

Міністерство освіти і науки України

Національний університет “Львівська політехніка”

Кафедра ЕОМ



Звіт

З лабораторної роботи №6

Варіант – 10

З дисципліни: «Кросплатформні засоби програмування»

На тему: «Параметризоване програмування»

Виконав: ст. гр. КІ-306

Миценко О. С.

Прийняв:

доцент кафедри ЕОМ

Іванов Ю. С.

Львів 2023

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 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.

Варіант завдання: Пенал

Код програми:

Файл Item.java

```
package KI306.Mytsenko.Lab6;
/**
 * The Item interface represents items that can be used and have price.
 * Classes that implement this interface should provide implementations
 * for the methods declared here.
 *
 * @author Oleksandr Mytsenko
 * @version 1.0
 */
public interface Item extends Comparable<Item> {
    /**
     * Get the price of the item.
     *
     * @return The price of the item as a double value.
     */
    double getPrice();
    /**
     * Perform an action with the item.
     */
    void use();
    /**
     * Print information about the item.
     */
    void print();
}
```

Файл Penal.java

```
package KI306.Mytsenko.Lab6;

import java.util.ArrayList;
/**
 * The Penal class represents a container for items of a specific type that implement the
 * Item interface.
 * It allows adding, retrieving, finding the maximum-price item, and using items.
 *
 * @param <T> The type of items that can be stored in the penal, which must implement the
 * Item interface.
 *
 * @author Oleksandr Mytsenko
 * @version 1.0
 */
public class Penal<T extends Item> {
    private ArrayList<T> array;
    /**
     * Constructs a new Penal object.
     * Initializes an empty array to store items.
     */
    public Penal() {
        array = new ArrayList<T>();
    }
    /**
     * Adds an item to the penal and prints a message indicating the addition.
     *
     * @param item The item to add to the penal.
     */
    public void putItem(T item) {
        array.add(item);
        System.out.print("Item added: ");
        item.print();
    }
    /**
     * Retrieves an item from the penal by its index.
     *
     * @param i The index of the item to retrieve.
     * @return The item at the specified index, or null if the index is out of bounds.
     */
    public T getItem(int i) {
        return array.get(i);
    }
    /**
     * Finds and returns the item with the maximum price in the penal.
     *
     * @return The item with the maximum price, or null if the penal is empty.
     */
    public T getMax() {
        if (!array.isEmpty()) {
            T max = array.get(0);
            for (int i = 1; i < array.size(); i++) {
                if (array.get(i).compareTo(max) > 0) {
                    max = array.get(i);
                }
            }
            return max;
        }
        return null;
    }
    /**

```

```

    * Uses an item in the penal by its index.
    *
    * @param i The index of the item to use.
    */
    public void useItem(int i) {
        array.get(i).use();
    }
}

```

Файл Rubber.java

```

package KI306.Mytsenko.Lab6;
/**
 * The Scissors class represents rubber item that implements the Item interface.
 * It has a name and a price , and it can be used for reading.
 *
 * @author Oleksandr Mytsenko
 * @version 1.0
 */
public class Rubber implements Item {
    private double price;
    private String name;

    /**
     * Constructs a new Rubber object with the specified name and price.
     *
     * @param name The name of the rubber.
     * @param price The price of the rubber as a double value.
     */
    public Rubber(String name, double price) {
        this.name = name;
        this.price = price;
    }

    /**
     * Get the price of the rubber.
     *
     * @return The price of the rubber as a double value.
     */

    @Override
    public double getPrice() {
        return price;
    }

    /**
     * Use the rubber.
     */
    @Override
    public void use() {
        System.out.println("Using rubber " + name);
    }

    /**
     * Print information about the rubber.
     */
    @Override
    public void print() {
        System.out.println("Rubber: " + name + ", price: " + price);
    }

    /**
     * Compares this rubber's price to the price of another item that implements the Item
     interface.
     *
     * @param item The item to compare to.
     */
}

```

```

    * @return A negative integer if this rubber is less expensive, a positive integer if
    *         it's more positive, or 0 if they have the same price.
    */
    @Override
    public int compareTo(Item item) {
        Double p = price;
        return p.compareTo(item.getPrice());
    }

}

