Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №4

Выполнила:

Студент 3 курса

Группы ПО-5

Нерода А.А.

Проверил:

Крощенко А.А.

Брест 2021

# Вариант-10

**Цель работы:** приобрести практические навыки в области объектно- ориентированного проектирования

**Задание 1** Реализовать указанный класс, включив в него вспомогательный внутренний класс или классы. Реализовать 2-3 метода (на выбор). Продемонстрировать использование реализованных классов.

1) Создать класс Notepad (записная книжка) с внутренним классом или классами, с помощью объектов которого могут храниться несколько записей на одну дату.

# Код программы:

package SSP.Lab4.FirstTask;

import java.io.BufferedReader; import java.io.IOException; import java.io.InputStreamReader; import java.util.\*;

public class Solution {

public static void main(String[] args) throws IOException { Notepad notepad = new Notepad();

notepad.show(); notepad.add(); notepad.add(); notepad.show(); notepad.deleteDate(); notepad.show();

}

public static class Notepad{ List<Date> dates;

public Notepad() {

this.dates = new ArrayList<>();

}

public class Date{

List<String> note = new ArrayList<>(); int dd;

int mm; int year;

public Date(int dd, int mm, int year) { this.dd = dd;

this.mm = mm; this.year = year;

}

@Override

public boolean equals(Object o) { if (this == o) return true;

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

return dd == date.dd &&

mm == date.mm && year == date.year;

}

@Override

public int hashCode() {

return Objects.*hash*(dd, mm, year);

}

}

void add() throws IOException {

Date date = dateInitialization("Добавления"); String note = noteInitialization();

Boolean addition = false;

for (int i = 0; i < this.dates.size() && !addition ; i++) { if(date.equals(dates.get(i))) {

dates.get(i).note.add(note); addition = true;

}

}

if (!addition){ date.note.add(note); this.dates.add(date);

}

}

public void deleteDate() throws IOException{

Date delete = dateInitialization("Удаления"); Iterator<Date> iterator = dates.iterator(); while(iterator.hasNext()){

Date date = iterator.next(); if(delete.equals(date))

iterator.remove();

}

}

public void show(){

for (int i = 0; i < this.dates.size(); i++) { System.*out*.printf("Дата

%d/%d/%d\n",dates.get(i).dd,dates.get(i).mm,dates.get(i).year);

for (int j = 0; j < dates.get(i).note.size(); j++) { System.*out*.println((j+1)+". "+dates.get(i).note.get(j));

}

}

}

private Date dateInitialization(String log) throws IOException{ BufferedReader reader = new BufferedReader(new

InputStreamReader(System.*in*));

int dd, mm, year; System.*out*.println("Операция : " + log); System.*out*.println("Введите день ");

dd = Integer.*parseInt*(reader.readLine()); System.*out*.println("Введите месяц ");

mm = Integer.*parseInt*(reader.readLine()); System.*out*.println("Введите год");

year = Integer.*parseInt*(reader.readLine());

Date date = new Date(dd,mm,year); return date;

}

private String noteInitialization() throws IOException { BufferedReader reader = new BufferedReader(new

InputStreamReader(System.*in*));

System.*out*.println("Введите запись"); String note = reader.readLine(); return note;

}

}

}

**Результаты работы:** Операция : Добавления Введите день

26

Введите месяц 11

Введите год 2020

Введите запись Написать отчёт Операция : Добавления Введите день

25

Введите месяц 11

Введите год 2020

Введите запись Реализовать себя Дата 26/11/2020

1. Написать отчёт Дата 25/11/2020

1. Реализовать себя

Операция : Удаления Введите день

25

Введите месяц 11

Введите год 2020

Дата 26/11/2020

1. Написать отчёт

**Задание 2** Реализовать агрегирование. При создании класса агрегируемый класс объявляется как атрибут (локальная переменная, параметр метода). Включить в каждый класс 2-3 метода на выбор. Продемонстрировать использование разработанных классов.

10) Создать класс Планета, используя класс Материк.

