Лабораторна робота №5

Тема: Java Collections Framework

Мета роботи: робота з Java Collections Framework

Хід роботи:

Завдання 1. Створити консольний Java проект java_lab_5 з пакетом com.education.ztu

Завдання 2. Створити клас Product та задати йому поля та методи на власний вибір.

Лістинг програми:

Product.java:

```
package com.education.ztu;

import java.util.Objects;

public class Product {
    private String name;
    private double price;
    private int quantity;

    public Product(String name, double price, int quantity) {
        this.name = name;
        this.price = price;
        this.quantity = quantity;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public double getPrice() {
        return price;
    }
    public void setPrice(double price) {
        if (price >= 0) {
            this.price = price;
        } else {
            System.out.println("Price cannot be negative.");
        }
    }
```

```
    public int getQuantity() {
        return quantity;
    }

    public void setQuantity(int quantity) {
        if (quantity >= 0) {
            this.quantity = quantity;
        } else {
            System.out.println("Quantity cannot be negative.");
        }
    }


    public void displayInfo() {
        System.out.println("Product Name: " + name);
        System.out.println("Price: $" + price);
        System.out.println("Quantity: " + quantity);
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj) return true;
        if (obj == null || getClass() != obj.getClass()) return false;
        Product product = (Product) obj;
        return Double.compare(product.price, price) == 0 && quantity ==
product.quantity && Objects.equals(name, product.name);
    }

    @Override
    public int hashCode() {
        return Objects.hash(name, price, quantity);
    }

    @Override
    public String toString() {
        return "Product{Name='" + name + "', Price=" + price + ", Quantity=" +
quantity + "}";
    }

}
```

Завдання 3. Створити динамічний масив, що містить об'єкти класу Product: • Використовуємо клас ArrayList або LinkedList. • Продемонструвати роботу з масивом використовуючи різні методи (add, addAll, get, indexOf, lastIndexOf, iterator, listIterator, remove, set, sort, subList, clear, contains, isEmpty, retainAll, size, toArray)

Лістинг програми:

Main.java:

```
package com.education.ztu;

import java.util.*;

public class Main {
    public static void main(String[] args) {
        List<Product> products = new ArrayList<>();
```

```
        products.add(new Product("Cheese", 70.20, 20));
        products.add(new Product("Juice", 45.00, 10));
        products.add(new Product("Yogurts", 28.50, 30));
        products.add(new Product("Ice cream", 18.99, 5));


        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.println("\nChoose an option:");
            System.out.println("======================");
            System.out.println("1. Add a product");
            System.out.println("2. Remove a product by index");
            System.out.println("3. Display all products");
            System.out.println("4. Sort products by price");
            System.out.println("5. Get product by index");
            System.out.println("6. Display size of the list");
            System.out.println("7. Convert list to array and display");
            System.out.println("8. Exit");

            int choice = scanner.nextInt();
            scanner.nextLine();

            switch (choice) {
                case 1:
                    System.out.print("Enter product name: ");
                    String name = scanner.nextLine();
                    System.out.print("Enter product price: ");
                    double price = scanner.nextDouble();
                    System.out.print("Enter product quantity: ");
                    int quantity = scanner.nextInt();
                    products.add(new Product(name, price, quantity));
                    System.out.println("Product added successfully.");
                    break;

                case 2:
                    System.out.print("Enter index of product to remove: ");
                    int indexToRemove = scanner.nextInt();
                    if (indexToRemove >= 0 && indexToRemove < products.size()) {
                        products.remove(indexToRemove);
                        System.out.println("Product removed.");
                    } else {
                        System.out.println("Invalid index.");
                    }
                    break;

                case 3:
                    System.out.println("All products:");
                    for (Product product : products) {
                        product.displayInfo();
                        System.out.println("---------------");
                    }
                    break;

                case 4:

products.sort(Comparator.comparingDouble(Product::getPrice));
                    System.out.println("Products sorted by price:");
                    for (Product product : products) {
                        product.displayInfo();
                        System.out.println();
                    }
                    break;
```

```
                case 5:
                    System.out.print("Enter index to get product: ");
                    int index = scanner.nextInt();
                    if (index >= 0 && index < products.size()) {
                        System.out.println("Product at index " + index + ": " +
products.get(index).getName());
                    } else {
                        System.out.println("Invalid index.");
                    }
                    break;

                case 6:
                    System.out.println("Size of the list: " + products.size());
                    break;

                case 7:
                    Product[] productArray = products.toArray(new Product[0]);
                    System.out.println("Array of products:");
                    for (Product product : productArray) {
                        product.displayInfo();
                        System.out.println();
                    }
                    break;

                case 8:
                    System.out.println("Exiting...");
                    scanner.close();
                    return;

                default:
                    System.out.println("Invalid choice, please try again.");
            }
        }
    }
}
```

Результат виконання програми:

```
Choose an option:
=======================
1. Add a product
2. Remove a product by index
3. Display all products
4. Sort products by price
5. Get product by index
6. Display size of the list
7. Convert list to array and display
8. Exit
```

