МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий

ОТЧЁТ

по лабораторной работе №1

"Вариант 20. (Хранение маршрутов общественного транспорта)"

по дисциплине:

Шаблоны проектирования программного обеспечения

|  |  |
| --- | --- |
|  | РУКОВОДИТЕЛЬ:  Жевнерчук Д. В.    СТУДЕНТЫ:  Зинкин Д.А.  Лекарев Р.В.  Деулин А.Л.  группы :21-ВМЗ-2 |

Нижний Новгород

2023

**Цель работы:** построить объектно-ориентированную модель предметной области с обязательным использованием 2-3 шаблонов GoF и реализовать ее на объектно-ориентированном языке программирования.

**Задание:** Разработайте объектно-ориентированную модель и программу, для хранения информации о маршрутах общественного транспорта, загруженности участков (отрезки пути между остановками), а также для получения комбинированных маршрутов, с помощью которых можно добраться из пункта А в пункт B. Программа должна быть расширяема по новым маршрутам и алгоритма оценки загруженности.

**Входные параметры:** 2 наименования остановок [ Откуда , Куда ]

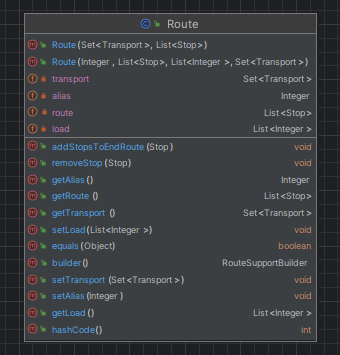
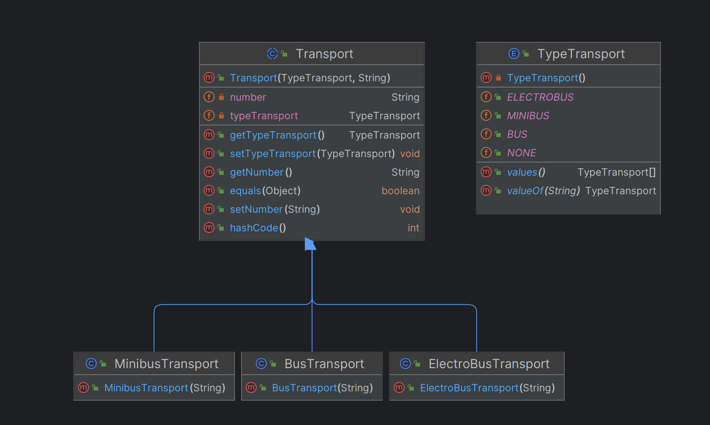


Рисунок 1. UML диаграмма модели в программе.(POJO)

Транспорт расширяем по типу.

В решении этой задачи мы использовали паттерн «Фабрика»

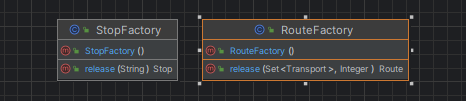


Рисунок 2. UML диаграмма класса(«Фабрика») в программе.

Так же в решении этой задачи мы использовали паттерн «Строитель»

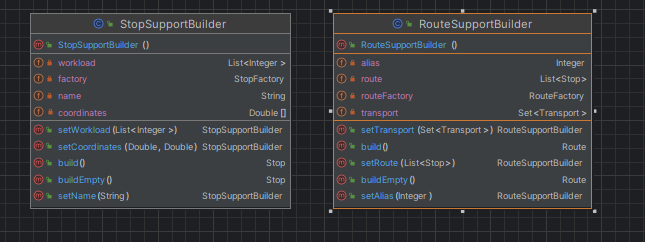


Рисунок 3. UML диаграмма класса(«Строитель») в программе.

В программе логика поиска и оценка загруженности пути вынесена в отдельные сервисы.

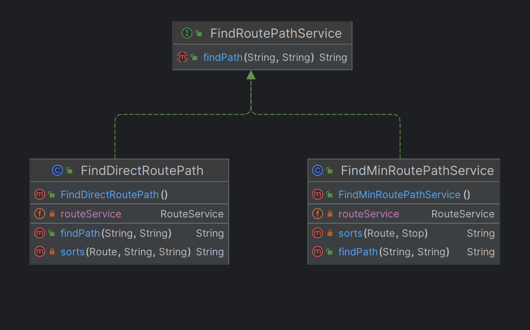


Рисунок 4.1. UML диаграмма класса(«Сервис поиска пути») в программе.

