Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Факультет компьютерных технологий и прикладной математики**

**Кафедра вычислительных технологий**

**ЛАБОРАТОРНАЯ РАБОТА №5**

**Дисциплина: Платформо-независимое программирование**

Работу выполнил: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.С. Агаджанян

Направление подготовки: 02.03.02 Фундаментальная информатика и информационные технологии

Преподаватель: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_В. И. Шиян

**Тема**: Параметризация в Java

**Задание**:

Используя коллекции и приемы паратметризации ( а) с помощью интерфейсов-предикатов и б) с лямбда-функцией) решить следующие варианты заданий:

1. Отфильтровать суда в порту, выбрав только низкотоннажные суда для легких перевозок. Выполнить г*руппировку* списка судов по категориям (буксиры/пассажирские/торговые), *поиск* самого дорогого по стоимости перевозки людей и грузов судна, сортировку судов по названию по алфавиту из общего списка и по категориям. Отдельно для буксиров и отдельно для пассажирских и отдельно для торговых судов выполнить выбор по грузоподъемности (тоннажу) (ввод параметра от…до) и сортировку с выбором самого малогабаритного и самого высокогабаритного судна.

**Листинг:**

**Main.java**

import java.util.\*;

import java.util.stream.Collectors;

public class Main {

    public static void main(String[] args) {

        // Пример данных о судах

        List<Ship> ships = new ArrayList<>();

        ships.add(new Ship("Alpha", "Cargo", 1500, 5000, 100));

        ships.add(new Ship("Bravo", "Tug", 500, 1000, 30));

        ships.add(new Ship("Charlie", "Passenger", 2000, 7000, 150));

        ships.add(new Ship("Delta", "Cargo", 800, 3000, 80));

        ships.add(new Ship("Echo", "Tug", 300, 800, 25));

        ships.add(new Ship("Foxtrot", "Passenger", 1200, 4500, 120));

        ships.add(new Ship("Golf", "Cargo", 2000, 6000, 110));

        ships.add(new Ship("Hotel", "Tug", 700, 1200, 35));

        ships.add(new Ship("India", "Passenger", 1500, 5500, 130));

        // 1. Фильтрация низкотоннажных судов (например, до 1000 тонн)

        double maxTonnageForLightTransport = 1000;

        ShipPredicate isLowTonnage = ship -> ship.getTonnage() <= maxTonnageForLightTransport;

        List<Ship> lowTonnageShips = ships.stream()

                .filter(ship -> isLowTonnage.test(ship)) // используем наш ShipPredicate

                .collect(Collectors.toList());

        System.out.println("Низкотоннажные суда (до " + maxTonnageForLightTransport + " тонн):");

        lowTonnageShips.forEach(System.out::println);

        System.out.println("---");

        // 2. Группировка по категориям

        Map<String, List<Ship>> shipsByCategory = ships.stream()

                .collect(Collectors.groupingBy(Ship::getCategory));

        System.out.println("Группировка по категориям:");

        shipsByCategory.forEach((category, shipList) -> {

            System.out.println(category + ":");

            shipList.forEach(System.out::println);

            System.out.println();

        });

        System.out.println("---");

        // 3. Поиск самого дорогого по стоимости перевозки судна

        Optional<Ship> mostExpensiveShip = ships.stream()

                .max(Comparator.comparingDouble(Ship::getTransportCost));

        System.out.println("Самое дорогое судно по стоимости перевозки: " + mostExpensiveShip.orElse(null));

        System.out.println("---");

        // 4. Сортировка по названию (общий список)

        List<Ship> sortedByName = ships.stream()

                .sorted(Comparator.comparing(Ship::getName))

                .collect(Collectors.toList());

        System.out.println("Сортировка по названию (общий список):");

        sortedByName.forEach(System.out::println);

        System.out.println("---");

        // 5. Сортировка по названию по категориям

        System.out.println("Сортировка по названию по категориям:");

        shipsByCategory.forEach((category, shipList) -> {

            System.out.println(category + ":");

            List<Ship> sortedByCategoryAndName = shipList.stream()

                    .sorted(Comparator.comparing(Ship::getName))

                    .collect(Collectors.toList());

            sortedByCategoryAndName.forEach(System.out::println);

            System.out.println();

        });

        System.out.println("---");