```

Файл Scissors.java

```

package KI306.Mytsenko.Lab6;
/**
 * The Scissors class represents scissors item that implements the Item interface.
 * It has a name and a price , and it can be used for reading.
 *
 * @author Oleksandr Mytsenko
 * @version 1.0
 */
public class Scissors implements Item {
    private double price;
    private String name;
    /**
     * Constructs a new Scissors object with the specified name and price.
     *
     * @param name The name of the scissors.
     * @param price The price of the scissors as a double value.
     */
    public Scissors(String name, double price) {
        this.name = name;
        this.price = price;
    }
    /**
     * Get the price of the scissors.
     *
     * @return The price of the scissors as a double value.
     */
    @Override
    public double getPrice() {
        return price;
    }
    /**
     * Use the scissors.
     */
    @Override
    public void use() {
        System.out.println("Using scissors " + name);
    }
    /**
     * Print information about the scissors.
     */
    @Override
    public void print() {
        System.out.println("Scissors: " + name + ", price: " + price);
    }
    /**
     * Compares this scissors' price to the price of another item that implements the
     * Item interface.

```

```

    *
    * @param item The item to compare to.
    * @return A negative integer if this scissors is less expensive, a positive integer
if
    *         it's more positive, or 0 if they have the same price.
    */
@Override
public int compareTo(Item item) {
    Double p = price;
    return p.compareTo(item.getPrice());
}
}

```

Файл Main.java

```

package KI306.Mytsenko.Lab6;
/**
 * Class EquationsApp Implements driver for Equations class
 *
 * @author Oleksandr Mytsenko
 * @version 1.0
 */

public class Main {
    /**
     * The main method is the entry point of the program.
     *
     * @param args The command-line arguments (not used in this program).
     */
    public static void main(String[] args) {
        Penal <? super Item> penal = new Penal<>();
        penal.putItem(new Rubber("Rubber1", 10.50));
        penal.putItem(new Scissors("Scissors1" , 55.75));
        penal.putItem(new Rubber("Rubber2" , 15.25));
        penal.putItem(new Scissors("Scissors2" , 100));

        System.out.println(penal.getMax().getPrice());

        // Item item = penal.getItem(2);
        //item.print();

        //item = penal.getItem(3);
        //item.use();

        //Item max = penal.getMax();
        // System.out.println("\nThe the most expensive item in penal is: ");
        //max.print();
    }
}

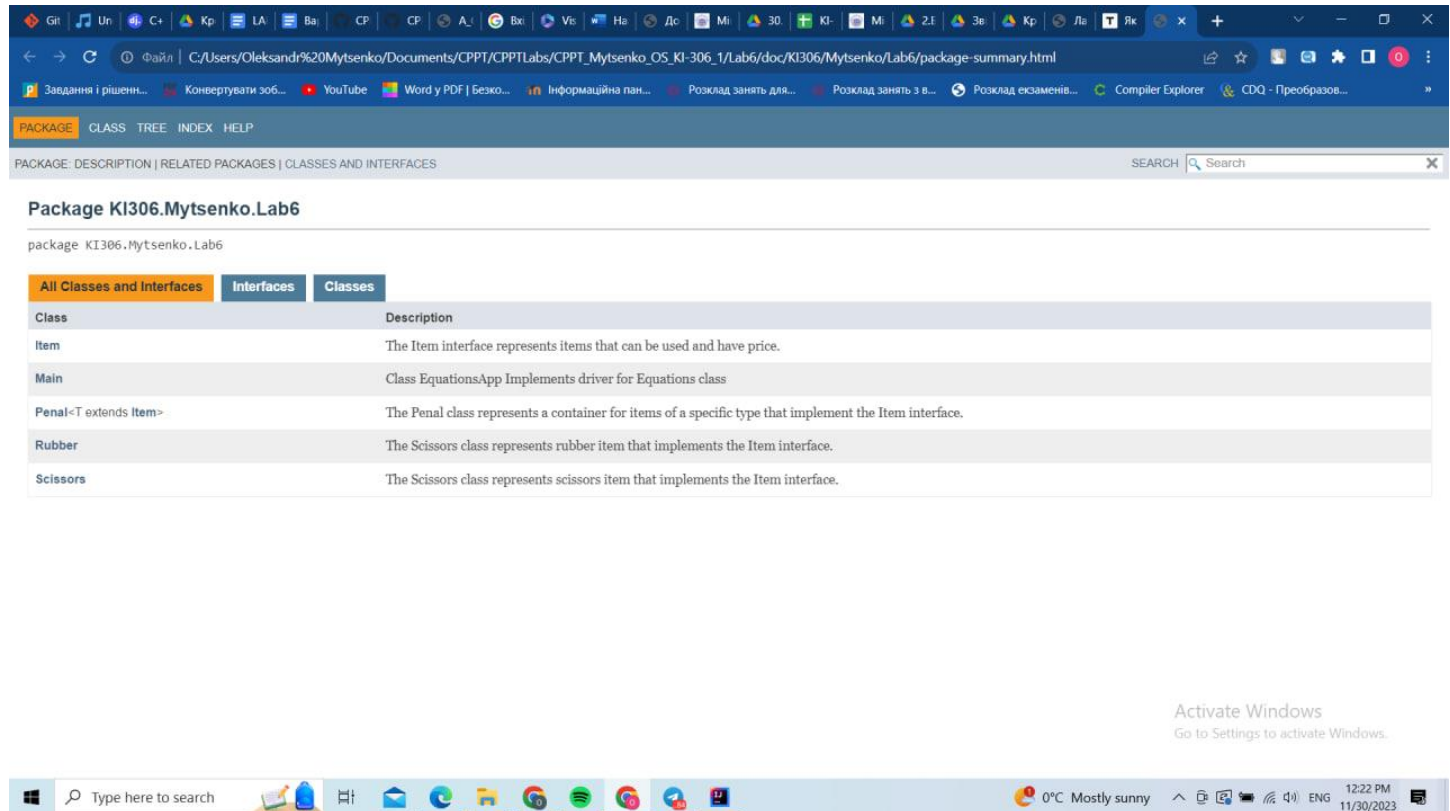
```

