Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра «Електронних обчислювальних машин»

Звіт

з лабораторної роботи № 6

з дисципліни: «Кросплатформенні засоби програмування»

на тему: «Параметризоване програмування»

Виконав:

студент групи КІ-306

Ширий Б. І.

Прийняв:

доцент кафедри ЕОМ

Іванов Ю. С.

Львів – 2022

МЕТОДИЧНІ ВІДОМОСТІ РОБОТИ

МЕТА

Оволодіти навиками параметризованого програмування мовою Java.

ЗАВДАННЯ

№1

Створити параметризований клас, що реалізує предметну область задану варіантом, а саме «Банка». 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

Для розробленої програми згенерувати документацію

№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.

ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ

ВИХІДНИЙ КОД

Написав програму, яка організовує роботу банки, де основний код класу навів у лістингу 2.1, а у лістингу 2.2 клас-драйвер.

Лістинг 2.1. Код основної програми.

```
/**
 * A class representing a "jar" that can hold objects of type T.
 * Objects in the jar can be ordered using comparison, implemented through the
Comparable<T> interface.
 *
 * @param <T> The type of objects that can be stored in the jar and compared.
 */
public class Jar<T extends Comparable<T>> extends JPanel {
    private List<T> items;

    /**
     * Default constructor. Creates an empty jar.
     */
    public Jar() {
        items = new ArrayList<>();
    }

    /**
     * Constructor that initializes the jar with a list of items.
     *
     * @param items The list of items for the initial filling of the jar.
     */
    public Jar(List<T> items) {
        this.items = new ArrayList<>(items);
    }

    /**
     * Copy constructor. Creates a copy of the jar based on another jar.
     *
     * @param otherJar Another jar object to be copied.
     */
    public Jar(Jar<T> otherJar) {
        this.items = new ArrayList<>(otherJar.items);
    }

    /**
     * Adds an object to the jar.
     *
     * @param item The object to add to the jar.
     */
    public void addItem(T item) {
        items.add(item);
    }

    /**
     * Removes an object from the jar at a specified index.
     *
     * @param index The index of the object to remove.
     * @return The removed object or null if the index is invalid.
     */
    public T removeItem(int index) {
        if (index >= 0 && index < items.size()) {
            return items.remove(index);
        } else {
            return null;
        }
    }
```

```
    /**
     * Randomly shuffles the objects in the jar.
     */
    public void shuffle() {
        Collections.shuffle(items);
    }

    /**
     * Finds the minimum object in the jar.
     *
     * @return The minimum object or null if the jar is empty.
     */
    public T findMinItem() {
        if (items.isEmpty()) {
            return null;
        }
        return Collections.min(items);
    }

    /**
     * Finds the maximum object in the jar.
     *
     * @return The maximum object or null if the jar is empty.
     */
    public T findMaxItem() {
        if (items.isEmpty()) {
            return null;
        }
        return Collections.max(items);
    }

    /**
     * Returns a list of objects in the jar (unmodifiable).
     *
     * @return An unmodifiable list of objects in the jar.
     */
    public List<T> getItems() {
        return Collections.unmodifiableList(items);
    }
}
```

Лістинг 2.2. Клас-драйвер.

```
package CE_306.Shyryi.Lab6;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.ArrayList;
import java.util.List;

/**
 * A driver class that demonstrates the usage of the Jar class for storing and
manipulating
 * items of different types (Integer and String).
 */
public class JarDriver {
    public static void main(String[] args) {
        JFrame frame = new JFrame("Item Jar");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(400, 400);

        Jar<Integer> intJar = new Jar<>();
        Jar<String> strJar = new Jar<>();

        JLabel itemsLabel = new JLabel();
```

```
        JButton addButton = new JButton("Add Item");
        addButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                String input = JOptionPane.showInputDialog("Enter an item:");
                if (input != null && !input.isEmpty()) {
                    intJar.addItem(Integer.parseInt(input));
                    strJar.addItem(input);
                }
            }
        });
        JButton removeButton = new JButton("Remove Item");
        removeButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                int index = Integer.parseInt(JOptionPane
                        .showInputDialog("Enter the index to remove:"));
                intJar.removeItem(index);
                strJar.removeItem(index);
            }
        });
        JButton shuffleButton = new JButton("Shuffle");
        shuffleButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                intJar.shuffle();
                strJar.shuffle();
            }
        });
        JButton findMinButton = new JButton("Find Min");
        findMinButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                JOptionPane.showMessageDialog(null,
                        "Min Integer: " + intJar.findMinItem() +
                        "\nMin String: " + strJar.findMinItem());
            }
        });
        JButton findMaxButton = new JButton("Find Max");
        findMaxButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                JOptionPane.showMessageDialog(null,
                        "Max Integer: " + intJar.findMaxItem() +
                        "\nMax String: " + strJar.findMaxItem());
            }
        });
        JButton showItemsButton = new JButton("Show Items");
        showItemsButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                itemsLabel.setText(
                "<html>"
                + "<p>" + "Integer Items: " + intJar.getItems().toString() + "</p>"
                + "<p>" + "String Items: " + strJar.getItems().toString() + "</p>"
                + "</html>");
            }
        });

        JPanel panel = new JPanel();
        panel.setLayout(new GridLayout(7, 1));
        panel.add(addButton);
        panel.add(removeButton);
        panel.add(shuffleButton);
        panel.add(findMinButton);
        panel.add(findMaxButton);
        panel.add(showItemsButton);
        panel.add(itemsLabel);

        frame.add(panel);
        frame.setVisible(true);
    }
}
```