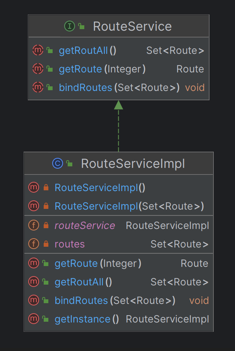


Рисунок 4.2. UML диаграмма класса(«Сервис хранения , выдачи , изменения машрутов») в программе.

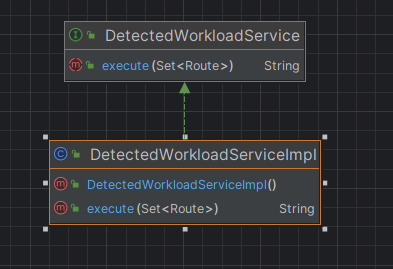


Рисунок 4.3. UML диаграмма класса(«Сервис оценки загруженности») в программе.

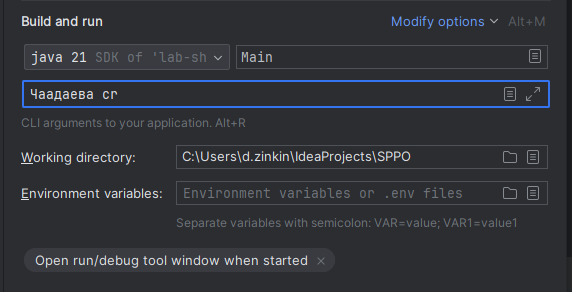
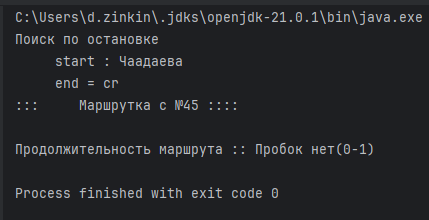


Рисунок 5. Пример работы программы  
Входные параметры:  
[Чаадаева,Сr]

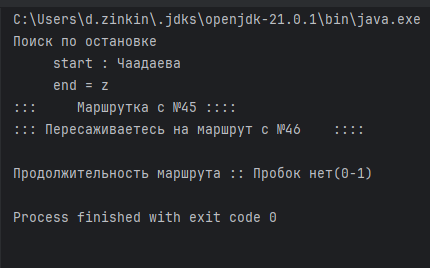
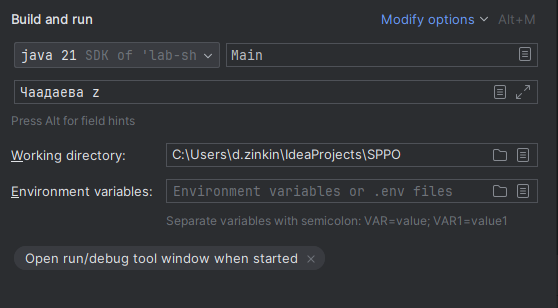


Рисунок 5. Пример работы программы  
Входные параметры:  
[Чаадаева,z]



**Листинг-программы**

**//** Билдер маршрутов

public class RouteSupportBuilder {

private RouteFactory routeFactory = new RouteFactory();

private Integer alias = -1;

private List<Stop> route = new ArrayList<>();

private Set<Transport> transport;

public RouteSupportBuilder setRoute(List<Stop> route) {

this.route = route;

return this;

}

public RouteSupportBuilder setTransport(Set<Transport> transport) {

this.transport = transport;

return this;

}

public RouteSupportBuilder setAlias(Integer alias){

this.alias = alias;

return this;

}

public Route build(){

Route route = routeFactory.release(transport,alias);

route.setAlias(alias);

for (var el:this.route) {

route.addStopsToEndRoute(el);

}

return route;

}

public Route buildEmpty(){

Route route = routeFactory.release(null,null);

route.setAlias(alias);

for (var el:this.route) {

route.addStopsToEndRoute(el);

}

return route;

}

}

**//** Билдер остановок

public class StopSupportBuilder {

private StopFactory factory = new StopFactory();

private String name = "NONE";

private final Double[] coordinates = new Double[2];

private List<Integer> workload = new ArrayList<>();

public StopSupportBuilder setName(String name) {

this.name = name;

return this;

}

public StopSupportBuilder setCoordinates(Double x,Double y) {

this.coordinates[0] = x;

this.coordinates[1] = y;

return this;

}

public StopSupportBuilder setWorkload(List<Integer> list){

this.workload = list;

return this;

}

public Stop build(){

Stop stop = factory.release(name);

stop.setCoordinates(coordinates[0],coordinates[1]);

stop.setName(name);

return stop;

