**Міністерство освіти і науки України**

**Національний університет «Львівська політехніка»**

**Кафедра ЕОМ**

Звіт



**Лабораторна робота № 6**

“ ПАРАМЕТРИЗОВАНЕ ПРОГРАМУВАННЯ”

з курсу “Кросплатформні засоби програмування”

Варіант: 4

Виконав:

ст.гр.КІ-205

Воробець Тетяна

Прийняв:

доцент кафедри ЕОМ

Олексів М.В.

**Львів 2024**

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

**Теоретичні відомості:** 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.

Параметризований клас – це клас з однією або більше змінними типу. Параметризовані методи визначаються в середині як звичайних класів так і параметризованих. На відміну від звичайних методів параметризовані методи мають параметризований тип, що дозволяє за їх допомогою опрацьовувати різнотипні набори даних. Реальні типи для методів, як і для класів, визначаються у місці виклику методу шляхом передачі реального типу у трикутних дужках.

Правила спадкування параметризованих типів:

1. Всі класи, що утворені з одного і того ж параметризованого класу з використанням різних значень змінних типів є незалежними навіть якщо між цими типами є залежність спадкування.

2. Завжди можна перетворити параметризований клас у «сирий» клас, при роботі з яким захист від некоректного коду є значно слабшим, що дозволяє здійснювати небезпечні присвоєння об’єктів параметризованого класу об’єктам «сирого» класу.

3. Параметризовані класи можуть розширювати або реалізовувати інші параметризовані класи.

Обмеження підтипу – дозволяє позначити будь-який параметризований тип, чий параметр типу є типом або підтипом вказаного у параметрі типу, що дозволяє одержувати результати роботи методів параметризованого типу, але не передавати параметри методам, що приймають параметри параметризованого типу. Це відбувається тому, що компілятор не здатний вивести з "?" конкретний тип параметру, але гарантує, що цей тип буде типом або підтипом вказаного у параметрі типу. Обмеження супертипу – дозволяє позначити будь-який параметризований тип, чий параметр типу є класом або суперкласом вказаного у параметрі типу, що дозволяє передавати параметри методам, що приймають параметри параметризованого типу. Це відбувається тому, що компілятор хоч і не здатний вивести з "?" конкретний тип параметру, що передається у метод, але він може безпечно привести передане значення до будь-якого з супертипів. При одержанні результатів роботи методів параметризованого типу нема ніякої гарантії стосовно типу результату, тому результат роботи можна присвоїти лише типу Object.

**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. Завантажити код на GitHub згідно методичних вказівок по роботі з GitHub.

4. Скласти звіт про виконану роботу з приведенням тексту програми, результату її виконання та фрагменту згенерованої документації та завантажити його у ВНС.

5. Дати відповідь на контрольні запитання.

**Код програми:**

Item.java

package KI305.Vorobets.Lab6;

/\*\*

\* Interface Item implements item

\*/

public interface Item {

public int getSize();

public String info();

}

Box.java

package KI305.Vorobets.Lab6;

/\*\*

\* Box implements a box

\*/

public class Box implements Item {

private int weight;

/\*\*

\* Constructor

\* **@param** <code>weight</code> weight

\*/

public Box(int weight) {

this.weight = weight;

}

/\*\*

\* Method returns the info about object

\*/

*@Override*

public String info() {

return "Box: weight=" + weight;

}

/\*\*

\* Method returns the weight box

\*/

*@Override*

public int getSize() {

return weight;

}

}

Parts.java

package KI305.Vorobets.Lab6;

/\*\*

\* Parts implements a part

\*/

public class Parts implements Item {

private String object;

private int size;

/\*\*

\* Constructor

\* **@param** <code>object</code> object

\* **@param** <code>size</code> size

\*/

public Parts(String object, int size) {

this.object = object;

this.size = size;

}

/\*\*

\* Method returns the size parts

\*/

*@Override*

public int getSize() {

return size;

}

/\*\*

\* Method returns the info about object

\*/

*@Override*

public String info() {

return "Parts: object='" + object + "', price=" + size;

}

}

Conveyor.java

/\*\*

\*

\*/

package KI305.Vorobets.Lab6;

import java.util.ArrayList;

/\*\*

\* Conveyor is main class

\*/

public class Conveyor<T extends Item> {

private ArrayList<T> items;

/\*\*

\* Constructor

\*/

public Conveyor() {

items = new ArrayList<>();

}

/\*\*

\* Method for add item

\* **@param** <code>item</code> item.

\*/

public void addItem(T item) {

items.add(item);

}

/\*\*

\* Method for add item

\*/

public T removeItem() {

if (!items.isEmpty()) {

return items.remove(0);

}

return null;

}

/\*\*

\* Method for check a item

\*/

public boolean hasItems() {

return !items.isEmpty();

}

/\*\*

\* Method for find a minimum object.