РЕЗУЛЬТАТИ ВИКОНАННЯ

Початковий вигляд програми наведено на рисунку 2.1.

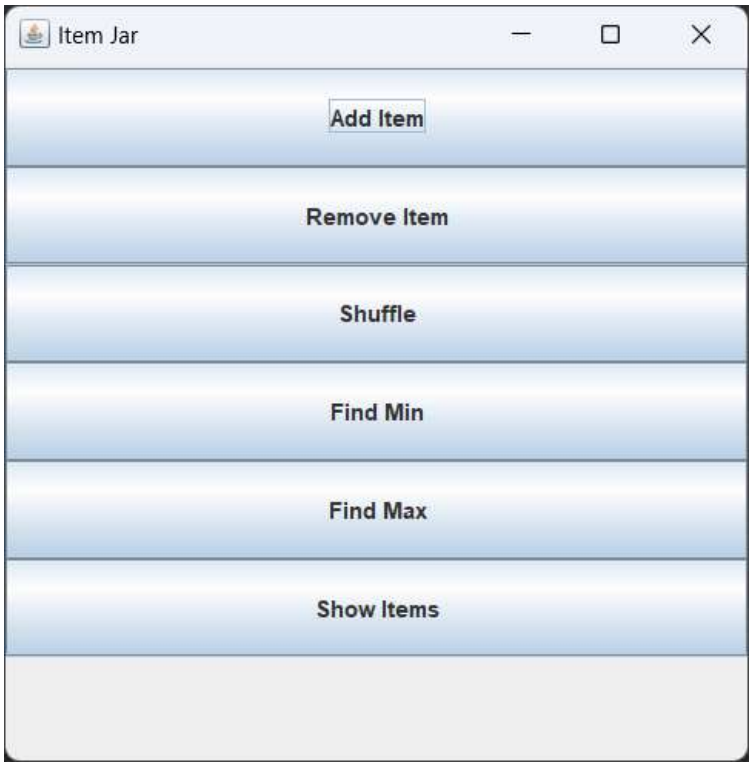

Рисунок 2.1. Початковий вигляд програми.

Як бачимо у програмі можна:

- Покласти предмет у банку - рисунок 2.2,
- Вийняти предмет з банки - рисунок 2.3,
- Потрясти банку - рисунки 2.5,
- Знайти найменший предмет - рисунок 2.6,
- Знайти найбільший предмет - рисунок 2.7,
- Переглянути предмети - рисунок 2.8.

Якщо ми натиснемо кнопку «Add Item», то отримаємо вікно, де можемо ввести значення предмету.

Рисунок 2.2. Вікно покладення предмету у банку.

Після того як ми поклали у банку предмети: «123», «234», «345», «456» та «567», давайте заберемо з банки предмет з індексом «2», натиснувши кнопку «Remove Item».

Рисунок 2.3. Вікно виймання предмету з банки.

Зміни після цієї операції наведені на рисунку 2.4 (натиск на кнопку «Show Items»).