Рис.1 Меню для вибору дії

```
3
All products:
Product Name: Cheese
Price: $70.2
Quantity: 20
--------------
Product Name: Juice
Price: $45.0
Quantity: 10
--------------
Product Name: Yogurts
Price: $28.5
Quantity: 30
--------------
Product Name: Ice cream
Price: $18.99
Quantity: 5
--------------
```

```
5
Enter index to get product: 2
Product at index 2: Yogurts
```

```
6
Size of the list: 4
```

```
2
Enter index of product to remove: 2
Product removed.
```

Рис.2 Виконання певних дії

Завдання 4. Створити чергу, що містить об'єкти класу Product:

- Використовуємо клас ArrayDeque.
- Продемонструвати роботу з чергою використовуючи методи (push, offerLast, getFirst, peekLast, pop, removeLast, pollLast та інші)

Лістинг програми:

MainQueue.java:

```java
package com.education.ztu;

import java.util.ArrayDeque;
import java.util.Deque;

public class MainQueue {
    public static void main(String[] args) {
        ArrayDeque<Product> queue = new ArrayDeque<>();

        queue.push(new Product("Cheese", 70.20, 20));
        queue.offerLast(new Product("Juice", 45.00, 10));

        System.out.println("First: " + queue.getFirst());
        System.out.println("Peek Last: " + queue.peekLast());

        queue.pop();
        System.out.println("After pop: " + queue);

        queue.removeLast();
        System.out.println("After removeLast: " + queue);
    }
}
```

Результат виконання програми:

```
First: Product{Name='Cheese', Price=70.2, Quantity=20}
Peek Last: Product{Name='Juice', Price=45.0, Quantity=10}
After pop: [Product{Name='Juice', Price=45.0, Quantity=10}]
After removeLast: []
```

Рис.3 Завдання 4

Завдання 5. Створити множину, що містить об'єкти класу Product:

- Використовуємо клас TreeSet.
- Продемонструвати роботу з множиною використовуючи методи (add, first, last, headSet, subSet, tailSet, ceiling, floor, higher, lower, pollFirst, pollLast, descendingSet)

Лістинг програми:

MainTreeSet.java:

```
package com.education.ztu;

import java.util.Comparator;
import java.util.NavigableSet;
import java.util.Set;
import java.util.TreeSet;

public class MainTreeSet {
    public static void main(String[] args) {
        TreeSet<Product> productSet = new
TreeSet<>(Comparator.comparingDouble(Product::getPrice));

        productSet.add(new Product("Cheese", 70.20, 20));
        productSet.add(new Product("Juice", 45.00, 10));
        productSet.add(new Product("Yogurts", 28.50, 30));
        productSet.add(new Product("Ice cream", 18.99, 5));

        Product searchProduct = new Product("", 45.00, 0);

        System.out.println("\nFirst product in the set: " + productSet.first());
        System.out.println("Last product in the set: " + productSet.last());

        System.out.println("\nProduct with price >= 45.00: " +
productSet.ceiling(searchProduct));
        System.out.println("Product with price <= 45.00: " +
productSet.floor(searchProduct));
        System.out.println("Product with price > 45.00: " +
productSet.higher(searchProduct));
        System.out.println("Product with price < 45.00: " +
productSet.lower(searchProduct));
    }
}
```

Результат виконання програми:

```
First product in the set: Product{Name='Ice cream', Price=18.99, Quantity=5}
Last product in the set: Product{Name='Cheese', Price=70.2, Quantity=20}

Product with price >= 45.00: Product{Name='Juice', Price=45.0, Quantity=10}
Product with price <= 45.00: Product{Name='Juice', Price=45.0, Quantity=10}
Product with price > 45.00: Product{Name='Cheese', Price=70.2, Quantity=20}
Product with price < 45.00: Product{Name='Yogurts', Price=28.5, Quantity=30}
```

Рис.4 Завдання 5

Завдання 5. Створити Map що містить пари (ключ, значення) - ім'я продукту та об'єкт продукту (клас Product).

- Використовуємо клас HashMap,
- Продемонструвати роботу з Map використовуючи методи (put, get, get, containsKey, containsValue, clear, putIfAbsent, keySet, values, putAll, remove, size)
- Викликати метод entrySet та продемонструвати роботу з набором значень, що він поверне (getKey, getValue, setValue)

Лістинг програми:

MainHashMap.java:

```
package com.education.ztu;

import java.util.HashMap;
import java.util.Map;

public class MainHashMap {
    public static void main(String[] args) {
        Map<String, Product> productMap = new HashMap<>();

        productMap.put("Cheese", new Product("Cheese", 70.20, 20));
        productMap.put("Juice", new Product("Juice", 45.00, 10));
        productMap.put("Yogurts", new Product("Yogurts", 28.50, 30));

        System.out.println("Get product by key 'Juice': " +
productMap.get("Juice"));
        System.out.println("Contains 'Cheese' key: " +
productMap.containsKey("Cheese"));
        productMap.remove("Yogurts");
        System.out.println("Map after removing 'Yogurts': " + productMap);
        System.out.println("Size of Map: " + productMap.size());
        System.out.println("Entries in the Map:");
        for (Map.Entry<String, Product> entry : productMap.entrySet()) {
            System.out.println("Key: " + entry.getKey() + ", Value: " +
entry.getValue());
        }
    }
}
```