\*/

public T findMin() {

if (items.isEmpty()) {

return null;

}

T min = items.get(0);

for (T item : items) {

if (item.getSize() < min.getSize()) {

min = item;

}

}

return min;

}

/\*\*

\* Method for count object

\*/

public int getItemCount() {

return items.size();

}

}

ConveyorApp.java

package KI305.Vorobets.Lab6;

/\*\*

\* ConveyorApp is main class

\*/

public class ConveyorApp {

public static void main(String[] args) {

Conveyor<Box> boxConveyor = new Conveyor<>();

boxConveyor.addItem(new Box(10));

boxConveyor.addItem(new Box(20));

boxConveyor.addItem(new Box(5));

Conveyor<Parts> partsConveyor = new Conveyor<>();

partsConveyor.addItem(new Parts("Car", 10));

partsConveyor.addItem(new Parts("Computer", 8));

partsConveyor.addItem(new Parts("Monitor", 3));

Conveyor<? super Item> super\_conveyor = new Conveyor<>();

super\_conveyor.addItem(new Box(7));

super\_conveyor.addItem(new Parts("Mouse", 6));

Item itBox = boxConveyor.findMin();

System.***out***.println(itBox.info());

Item itPart = partsConveyor.findMin();

System.***out***.println(itPart.info());

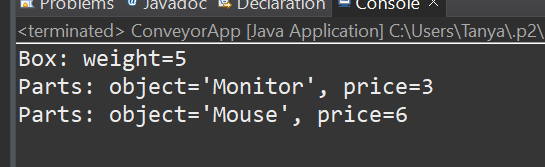
Item itSuper = super\_conveyor.findMin();

System.***out***.println(itSuper.info());