        // 6. Выбор по грузоподъемности и сортировка (для каждой категории)

        double minTonnage = 400;

        double maxTonnage = 1800;

        System.out.println("Выбор по грузоподъемности (" + minTonnage + " - " + maxTonnage + " тонн) и сортировка по габаритам (длине):");

        shipsByCategory.forEach((category, shipList) -> {

            System.out.println(category + ":");

            // Фильтрация по тоннажу с использованием ShipPredicate и лямбда-выражения

            ShipPredicate tonnageRangePredicate = ship -> ship.getTonnage() >= minTonnage && ship.getTonnage() <= maxTonnage;

            List<Ship> filteredByTonnage = shipList.stream()

                    .filter(ship -> tonnageRangePredicate.test(ship))

                    .collect(Collectors.toList());

            if (!filteredByTonnage.isEmpty()) {

                // Сортировка по длине

                List<Ship> sortedByLength = filteredByTonnage.stream()

                        .sorted(Comparator.comparing(Ship::getLength))

                        .collect(Collectors.toList());

                // Самое малогабаритное судно

                Ship smallestShip = sortedByLength.get(0);

                // Самое высокогабаритное судно

                Ship largestShip = sortedByLength.get(sortedByLength.size() - 1);

                System.out.println("  Суда в диапазоне тоннажа:");

                sortedByLength.forEach(System.out::println);

                System.out.println("  Самое малогабаритное: " + smallestShip);

                System.out.println("  Самое высокогабаритное: " + largestShip);

            } else {

                System.out.println("  Нет судов в заданном диапазоне тоннажа.");

            }

            System.out.println();

        });

        System.out.println("---");

    }

}

**Ship.java**

public class Ship {

    private String name;

    private String category; // "Tug", "Passenger", "Cargo"

    private double tonnage;  // Тоннаж

    private double transportCost; // Стоимость перевозки

    private double length;      // Длина (для определения габаритов)

    public Ship(String name, String category, double tonnage, double transportCost, double length) {

        this.name = name;

        this.category = category;

        this.tonnage = tonnage;

        this.transportCost = transportCost;

        this.length = length;

    }

    public String getName() {

        return name;

    }

    public String getCategory() {

        return category;

    }

    public double getTonnage() {

        return tonnage;

    }

    public double getTransportCost() {

        return transportCost;

    }

    public double getLength() {

        return length;

    }

    @Override

    public String toString() {

        return "Ship{" +

                "name='" + name + '\'' +

                ", category='" + category + '\'' +

                ", tonnage=" + tonnage +

                ", transportCost=" + transportCost +

                ", length=" + length +

                '}';

    }

}

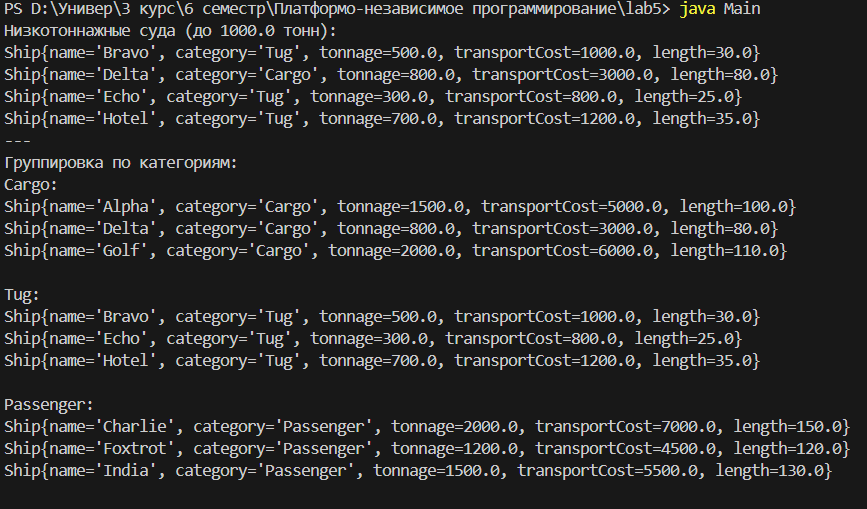
**ShipPredicate.java**

public interface ShipPredicate {

    boolean test(Ship ship);

}

**Результат:**

****

**Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.**

**Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.**