Министерство образования и молодежной политики Свердловской области



ГАПОУ СО «Екатеринбургский колледж транспортного строительства»

Отчёт по программе «**Практическая работа 9.2**»

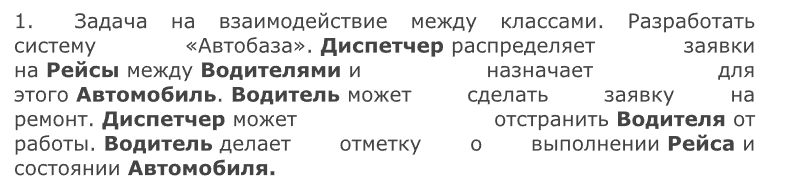
Выполнил: Жеребцев Виталий

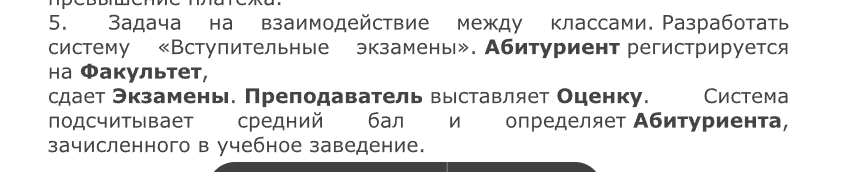
Группа: ПР-24

Преподаватель: Мирошниченко Г.В

2024

**Задание:**

****

****

**Входные данные**

**Name,фио абитурента**

**O1,оценка за 1 экз,int**

**O2,оценка за 2 экз,int**

**Model,модель авто,string**

**Листинг программы**

**Class Abiturent**

package VsypitelExamen  
  
open class Abiturient(name: String, facultet: Facultet): Chelovek(name, facultet){  
 val exam = *mutableListOf*<Exam>()  
  
 fun RegisterExam(ex: Exam) {  
 exam.add(ex)  
 }  
 fun getExams(): List<Exam> {  
 return exam  
 }  
 open fun SredniBall() {  
  
 if (exam.isEmpty())  
 {  
 sredBal = 0.0  
 }  
 else  
 {  
 var sum = 0  
 exam.*forEach* **{** exam **->** sum += exam.bal  
 **}** sredBal = sum.toDouble()/ getExams().size  
 }  
 }  
 fun GetSredBall(): Double {  
 return sredBal;  
 }  
 fun Input()  
 {  
 try {  
 *println*("Введите ваше имя и фамилию")  
 val name = *readLine*()!!.toString()  
 this.name = name  
 val abiturient = Abiturient(name, this.facultet)  
  
 *println*("Введите оценку за первый экзамен")  
 val o1 = *readLine*()!!.*toInt*()  
 val exam1 = Exam("Предмет 1", o1)  
 RegisterExam(exam1)  
  
 *println*("Введите оценку за второй экзамен")  
 val o2 = *readLine*()!!.*toInt*()  
 val exam2 = Exam("Предмет 2", o2)  
 RegisterExam(exam2)  
 facultet.Zachislit(abiturient)  
 }  
 catch (ex:Exception)  
 {  
 *println*("Ошибка: ${ex.message}")  
 }  
 }  
}

**Class chelovek**

package VsypitelExamen  
  
open class Chelovek(var name: String, val facultet: Facultet) {  
 var sredBal = 0.0  
 init {  
 name  
 facultet  
 }  
}

**Class Exam**

package VsypitelExamen  
  
open class Exam (val predmet: String, var bal: Int) {  
  
 fun InfoExams()  
 {  
 *println*("Информация о экзаменах:")  
 *println*("Математика")  
 *println*("Технология")  
 *println*("Информатика")  
 }  
 fun Prepdavatel()  
 {  
 *println*("Состав преподавателей: ")  
 *println*("Математика - Иванов Илья")  
 *println*("Технология - Измайлов Василий")  
 *println*("Информатика - Лазаренко Александр")  
 }  
}