Результат виконання програми:

```
Get product by key 'Juice': Product{Name='Juice', Price=45.0, Quantity=10}
Contains 'Cheese' key: true
Map after removing 'Yogurts': {Juice=Product{Name='Juice', Price=45.0, Quantity=10}, Cheese=Product{Name='Cheese', Price=70.2, Quantity=20}}
Size of Map: 2
Entries in the Map:
Key: Juice, Value: Product{Name='Juice', Price=45.0, Quantity=10}
Key: Cheese, Value: Product{Name='Cheese', Price=70.2, Quantity=20}
```

Рис.5 Завдання 6

Завдання 6. Продемонструвати роботу з класом Collections:

- Для роботи використати масив створений через Arrays.asList
- Метод Collections.sort()
- Метод Collections.binarySearch()
- Методы Collections.reverse(), Collections.shuffle()
- Метод Collections.fill()
- Методы Collections.max(), Collections.min()
- Метод Collections.copy()
- Метод Collections.rotate()
- Метод Collections.checkedCollection()
- Метод Collections.frequency()

Лістинг програми:

MainCollections.java:

```
package com.education.ztu;

import java.util.*;

public class MainCollections {
    public static void main(String[] args) {
        List<Product> products = new ArrayList<>(Arrays.asList(
                new Product("Cheese", 70.20, 20),
                new Product("Juice", 45.00, 10),
                new Product("Yogurts", 28.50, 30)
        ));

        Collections.sort(products,
Comparator.comparingDouble(Product::getPrice));
        System.out.println("Sorted by price: " + products);

        Collections.sort(products, Comparator.comparing(Product::getName));
        System.out.println("\nSorted by name: " + products);

        int index = Collections.binarySearch(products, new Product("Juice",
45.00, 0), Comparator.comparingDouble(Product::getPrice));
        System.out.println("\nBinary search for Juice by price: " + index);

        Collections.reverse(products);
        System.out.println("\nReversed list: " + products);

        Collections.shuffle(products);
        System.out.println("\nShuffled list: " + products);
```

```
        List<Product> fillList = new ArrayList<>(Arrays.asList(new Product[3]));
        Collections.fill(fillList, new Product("Default Product", 100.0, 0));
        System.out.println("\nFilled list: " + fillList);

        Product maxProduct = Collections.max(products,
Comparator.comparingDouble(Product::getPrice));
        Product minProduct = Collections.min(products,
Comparator.comparingDouble(Product::getPrice));
        System.out.println("\nMax price product: " + maxProduct);
        System.out.println("\nMin price product: " + minProduct);

        List<Product> copyList = new ArrayList<>(Arrays.asList(new
Product[products.size()]));
        Collections.copy(copyList, products);
        System.out.println("\nCopied list: " + copyList);

        Collections.rotate(products, 2);
        System.out.println("\nRotated list: " + products);

        Collection<Product> checkedCollection =
Collections.checkedCollection(products, Product.class);
        System.out.println("\nChecked collection: " + checkedCollection);

        int frequency = Collections.frequency(products, new Product("Juice",
45.00, 10));
        System.out.println("\nFrequency of Juice: " + frequency);
    }
}
```

Результат виконання програми:

```
Sorted by price: [Product{Name='Yogurts', Price=28.5, Quantity=30}, Product{Name='Juice', Price=45.0, Quantity=10}, Product{Name=

Sorted by name: [Product{Name='Cheese', Price=70.2, Quantity=20}, Product{Name='Juice', Price=45.0, Quantity=10}, Product{Name='Y

Binary search for Juice by price: 1

Reversed list: [Product{Name='Yogurts', Price=28.5, Quantity=30}, Product{Name='Juice', Price=45.0, Quantity=10}, Product{Name='C

Shuffled list: [Product{Name='Juice', Price=45.0, Quantity=10}, Product{Name='Cheese', Price=70.2, Quantity=20}, Product{Name='Yo

Filled list: [Product{Name='Default Product', Price=100.0, Quantity=0}, Product{Name='Default Product', Price=100.0, Quantity=0},

Max price product: Product{Name='Cheese', Price=70.2, Quantity=20}

Min price product: Product{Name='Yogurts', Price=28.5, Quantity=30}
```

```
Copied list: [Product{Name='Juice', Price=45.0, Quantity=10}, Product{Name='Cheese', Price=70.2, Quantity=20}, Product

Rotated list: [Product{Name='Cheese', Price=70.2, Quantity=20}, Product{Name='Yogurts', Price=28.5, Quantity=30}, Produ

Checked collection: [Product{Name='Cheese', Price=70.2, Quantity=20}, Product{Name='Yogurts', Price=28.5, Quantity=30},

Frequency of Juice: 1
```

Рис.6 Завдання 7

Висновок: під час виконання лабораторної роботи я попрацювала з Java Collections Framework, що є потужним інструментом для організації і обробки даних.