# Код программы:

package SSP.Lab4.SecondTask;

import java.util.ArrayList; import java.util.List;

public class Solution {

public static void main(String[] args) { Planet planet = new Planet("Земля"); planet.add("Евразия",

"Северная Америка", "Южная Америка", "Африка", "Австралия", "Антарктида");

planet.showInfo(); System.*out*.println("\nЗамена - Евразия "); planet.changeMainland("Евразия","Брэнт"); planet.showInfo();

}

public static class Planet { String name;

List<Mainland> mainlandList = new ArrayList<>(); public Planet(String name) {

this.name = name;

}

public void showInfo(){ System.*out*.println("Планета: " + this.name); mainlandList.forEach((k)->{

k.showName();

});

}

public void add(String...mainlands){

for (int i = 0; i < mainlands.length; i++) { mainlandList.add(new Mainland(mainlands[i]));

}

}

public void changeMainland(String name, String rename){ for (int i = 0; i < mainlandList.size(); i++) {

if(mainlandList.get(i).name.equals(name))

mainlandList.get(i).changeName(rename);

}

}

private class Mainland { public String name;

public Mainland(String name) { this.name = name;

}

public void showName(){ System.*out*.println(name);

}

public void changeName(String name){ this.name = name;

}

}

}

}

**Результат:** Планета: Земля Евразия

Северная Америка Южная Америка Африка Австралия Антарктида

Замена - Евразия Планета: Земля Брэнт

Северная Америка Южная Америка Африка Австралия Антарктида

# Задание 3

Построить модель программной системы с применением отношений (обобщения, агрегации, ассоциации, реализации) между классами. Задать атрибуты и методы классов. Реализовать (если необходимо) дополнительные классы. Продемонстрировать работу разработанной системы.

# Код программы:

package SSP.Lab4.ThirdTask;

import java.util.ArrayList; import java.util.Arrays; import java.util.List;

public class Solution {

public static void main(String[] args) { Bus bus1 = new Bus(1);

Bus bus2 = new Bus(38); Bus bus3 = new Bus(21);

Trolleybus trolley1 = new Trolleybus(4); Trolleybus trolley2 = new Trolleybus(5); Trolleybus trolley3 = new Trolleybus(228);

Route route = new Route("Больница - Красный двор",4,8); route.add(bus1,bus2,bus3, trolley1, trolley2); route.addAddition(trolley3);

route.start();

}

public static class Route{ String name;

Checker checker = new Checker(); double distance;

int numberOfStops;

double interval = (distance/numberOfStops)\*60;//мин

public Route(String name,double distance, int numberOfStops) { this.name = name;

this.distance = distance; this.numberOfStops = numberOfStops;

}

void add(CityVehicle...vehicles){ for(CityVehicle temp: vehicles){

this.checker.vehicles.add(new VehicleByRoute(temp,distance,numberOfStops));

}

}

void addAddition(CityVehicle...vehicles){ for(CityVehicle temp: vehicles){

this.checker.vehiclesAddition.add(new VehicleByRoute(temp,distance,numberOfStops));

}

}

void start(){

new Thread(checker).start(); synchronized (checker) {

while (!checker.vehicles.isEmpty()) {

new Thread(checker.vehicles.get(0)).start(); checker.vehicles.remove(0);

try { Thread.*sleep*((long)(100\*checker.intervalMultiply));

} catch (InterruptedException e) { e.printStackTrace();

}

}

}

}

private class VehicleByRoute implements Runnable{ CityVehicle vehicle;

double distance; int numberOfStops;

public VehicleByRoute(CityVehicle vehicle, double distance,int numberOfStops) {

this.vehicle = vehicle; this.distance = distance; this.numberOfStops = numberOfStops;

}

@Override

public void run() {

double interval = distance/numberOfStops; startTrip();

for(double d = 0, i = 0; d < distance;d+=interval,i++){ try {

Thread.*sleep*((long)((interval/this.vehicle.speed)\*60\*60) );

} catch (InterruptedException e) { e.printStackTrace();