**class Facultet**

package VsypitelExamen  
  
open class Facultet(val name: String)  
{  
 val abiturienti = *mutableListOf*<Abiturient>()  
 fun Zachislit(ab: Abiturient)  
 {  
 abiturienti.add(ab)  
 }  
}

**class Prepodovatel**

package VsypitelExamen  
  
open class Prepdavatel(  
 name: String,  
 facultet: Facultet  
): Chelovek(name,facultet){  
 open fun OperedelitAbutyrienta(ab: Abiturient)  
 {  
 ab.SredniBall()  
 if (ab.sredBal in 4.0..5.0)  
 {  
 *println*("Абитуриент ${ab.name} был зачислен на ${ab.facultet.name}")  
 }  
 else  
 {  
 *println*("Абитуриент ${ab.name} не зачислен на ${ab.facultet.name}")  
 }  
 }  
}

**class Car**

package Autobaza  
  
class Car (var model: String) {  
  
 fun CarInput(car: Car)  
 {  
 *println*("Введите модель машины: ")  
 car.model = *readLine*()!!.toString()  
 }  
  
 fun CarOutput(car: Car) : String  
 {  
 return car.model;  
 }  
}

**class Dispatcher**

package Autobaza  
  
class Dispatcher(name: String) : Person(name)  
{  
 fun NaznachitTrip(voditel: Voditel, car: Car) {  
 val trip = Trip(voditel, car)  
 *println*("Диспетчер $name назначает рейс водителю ${voditel.name}")  
 voditel.OtmetkaTrip(trip)  
 }  
  
 fun OtstranitVoditelya(voditel: Voditel) {  
 *println*("Диспетчер $name отстраняет водителя ${voditel.name}")  
 voditel.otstranen = true  
 }  
}

**class person**

package Autobaza  
  
open class Person(var name: String) {  
  
 init {  
 name  
 }  
  
 fun inputName() {  
 *println*("Введите имя:")  
 name = *readLine*()!!.toString()  
 }  
  
 open fun PersonMakeZaiyavky()  
 {  
 *println*("$name делает заявку")  
 }  
}

**class Repair**

package Autobaza  
  
class Repair(val voditel: Voditel) {  
 fun repairCar() {  
 *println*("${voditel.name} ремонтирует автомобиль")  
 }  
}

**class Trip**

package Autobaza  
  
class Trip(val voditel: Voditel, val car: Car) {  
 var tripname = "Москва - Екатеринбург"  
  
 fun complete() {  
 *println*("Рейс выполнен и автомобиль ${car.model} в хорошем состоянии")  
 }  
  
 fun InfoTrip(v:Voditel, car: Car, d:Dispatcher)  
 {  
 *println*("Информация о поездке: ")  
 *println*("Водитель: ${v.name}")  
 *println*("Модель машины: ${car.model}")  
 *println*("Рейс: ${tripname}")  
 *println*("Диспетчер: ${d.name}")  
 }  
}