}

public Stop buildEmpty(){

return factory.release(name);

}

}

//Фабрика маршрутов

public class RouteFactory {

public Route release(Set<Transport> transport, Integer number) {

return new Route(number,new ArrayList<>(),new ArrayList<>(),transport);

}

}

// Фабрика остановок

public class StopFactory {

public Stop release(String name) {

return new Stop(null,new Double[2]);

}

}

// Перечисление типов транспорта

public enum TypeTransport {

BUS,

MINIBUS,

ELECTROBUS,

NONE

}

//Абстрактная модель транспорта

public abstract class Transport {

private TypeTransport typeTransport;

private String number;

public Transport(TypeTransport typeTransport, String number) {

this.typeTransport = typeTransport;

this.number = number;

}

public TypeTransport getTypeTransport() {

return typeTransport;

}

public void setTypeTransport(TypeTransport typeTransport) {

this.typeTransport = typeTransport;

}

public String getNumber() {

return number;

}

public void setNumber(String number) {

this.number = number;

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (o == null || getClass() != o.getClass()) return false;

Transport transport = (Transport) o;

return typeTransport == transport.typeTransport && Objects.equals(number, transport.number);

}

@Override

public int hashCode() {

return Objects.hash(typeTransport, number);

}

}

// Модель автобуса

public class BusTransport extends Transport {

public BusTransport(String number) {

super(TypeTransport.BUS, number);

}

}

// Модель троллейбуса

public class ElectroBusTransport extends Transport {

public ElectroBusTransport(String number) {

super(TypeTransport.ELECTROBUS, number);

}

}

public class MinibusTransport extends Transport {

public MinibusTransport(String number) {

super(TypeTransport.MINIBUS, number);

}

}

// Модель марщрута

public class Route {

private Integer alias = -1;

private final List<Stop> route;

private List<Integer> load;

private Set<Transport> transport;

public Route(Set<Transport> transport,List<Stop> route) {

this.transport = transport;

this.route = route;

}

public Route(Integer alias, List<Stop> route,List<Integer> load, Set<Transport> transport) {

this.alias = alias;

this.route = route;

this.load = load;

this.transport = transport;

}

public List<Stop> getRoute() {

return route;

}

public void addStopsToEndRoute(Stop stop){

route.add(stop);

}

public void removeStop(Stop stop){

route.remove(stop);

}

public Set<Transport> getTransport() {

return transport;

}

public void setTransport(Set<Transport> transport) {

this.transport = transport;

}

public Integer getAlias() {

return alias;

}

public void setAlias(Integer alias) {

this.alias = alias;

}

public List<Integer> getLoad() {

return load;

}

public void setLoad(List<Integer> load) {

this.load = load;

}

public static RouteSupportBuilder builder(){

return new RouteSupportBuilder();

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (o == null || getClass() != o.getClass()) return false;

Route route1 = (Route) o;

return Objects.equals(route, route1.route);

}

@Override

public int hashCode() {

return Objects.hash(route);

}

}

// Модель остановок

public class Stop {

private String name;

private final Double[] coordinates = new Double[2];;

public Stop(String name, Double[] coordinates) {

this.name = name;

if(this.coordinates.length > coordinates.length) {

for (int i = 0; i < this.coordinates.length; i++) {

this.coordinates[i] = coordinates[i];

}

}

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public void setCoordinates(Double x,Double y){

this.coordinates[0] = x;

this.coordinates[1] = y;

}

public Double[] getCoordinates() {

return coordinates;

}

public static StopSupportBuilder builder(){

return new StopSupportBuilder();

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (o == null || getClass() != o.getClass()) return false;

Stop stop = (Stop) o;

return Objects.equals(name, stop.name);

}

@Override

public int hashCode() {

return Objects.hash(name);

}

@Override

public String toString() {

return String.format("Остановка ощественного транспорта : %s",name);

}

}

// Интерфейс оценки загруженности

public interface DetectedWorkloadService {

String execute(Set<Route> stopList);

}

// Интерфейс поиска пути

public interface FindRoutePathService {

String findPath(String from,String to);

}

// Интерфейс RoutService

public interface RouteService {

Route getRoute(Integer alias);

Set<Route> getRoutAll();

void bindRoutes(Set<Route> routeList);