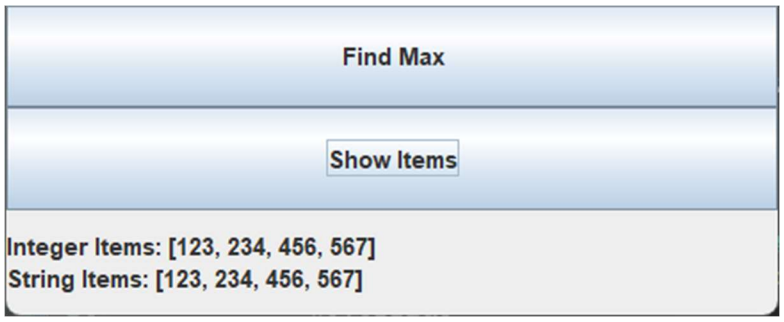

Рисунок 2.4. Вікно змін після видалення елементу.

Також, банку можна трясти, натиснувши кнопку «Shuffle». Тряска банки працює так: елементи переплутуються між собою, а саме елементи міняють свій порядок у випадковій послідовності.

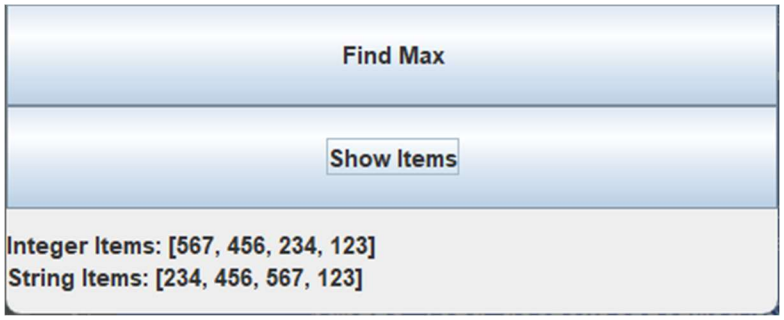

Рисунок 2.5. Банка, після тряски.

Відповідно, можна знайти мінімальний та максимальний елементи банки. Для чисел це працює стандартно: найменше число є мінімальним, а найбільше – максимальним. Для String це працює так: враховується перший Char його код порівнюється зі всіма іншими першими символами елементів банки, таким чином, вибирається предмет з найбільшим та найменшим кодом першого Char. Переглянути елементи банки можна, натиснувши кнопку «Show Items».

Рисунок 2.6. Знаходження мінімального предмету банки.

Рисунок 2.7. Знаходження максимального предмету банки.