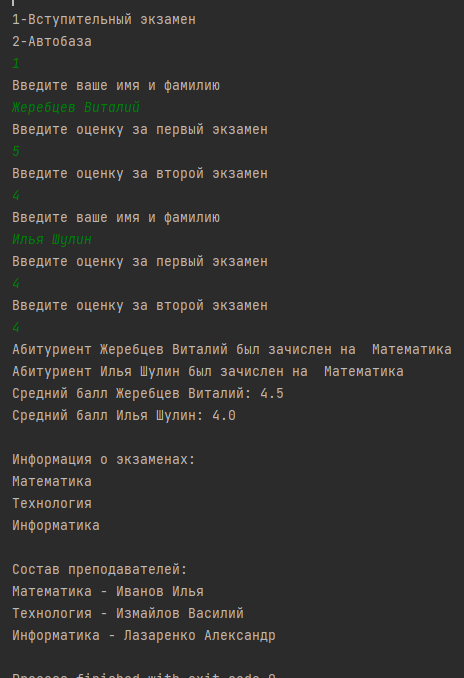
**class Voditel**

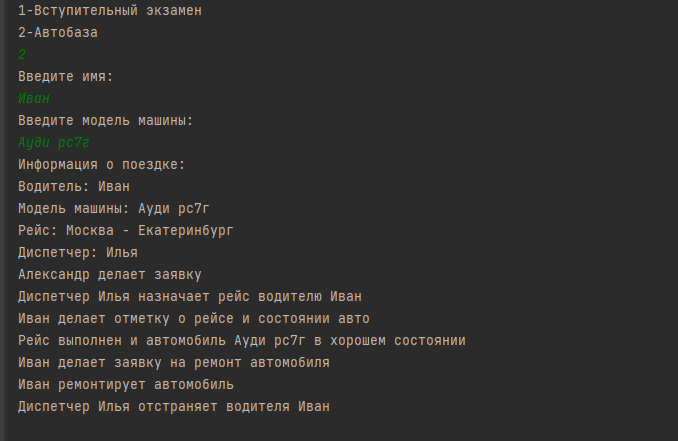
package Autobaza  
  
class Voditel(name: String): Person(name)  
{  
 var otstranen = false  
 fun OtmetkaTrip(trip: Trip) {  
 *println*("$name делает отметку о рейсе и состоянии авто")  
 trip.complete()  
 }  
 fun RemontProsba() {  
 *println*("$name делает заявку на ремонт автомобиля")  
 }

**Main:**

import VsypitelExamen.\*  
import Autobaza.\*  
  
// Вариант 5 - Вступительный экзамен  
// Вариает 1 - Автобаза  
  
fun main() {  
 try {  
 *println*("\n1-Вступительный экзамен\n2-Автобаза")  
 var choose = *readLine*()!!.*toInt*()  
 if (choose == 1)  
 {  
 val facultet = Facultet("Математика")  
 val ch = Chelovek("", facultet)  
 val prepod = Prepdavatel("Лазаренко Александр", facultet)  
 val abiturient1 = Abiturient("",facultet)  
 val abiturient2 = Abiturient("",facultet)  
 var ex = Exam("",0)  
 abiturient1.Input()  
 abiturient2.Input()  
 prepod.OperedelitAbutyrienta(abiturient1)  
 prepod.OperedelitAbutyrienta(abiturient2)  
 *println*("Средний балл ${abiturient1.name}: ${abiturient1.GetSredBall()}")  
 *println*("Средний балл ${abiturient2.name}: ${abiturient2.GetSredBall()}")  
 *println*("")  
 ex.InfoExams()  
 *println*("")  
 ex.Prepdavatel()  
 }  
 else if (choose == 2)  
 {  
 val dispatcher = Dispatcher("Илья")  
 val voditel = Voditel("")  
 val car = Car("")  
 val person = Person("Александр")  
 val trip = Trip(voditel,car)  
 voditel.inputName()  
 car.CarInput(car)  
 Thread.sleep(20000)  
 trip.InfoTrip(voditel,car,dispatcher)  
 Thread.sleep(20000)  
 person.PersonMakeZaiyavky()  
 Thread.sleep(20000)  
 dispatcher.NaznachitTrip(voditel, car)  
 Thread.sleep(20000)  
 voditel.RemontProsba()  
 Thread.sleep(20000)  
 val repair = Repair(voditel)  
 repair.repairCar()  
 Thread.sleep(20000)  
 dispatcher.OtstranitVoditelya(voditel)  
 }  
 else  
 {  
 *println*("Нужно выбрать 1 или 2")  
 System.exit(0)  
 }  
 }  
 catch (ex:Exception)  
 {  
 *println*("Ошибка: ${ex.message}")  
 }  
  
  
  
}

**Тестовые ситуации**

****

****