}

Arrived(i+1);

int accident = (int)(Math.*random*()\*10); if(accident <= 1) {

System.*out*.print("Произошла авария!!!!"); synchronized (vehicle) {

this.vehicle.isAlive = false;

}

break;

}

}

движение");

}

System.*out*.println(vehicle.toString() + " закончил

vehicle.isArrive = true;

private synchronized void startTrip(){ vehicle.start();

№" + num);

}

private synchronized void Arrived(double num) { System.*out*.println(vehicle.toString() + " прибыл на остановку

}

}

public class Checker implements Runnable{ boolean interrupt=false;

double intervalMultiply=1;

List<VehicleByRoute> vehicles = new ArrayList<>(); List<VehicleByRoute> vehiclesAddition = new ArrayList<>();

@Override

public void run() {

List<VehicleByRoute> list = new ArrayList<>(vehicles); while(!interrupt) {

interrupt = true; synchronized (vehicles) {

for (VehicleByRoute vr : list) { if (!vr.vehicle.isAlive) {

if (!vehiclesAddition.isEmpty()) { list.add(vehiclesAddition.get(0)); vehicles.add(0,vehiclesAddition.get(0)); vehiclesAddition.remove(0);

break;

} else

intervalMultiply += 5;

}

if(!vr.vehicle.isArrive) interrupt=false;

}

}

}

}

}

}

public static class Bus extends CityVehicle{

public Bus(int number) { super(number,50); this.name = "Автобус";

}

}

public static class Trolleybus extends CityVehicle{ public Trolleybus(int number) {

super(number,40); this.name = "Троллейбус";

}

}

}

# Класс родитель:

package SSP.Lab4.ThirdTask;

public class CityVehicle{ boolean isAlive;

boolean isArrive = false; double speed;

int number; String name; boolean start(){

if(isAlive) {

System.*out*.println(this.name +" №"+this.number + " начал

движение");

}

else{

System.*out*.println(this.name +" №"+this.number + " поломан");

}

return isAlive;

}

public CityVehicle(int number, int speed) { isAlive = true;

this.number = number; this.speed = speed;

}

@Override

public String toString() {

return name + " №" + number;

}

}

10% - поломки

# Результат:

Автобус №1 начал движение

Автобус №1 прибыл на остановку №1.0 Автобус №1 прибыл на остановку №2.0 Автобус №38 начал движение

Автобус №1 прибыл на остановку №3.0 Автобус №38 прибыл на остановку №1.0 Автобус №1 прибыл на остановку №4.0 Автобус №38 прибыл на остановку №2.0 Автобус №21 начал движение

Автобус №1 прибыл на остановку №5.0 Автобус №38 прибыл на остановку №3.0 Автобус №21 прибыл на остановку №1.0 Автобус №1 прибыл на остановку №6.0 Автобус №38 прибыл на остановку №4.0 Автобус №21 прибыл на остановку №2.0 Автобус №1 прибыл на остановку №7.0 Автобус №38 прибыл на остановку №5.0 Троллейбус №4 начал движение Автобус №21 прибыл на остановку №3.0 Автобус №1 прибыл на остановку №8.0 Автобус №1 закончил движение Автобус №38 прибыл на остановку №6.0

Троллейбус №4 прибыл на остановку №1.0

Автобус №21 прибыл на остановку №4.0 Автобус №38 прибыл на остановку №7.0 Автобус №21 прибыл на остановку №5.0 Автобус №38 прибыл на остановку №8.0 Автобус №38 закончил движение Троллейбус №4 прибыл на остановку №2.0

Произошла авария!!!!Троллейбус №4 закончил движение

Троллейбус №228 начал движение Автобус №21 прибыл на остановку №6.0

Троллейбус №228 прибыл на остановку №1.0 Автобус №21 прибыл на остановку №7.0 Автобус №21 прибыл на остановку №8.0 Троллейбус №228 прибыл на остановку №2.0 Автобус №21 закончил движение Троллейбус №228 прибыл на остановку №3.0

Произошла авария!!!!Троллейбус №228 закончил движение

**Вывод:** в ходе работы приобрёл практические навыки в области объектно- ориентированного проектирования.