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 №6**

**Дисциплина: Программирование для мобильных платформ**

Работу выполнил: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_А.Д. Воробьев

Направление подготовки: 02.03.02 Фундаментальная информатика и информационные технологии

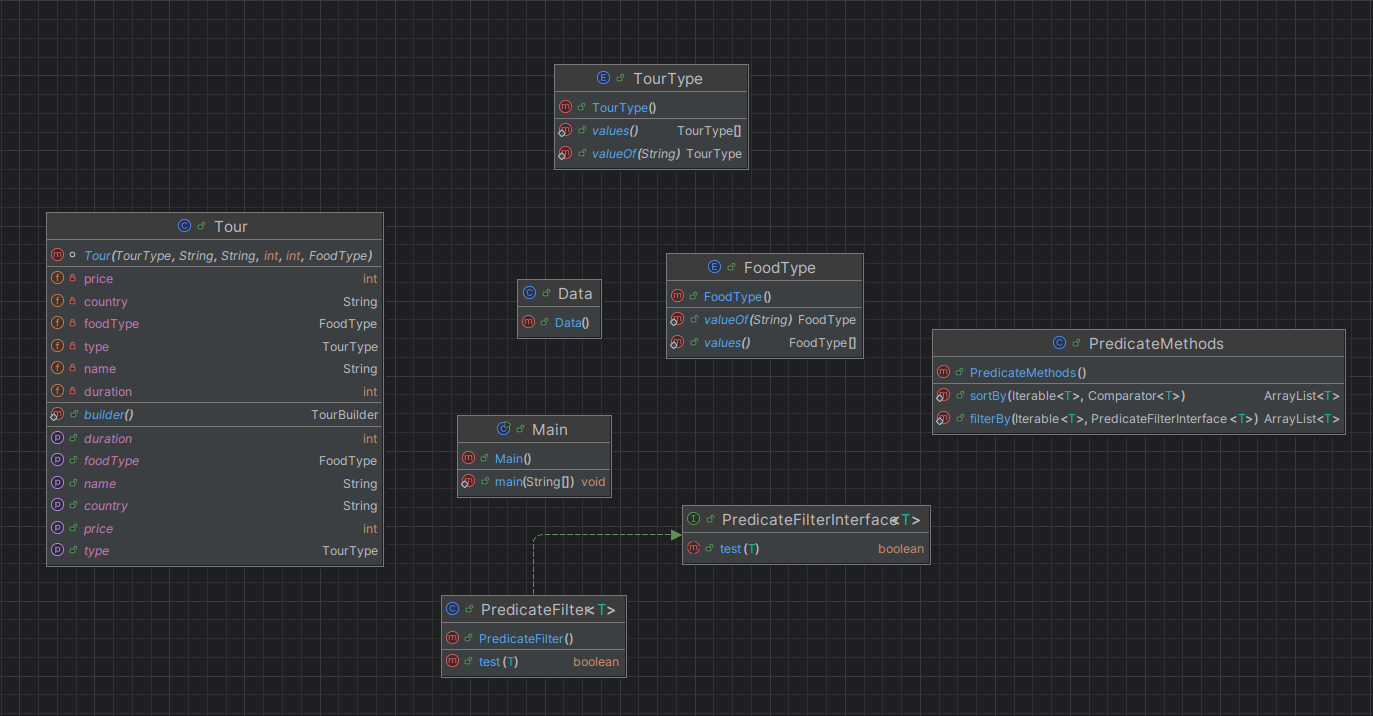
Преподаватель: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Т.А. Приходько

**Тема работы:** Параметризация в JAVA.

**Задача -** Используя коллекции и приемы паратметризации (  а) с помощью интерфейсов-предикатов и б) с лямбда-функцией)

**Мой вариант (14) -** В туристическом агентстве предлагаются различные туры: пляжные, экскурсионные, горнолыжные и круизы. Каждый тур характеризуется названием, страной, длительностью, ценой и типом питания (все включено/завтраки/без питания). Требуется разделить туры по типам, отсортировать по длительности и цене внутри каждого типа. Реализовать возможность фильтрации туров по стране и заданному диапазону цен, а также поиск тура по названию.