}

// Далее следует инмплементация этих интерфейсов

public class DetectedWorkloadServiceImpl implements DetectedWorkloadService {

@Override

public String execute(Set<Route> route) {

StringBuffer stringBuffer = new StringBuffer();

route.stream().forEach(action ->{

Map<Integer,Integer> positionAndCount = new HashMap<>();

positionAndCount.put(0,0);

positionAndCount.put(1,0);

positionAndCount.put(2,0);

positionAndCount.put(3,0);

positionAndCount.put(4,0);

positionAndCount.put(5,0);

if(action.getAlias().equals(45)){

if(action.getLoad() == null || action.getLoad().isEmpty()){

stringBuffer.append("Продолжительность маршрута :: Пробок нет(0-1)\n");

}

for(var el:action.getLoad()){

if(!positionAndCount.containsKey(el)){

positionAndCount.put(el,1);

} else {

var nextCount = positionAndCount.get(el)+1;

positionAndCount.put(el,nextCount);

}

}

if(positionAndCount.get(5) > positionAndCount.get(4)

&& positionAndCount.get(5) > positionAndCount.get(3)

&& positionAndCount.get(5) > positionAndCount.get(2)

&& positionAndCount.get(5) > positionAndCount.get(1)

&& positionAndCount.get(5) > positionAndCount.get(0)){

stringBuffer.append("Продолжительность маршрута :: Очень долго(5)");

}

if(positionAndCount.get(4) > positionAndCount.get(5)

&& positionAndCount.get(4) > positionAndCount.get(3)

&& positionAndCount.get(4) > positionAndCount.get(2)

&& positionAndCount.get(4) > positionAndCount.get(1)

&& positionAndCount.get(4) > positionAndCount.get(0)){

stringBuffer.append("Продолжительность маршрута :: Умеренно долго(4)");

}

if(positionAndCount.get(3) > positionAndCount.get(4)

&& positionAndCount.get(3) > positionAndCount.get(5)

&& positionAndCount.get(3) > positionAndCount.get(2)

&& positionAndCount.get(3) > positionAndCount.get(1)

&& positionAndCount.get(3) > positionAndCount.get(0)){

stringBuffer.append("Продолжительность маршрута :: Хорошо(3)");

}

if(positionAndCount.get(2) > positionAndCount.get(4)

&& positionAndCount.get(2) > positionAndCount.get(3)

&& positionAndCount.get(2) > positionAndCount.get(5)

&& positionAndCount.get(2) > positionAndCount.get(1)

&& positionAndCount.get(2) > positionAndCount.get(0)){

stringBuffer.append("Продолжительность маршрута :: Довольно Быстро(2)");

}

if(positionAndCount.get(1) > positionAndCount.get(4)

&& positionAndCount.get(1) > positionAndCount.get(3)

&& positionAndCount.get(1) > positionAndCount.get(2)

&& positionAndCount.get(1) > positionAndCount.get(5)

&& positionAndCount.get(1) > positionAndCount.get(0)){

stringBuffer.append("Продолжительность маршрута :: Пробок нет(0-1)");

}

if(positionAndCount.get(0) > positionAndCount.get(4)

&& positionAndCount.get(0) > positionAndCount.get(3)

&& positionAndCount.get(0) > positionAndCount.get(2)

&& positionAndCount.get(0) > positionAndCount.get(5)

&& positionAndCount.get(0) > positionAndCount.get(1)){

stringBuffer.append("Продолжительность маршрута :: Пробок нет(0-1)");

}

}

});

return stringBuffer.toString();

}

}

public class FindDirectRoutePath implements FindRoutePathService {

private RouteService routeService = RouteServiceImpl.getInstance();

@Override

public String findPath(String from, String to) {

StringBuilder stringBuilder = new StringBuilder();

for(var route:routeService.getRoutAll()){

stringBuilder.append(sorts(route,from, to));

}

return stringBuilder.toString();

}

private String sorts(Route route, String from, String to) {

StringBuilder stringBuilder = new StringBuilder();

if(route.getRoute().contains(Stop.builder().setName(from).build()) && route.getRoute().contains(Stop.builder().setName(to).build())){

stringBuilder.append("::: \tМаршрутка с №").append(route.getAlias()).append("\t::::\n");

}

return stringBuilder.toString();

}

}

public class FindMinRoutePathService implements FindRoutePathService {

private RouteService routeService = RouteServiceImpl.getInstance();

@Override

public String findPath(String from, String to) {

StringBuilder stringBuilder = new StringBuilder();

List<Route> routes = new java.util.ArrayList<>(routeService.getRoutAll().stream().toList());

var stopFrom = Stop.builder().setName(from).build();

var stopTo = Stop.builder().setName(to).build();

int step = 0;

boolean flag = false;

