 @Override  
 public int compareTo(Item item) {  
 Integer cmp = weightTools;  
 return cmp.compareTo(item.getWeight());  
 }  
}

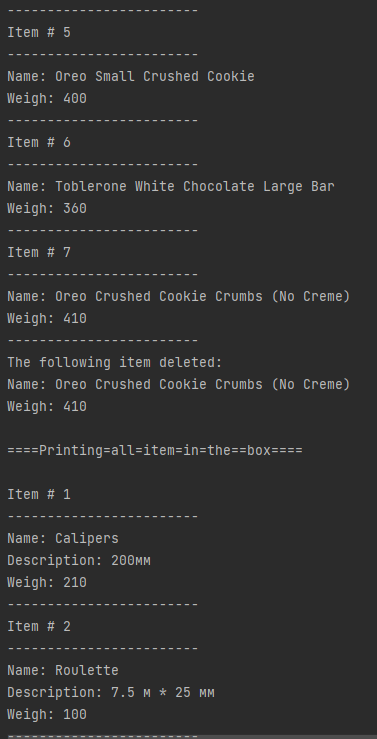
Item.java

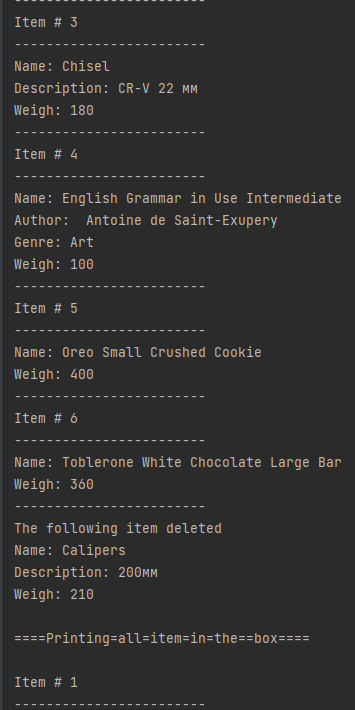
package KI35.Sabadash;  
  
*/\*\*  
 \* Interface<code>Item</code> that contain method for all items  
 \*  
 \** ***@author*** *Sabadash Yurii KI-35  
 \** ***@version*** *1.0  
 \*/*public interface Item extends Comparable<Item> {  
 */\*\*  
 \* Method to get weight of item  
 \*  
 \** ***@return*** *The weight of item  
 \*/* public int getWeight();  
  
 */\*\*  
 \* Method to get info of item  
 \*/* public void printInfo();  
}

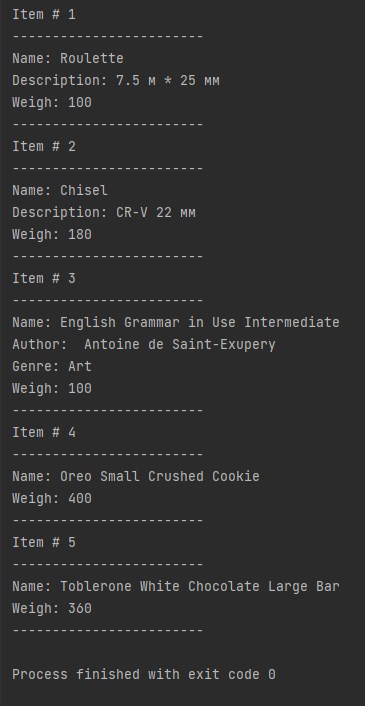
Результат роботи програми



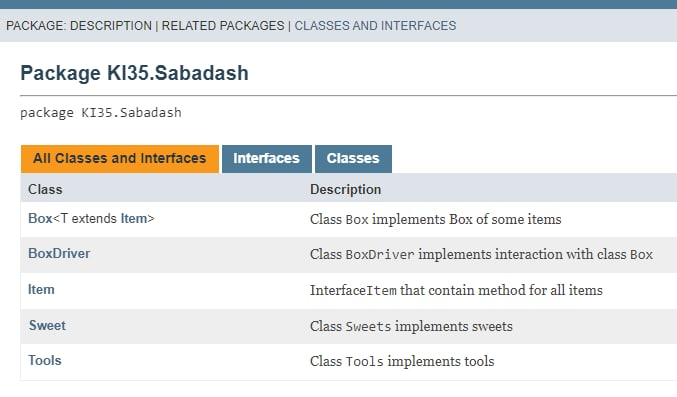








**Фрагмент згенерованої документації**



Контрольні питання

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

1. Розкрийте синтаксис визначення простого параметризованого класу.

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

[public] class НазваКласу {…}

Висновок: на даній лабораторній роботі я ознайомився з поняттям параметризованого програмування. Дізнався його основні функції та сфери застосування. Практично оволодів навичками цїєї сфери. Написав та протестував програму з використанням параметризованого програмування.