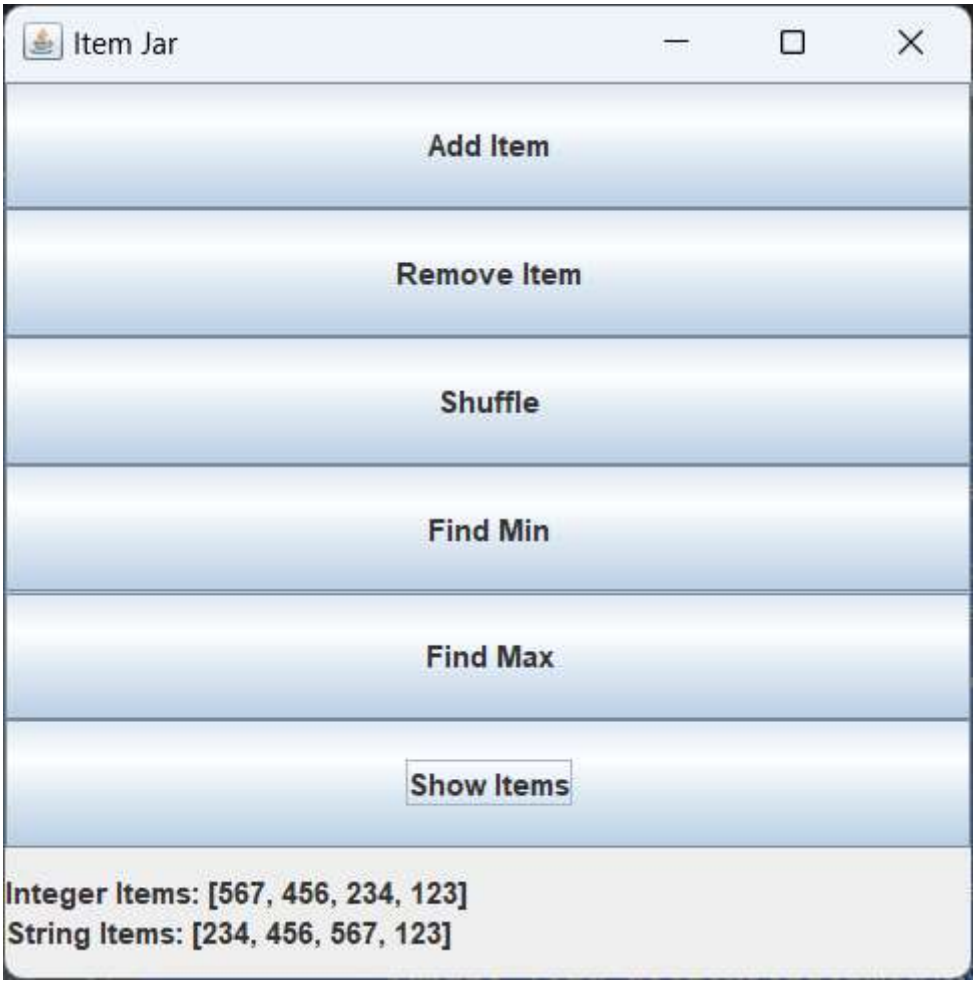

Рисунок 2.8. Показ всіх елементів банки.

ДОКУМЕНТАЦІЯ

Згенерував документацію, фрагмент якої навів на рисунку 2.8.

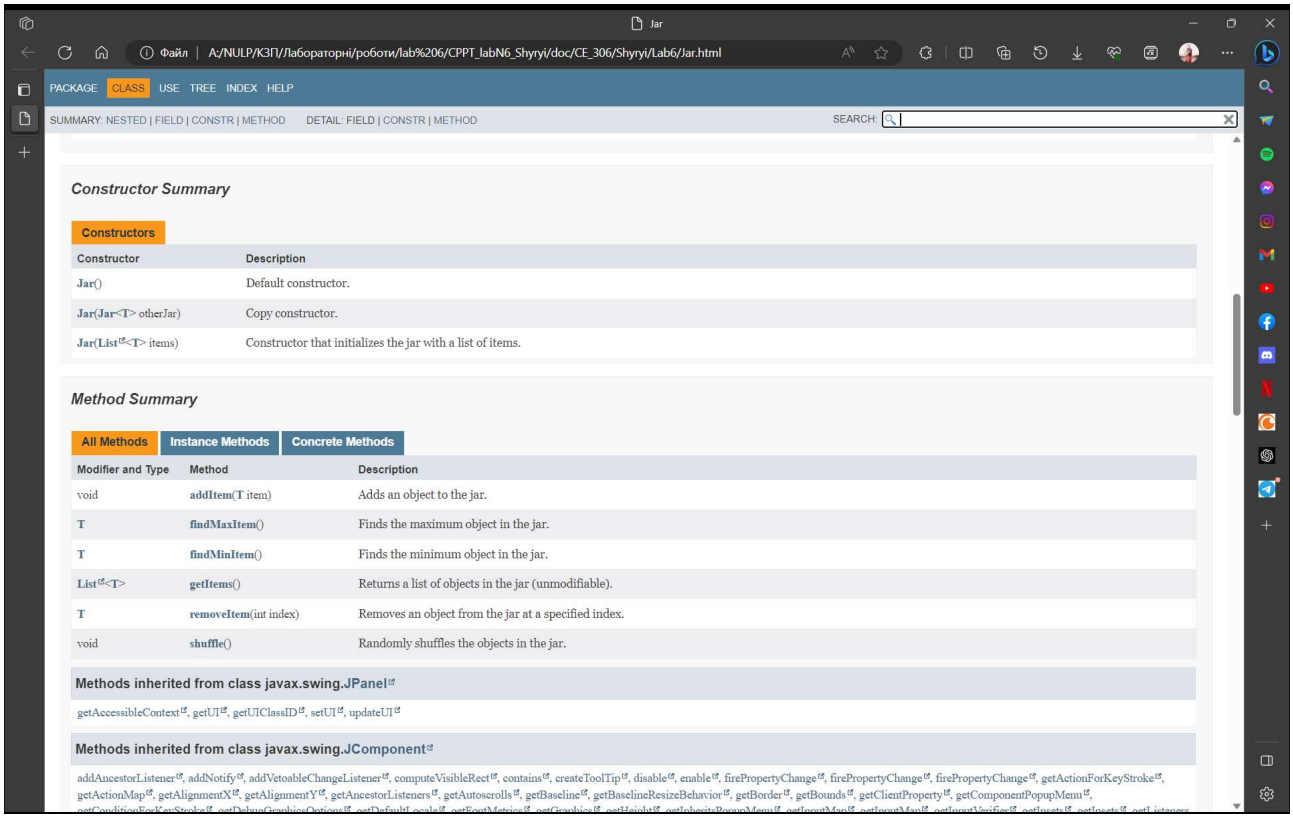

Рисунок 2.9. Фрагмент згенерованої документації.