На рисунке ниже представлена диаграмма разработанных классов:

Рисунок 1 — Диаграмма классов

**Листинг программ**

Файл Data.java

public class Data {  
 public static ArrayList<Tour> *data* = new ArrayList<>(Arrays.*asList*(  
 Tour.*builder*()  
 .type(TourType.*beach*)  
 .foodType(FoodType.*allInclusive*)  
 .price(1000)  
 .duration(123)  
 .country("Russia")  
 .name("Welcome to Russia")  
 .build(),  
 Tour.*builder*()  
 .type(TourType.*ski*)  
 .foodType(FoodType.*noFood*)  
 .price(2000)  
 .duration(123)  
 .country("Russia")  
 .name("Welcome to Russia")  
 .build(),  
 Tour.*builder*()  
 .type(TourType.*cruise*)  
 .foodType(FoodType.*breakFast*)  
 .price(1000)  
 .duration(123)  
 .country("Germany")  
 .name("Welcome to Germany")  
 .build(),  
 Tour.*builder*()  
 .type(TourType.*excursion*)  
 .foodType(FoodType.*allInclusive*)  
 .price(5000)  
 .duration(123)  
 .country("Russia")  
 .name("Welcome to Russia")  
 .build()  
 ));  
}

Файл FoodType.java

public enum FoodType {  
 *allInclusive*,  
 *noFood*,  
 *breakFast*}

Файл Tour.java

@Getter  
@Setter  
@Builder  
public class Tour {  
 private TourType type;  
 private String country;  
 private String name;  
 private int duration;  
 private int price;  
 private FoodType foodType;  
}

Файл TourType.java

public enum TourType {  
 *beach*,  
 *excursion*,  
 *ski*,  
 *cruise*}

Файл PredicateFilter.java

public class PredicateFilter<T> implements PredicateFilterInterface<T>{  
 @Override  
 public boolean test(T o) {  
 return true;  
 }  
}

Файл PredicateFilterInterface.java

public interface PredicateFilterInterface<T>{  
 boolean test(T t);  
}

Файл PredicateMethods.java

public class PredicateMethods {  
 public static <T> ArrayList<T> filterBy(Iterable<T> initialIterable, PredicateFilterInterface<T> predicate) {  
 ArrayList<T> list = new ArrayList<>();  
 for (T item : initialIterable) {  
 if (predicate.test(item)) {  
 list.add(item);  
 }  
 }  
 return list;  
 }  
  
 public static <T> ArrayList<T> sortBy(Iterable<T> initialList, Comparator<T> predicate) {  
 ArrayList<T> list = new ArrayList<>();  
 initialList.forEach(list::add);  
  
 list.sort(predicate);  
  
 return list;  
 }  
}

Файл Main.java

public class Main {  
 public static void main(String[] args) {  
 ArrayList<Tour> data = Data.*data*;  
 System.*out*.println(data);  
  
 ArrayList<Tour> filteredData = PredicateMethods.*filterBy*(data, new PredicateFilter<>() {  
 @Override  
 public boolean test(Tour item) {  
 return item.getPrice() > 1000;  
 }  
 });  
  
 System.*out*.println(filteredData);  
  
 ArrayList<Tour> sortedData = PredicateMethods.*sortBy*(data, new Comparator<Tour>() {  
 @Override  
 public int compare(Tour first, Tour second) {  
 return Integer.*compare*(first.getPrice(), second.getPrice());  
 }  
 });  
  
 System.*out*.println(sortedData);  
  
 ArrayList<Tour> filteredDataLambda = PredicateMethods.*filterBy*(data,  
 (item) -> item.getPrice() > 1000  
 );  
  
 System.*out*.println(filteredDataLambda);  
  
 ArrayList<Tour> sortedDataLambda = PredicateMethods.*sortBy*(data,  
 Comparator.*comparingInt*(Tour::getPrice)  
 );  
  
 System.*out*.println(sortedDataLambda);  
 }  
}