Route thisRoute = null;

for(var route:routes) {

if(route.getRoute().contains(stopFrom)){

stringBuilder.append("::: \tМаршрутка с №").append(route.getAlias()).append("\t::::\n");

for(int i = 0;i < route.getRoute().size()-1;i++){

if (route.getRoute().get(i).equals(stopFrom)) {

thisRoute = route;

stopFrom = route.getRoute().get(i+1);

break;

}

}

}

if(route.getRoute().contains(stopTo)){

flag = true;

}

}

routes.remove(thisRoute);

for(var route:routes){

if(route.getRoute().contains(stopTo)){

stringBuilder.append(sorts(route,stopTo));

}

}

if(!flag){

return "";

}

return stringBuilder.toString();

}

private String sorts(Route route,Stop to) {

StringBuilder stringBuilder = new StringBuilder();

if(route.getRoute().contains(to)){

stringBuilder

.append(":::\tПересаживаетесь на маршрут с №").append(route.getAlias()).append("\t::::\n");

}

return stringBuilder.toString();

}

}

public class RouteServiceImpl implements RouteService {

private static RouteServiceImpl routeService;

public static RouteServiceImpl getInstance() {

if(routeService == null){

routeService = new RouteServiceImpl();

}

return routeService;

}

private Set<Route> routes;

private RouteServiceImpl() {

routes = new HashSet<>();

}

private RouteServiceImpl(Set<Route> routes) {

this.routes = routes;

}

@Override

public Route getRoute(Integer alias) {

for (var route:routes) {

if(route.getAlias().equals(alias)){

return route;

}

}

throw new RuntimeException("Такого маршрута не существует");

}

@Override

public void bindRoutes(Set<Route> routeList) {

routes = routeList;

}

@Override

public Set<Route> getRoutAll() {

return routes;

}

}

**MAIN :**

public class Main {

static RouteService routeService = RouteServiceImpl.getInstance();

static FindRoutePathService findRoutePathService = new FindDirectRoutePath();

static DetectedWorkloadService detectedWorkloadService = new DetectedWorkloadServiceImpl();

public static void main(String[] args) {

if(args.length < 1){

System.err.println("Не верные входные аргументы!");

return;

}

/\*\* Инициализируем \*\*/

String from = args[0];

String to = args[1];

/\*\* Создаем маршруты \*\*/

Set<Route> routes = new HashSet<>();

routes.add(Route.builder()

.setRoute(new ArrayList<>(){{

add(Stop.builder()

.setCoordinates(54.224d,43.12)

.setName("Чаадаева")

.build());

add(Stop.builder()

.setCoordinates(54.225d,43.124)

.setName("t")

.build());

add(Stop.builder()

.setCoordinates(54.226d,43.55)

.setName("cr")

.build());

}})

.setAlias(45)

.setTransport(

new HashSet<>(){{

add(new BusTransport("n222af"));

add(new BusTransport("n555af"));

add(new MinibusTransport("s425ff"))

;}}

)

.build());

routes.add(Route.builder()

.setRoute(new ArrayList<>(){{

add(Stop.builder()

.setCoordinates(54.224d,43.12)

.setName("cr")

.build());

add(Stop.builder()

.setCoordinates(0.44,0.33)

.setName("z")

.build());

}})

.setAlias(46)

.setTransport(

new HashSet<>(){{

add(new BusTransport("n222ax"));

add(new BusTransport("n555ax"));

}}

)

.build());

/\*\* Наш сервис выступает одновременно и хранилищем данных \*\*/

routeService.bindRoutes(routes);

/\*\* Вывод остановок одного маршрута \*\*/

System.out.println(

String.format("Поиск по остановке \n\t start : %s \n\t end = %s",from,to)

);

/\*\* Ищем и выводим номер маршрута до точки \*\*/

System.out.println(findRouteToPoint(from,to));

System.out.printf(detectedWorkloadService.execute(routes));

}

public static String findRouteToPoint(String from,String to){

String response = findRoutePathService.findPath(from,to);

if(response == null || response.isEmpty() || response.isBlank()){

findRoutePathService = new FindMinRoutePathService();

response = findRoutePathService.findPath(from,to);

}

findRoutePathService = new FindDirectRoutePath();

return response;

}}

**Результат.**

Нами написана ООП-приложение для хранения в оперативной памяти маршрутов общественного транспорта и поиска по наименованиям 2-х остановок (откуда,куда), и загруженности пути.  
 Приложение маштабируемое по транспорту и алгоритму поиска пути,расчету загруженности