}

}

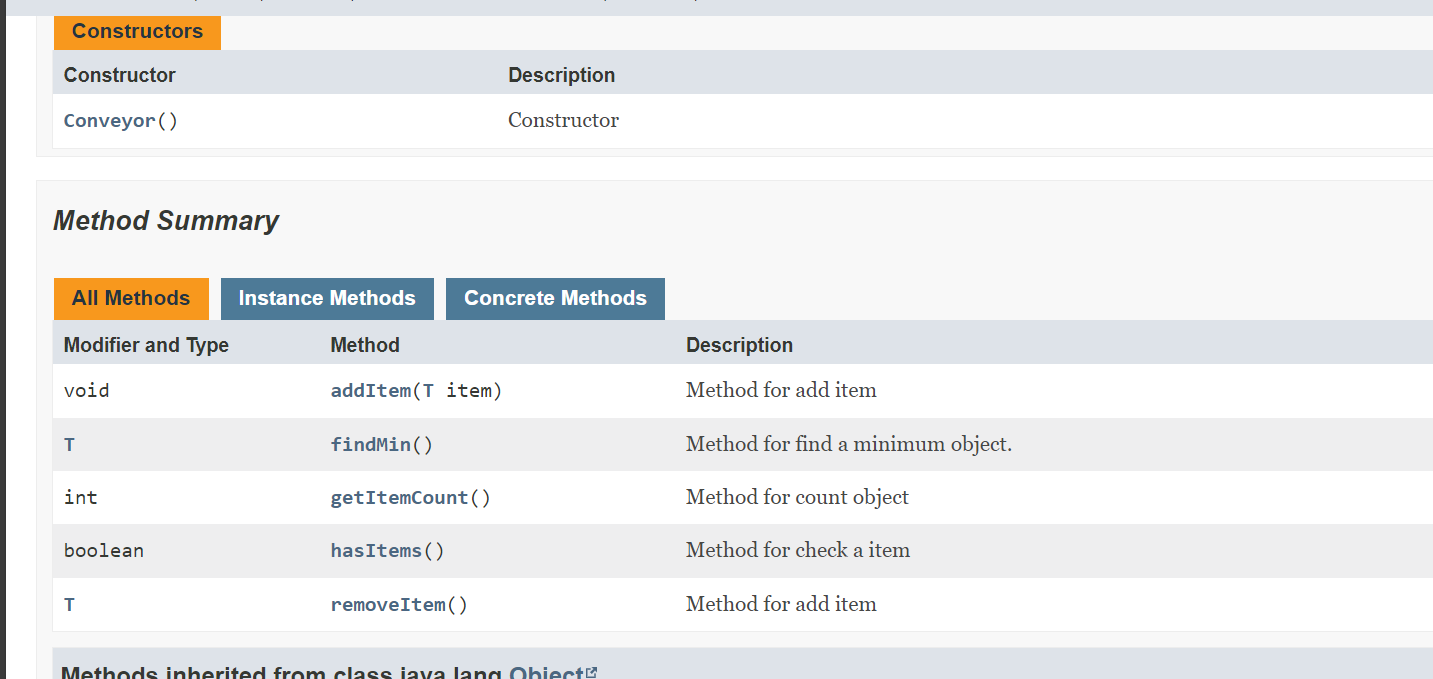
**Результат:**

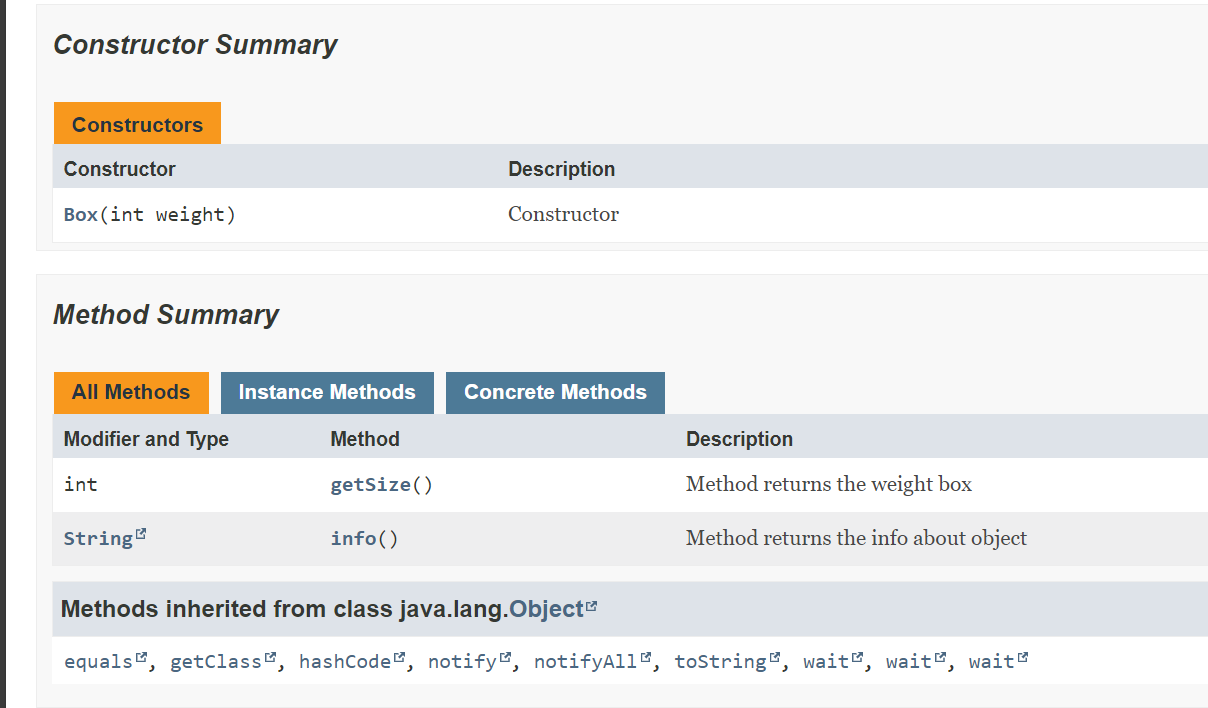
****

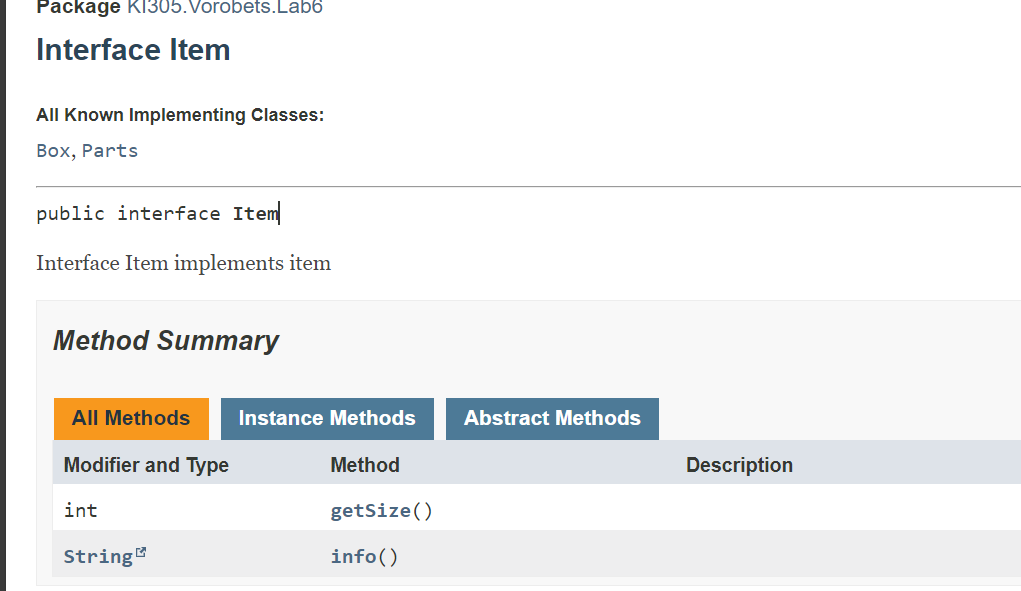
**Посилання на репозиторій:**

<https://github.com/NikaDe7/CPPT_Vorobets_TI_KI-35_1.git>

**Документація:**







**Висновок:** оволоділа навиками параметризованого програмування мовою Java.