ВІДПОВІДІ НА КОНТРОЛЬНІ ПИТАННЯ

ДАЙТЕ ВИЗНАЧЕННЯ ТЕРМІНУ «ПАРАМЕТРИЗОВАНЕ ПРОГРАМУВАННЯ».

Параметризоване програмування - це підхід у програмуванні, який дозволяє створювати загальні структури, які можна налаштовувати для різних типів даних чи функціональностей.

РОЗКРИЙТЕ СИНТАКСИС ВИЗНАЧЕННЯ ПРОСТОГО ПАРАМЕТРИЗОВАНОГО КЛАСУ.

Синтаксис наведено у лістингу 2.3.

Лістинг 2.3.

```
public class MyGenericClass<T> {
    // Код класу з використанням параметра T
}
```

РОЗКРИЙТЕ СИНТАКСИС СТВОРЕННЯ ОБ'ЄКТУ ПАРАМЕТРИЗОВАНОГО КЛАСУ.

Синтаксис наведено у лістингу 2.4.

Лістинг 2.4.

```
MyGenericClass<Integer> obj = new MyGenericClass<Integer>();
```

РОЗКРИЙТЕ СИНТАКСИС ВИЗНАЧЕННЯ ПАРАМЕТРИЗОВАНОГО МЕТОДУ.

Синтаксис наведено у лістингу 2.5.

Лістинг 2.5.

```
public <T> void myGenericMethod(T value) {
    // Код методу з використанням параметра T
}
```

РОЗКРИЙТЕ СИНТАКСИС ВИКЛИКУ ПАРАМЕТРИЗОВАНОГО МЕТОДУ.

Синтаксис наведено у лістингу 2.6.

Лістинг 2.6.

```
myGenericMethod(42); // Виклик методу зі значенням типу Integer
```

ЯКУ РОЛЬ ВІДІГРАЄ ВСТАНОВЛЕННЯ ОБМЕЖЕНЬ ДЛЯ ЗМІННИХ ТИПІВ?

Встановлення обмежень для змінних типів дозволяє обмежити допустимі типи даних, які можна використовувати як параметр типу в параметризованих класах або методах. Це робить програму більш безпечною і допомагає уникнути помилок під час компіляції.

ЯК ВСТАНОВИТИ ОБМЕЖЕННЯ ДЛЯ ЗМІННИХ ТИПІВ?

Обмеження для змінних типів можна встановити за допомогою ключового слова extends для обмеження верхньої межі або super для обмеження нижньої межі. Приклад наведений у лістингу 2.7.

Лістинг 2.7.

```
public class MyGenericClass<T extends Number> {
    // Код класу, де T обмежено типами-нащадками класу Number
}
```

РОЗКРИЙТЕ ПРАВИЛА СПАДКУВАННЯ ПАРАМЕТРИЗОВАНИХ ТИПІВ.

Параметризовані типи не спадкові. Тобто, якщо у вас є параметризований клас A<T>, і ви створюєте підклас B від A, то B не буде параметризованим класом, і тип T не буде доступним у B.

ЯКЕ ПРИЗНАЧЕННЯ ПІДСТАНОВОЧНИХ ТИПІВ?

Підстановочні типи (wildcards) використовуються для більш гнучкого програмування, коли точний тип не є важливим, а важлива лише ієрархія типів. Вони дозволяють передавати аргументи методам з різними типами, що є підтипами вказаного типу.

ЗАСТОСУВАННЯ ПІДСТАНОВОЧНИХ ТИПІВ.

Підстановочні типи часто використовуються при роботі з колекціями, якщо вам потрібно приймати або повертати різні типи об'єктів, які є підтипами загального типу.

ВИСНОВОК

Створив параметризований клас, який реалізує предметну область задану варіантом. Цей клас містить методи для опрацювання даних, включаючи розміщення та виймання елементів. Парні варіанти реалізують пошук мінімального елементу, а непарні - максимального. Написав програму-драйвер для класу, яка містить мінімум два різні класи, екземпляри яких розміщуються у створеному класі-контейнері. Код лабораторної роботи написано на мові програмування Java та включає коментарі, які допомагають автоматично згенерувати документацію до розробленого пакету. Згенеровано документацію до розробленого пакету, яка містить опис класів та методів, їх параметри та повернені значення.