Результати роботи програми:

```
"C:\Users\Oleksandr Mytsenko\.jdk\openjdk-20.0.2\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition
Item added: Rubber: Rubber1, price: 10.5
Item added: Scissors: Scissors1, price: 55.75
Item added: Rubber: Rubber2, price: 15.25
Item added: Scissors: Scissors2, price: 100.0
100.0

Process finished with exit code 0
```

Фрагмент згенерованої документації:



Package KI306.Mytsenko.Lab6

package KI306.Mytsenko.Lab6

Class	Description
Item	The Item interface represents items that can be used and have price.
Main	Class EquationsApp Implements driver for Equations class
Penal<T extends Item>	The Penal class represents a container for items of a specific type that implement the Item interface.
Rubber	The Scissors class represents rubber item that implements the Item interface.
Scissors	The Scissors class represents scissors item that implements the Item interface.

Activate Windows
Go to Settings to activate Windows.

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 { // Тіло класу }`
3. Розкрийте синтаксис створення об'єкту параметризованого класу. - `НазваКласу зміннаКласу = new НазваКласу(параметри);`
4. Розкрийте синтаксис визначення параметризованого методу. - `public типПовернення назваМетоду(параметри) { // Тіло методу }`
5. Розкрийте синтаксис виклику параметризованого методу. - `(НазваКласу|НазваОб'єкту).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.
 - Всі класи, створені з параметризованого класу, незалежні один від одного.
 - 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 підстановочних типів. - `(unbounded wildcard)` дозволяє читати об'єкти з колекції без змінення її.
 - - `(bounded wildcard)` дозволяє читати об'єкти з колекції, але забороняє додавання в неї нових об'єктів.
 - - `(lower bounded wildcard)` дозволяє додавати об'єкти в колекцію, але забороняє їх читання.

Висновок

У ході виконання даної лабораторної роботи, я отримав важливі навички параметризованого програмування мовою Java. Ознайомився з різними аспектами мови, такими як використання параметрів у методах, створення та використання класів та інтерфейсів.