Министерство образования и молодежной политики Свердловской области



ГАПОУ СО «Екатеринбургский колледж транспортного строительства»

# Отчёт по программе «Классы и объекты в Kotlin. Практическое занятие 8 (задание 1).»

Выполнил: Плоских Александр Павлович

Группа: ПР-21

Преподаватель: Мирошниченко Г.В

2023

**Задача**

****

**Входные и выходные данные**

**Входные**

Class Airlane

var airline-string  
var departurelocation=-string  
var destination-string  
var departureDateTime-Dste  
var flightDuration-string

**Выходные**

String- в зависимости от введенных пользователем данных

**Блок-схема**

**Листинг программы**

**main**

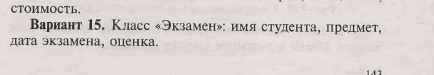
import java.time.LocalDateTime  
fun main() = try{  
var fl=Airplane();  
 fl.Input(fl)  
 *println*("Авиакомпания:${fl.airline}\nМесто отбытия:${fl.departurelocation}\nМесто прыбытия:${fl.destination}")  
 var formattedDateTime=fl.formatDateTime()  
 *println*("Дата и время :$formattedDateTime ")  
 *println*("Была ли у вас задержка самолета? 1-да 2-нет")  
 if(*readln*()!!.*toInt*()==1){  
 *println*("как долго у вас была задержка в часаха")  
 var kk=*readln*()!!.*toLong*()  
 if(kk>0&&kk<30){  
 fl.addHoursToDeparture(kk)  
 }  
 else *println*("sorry")  
 }  
 else *println*("OK")  
 var hdp=fl.calculateArivvalDateTime()  
 *println*("Время прибытия:$hdp")  
}catch (e: Exception){  
 *println*("${e.message}")  
}

**class**

import java.time.LocalDateTime  
import java.time.format.DateTimeFormatter  
import java.time.temporal.ChronoUnit  
class Airplane{  
 var airline=""  
 var departurelocation=""  
 var destination=""  
 var departureDateTime=LocalDateTime.of(1,1,1,1,1)  
 var flightDuration=""  
  
 fun Input(fl:Airplane){  
 *println*("Введите дату вылета")  
 *println*("Введите год")  
 var year= *readln*()!!.*toInt*()  
 if(year>2000&&year<2024){  
 *println*(" месяц")  
 var month= *readln*()!!.*toInt*()  
 if(month>0&&month<13){  
 *println*(" день")  
 var day= *readln*()!!.*toInt*()  
 if(day>0&&day<32){  
 *println*(" час")  
 var hour= *readln*()!!.*toInt*()  
 if(hour>0&&hour<24){  
 *println*(" минуту")  
 var minutes= *readln*()!!.*toInt*()  
 if(minutes>-1&&minutes<60){  
 *println*("Введите компанию которой вы летели")  
 airline=*readln*()!!.toString()  
 *println*("Введите откуда вы вылетали")  
 departurelocation= *readln*()!!.toString()  
 *println*("Введите куда летите")  
 destination=*readln*()!!.toString()  
 *println*("Введите время в полете (пример 7:30)")  
 flightDuration=*readln*()!!.toString()  
 departureDateTime=LocalDateTime.of(year,month,day,hour,minutes)  
  
 }  
 else *println*("sorry")  
 }  
 else *println*("sorry")  
 }  
 else *println*("sorry")  
 }  
 else *println*("sorry")  
 }  
 }  
  
 fun formatDateTime():String{  
 val formatter=DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm")  
 return departureDateTime.format(formatter)  
 }  
  
 fun addHoursToDeparture(hoursToAdd: Long){  
 departureDateTime=departureDateTime.plusHours(hoursToAdd)  
 }  
  
 fun calculateArivvalDateTime():LocalDateTime{  
 var durationArray=flightDuration.*split*(":")  
 val hours=durationArray[0].*toLong*()  
 val minutes=durationArray[1].*toLong*()  
 return departureDateTime.plusHours(hours).plusMinutes(minutes)  
 }  
  
 fun calculateFlightDuration(arrivalDateTime: LocalDateTime):String{  
 val Irrationals=ChronoUnit.*HOURS*.between(departureDateTime,arrivalDateTime)  
 val durationMinutes=ChronoUnit.*MINUTES*.between(departureDateTime,arrivalDateTime)%60  
 return "$Irrationals:${String.*format*("%02d",durationMinutes)}"  
 }  
  
}

**Тестовые ситуации**

**Задача**

****

**Входные и выходные данные**

**Входные**

**Выходные**

**Блок-схема**

**Листинг программы**

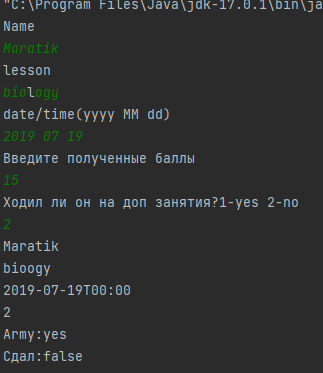
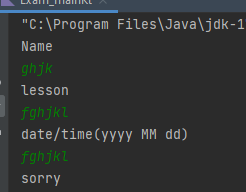
**Main**

fun main(){  
 try{  
 var ex=Exam()  
 ex.Input(ex)  
 ex.improveGrade()  
 ex.Grade\_mm()  
 ex.output()  
 *println*("Сдал:${ex.Passed()}")  
 }catch(e:Exception){  
 *println*("sorry")  
 }  
  
}

**class**

import java.time.LocalDate  
import java.time.LocalDateTime  
import java.time.format.DateTimeFormatter  
  
  
class Exam {  
 var name=""  
 var lesson=""  
 var date= LocalDateTime.of(1,1,1,1,1)  
 var points=0  
 var grade=0  
 var army=""  
  
 fun Input(ex: Exam){  
 *println*("Name")  
 name=*readln*()  
 *println*("lesson")  
 lesson=*readln*()  
 *println*("date/time(yyyy MM dd)")  
 val formatter = DateTimeFormatter.ofPattern("yyyy MM dd")  
 date = LocalDate.parse(*readln*(), formatter).atStartOfDay()  
 *println*("Введите полученные баллы")  
 points= *readln*()!!.*toInt*()  
 }  
 fun Grade\_mm(){  
 when{  
 points>80->grade=5  
 points<=80&&points>60->grade=4  
 points<=60&&points>40->grade=3  
 points<=40->grade=2;  
 }  
  
 }  
  
 fun Passed():Boolean{  
 return points>=60  
 }  
  
 fun improveGrade(){  
 *println*("Ходил ли он на доп занятия?1-yes 2-no")  
 if(*readln*()!!.*toInt*()==1)  
 points+=5  
 }  
  
  
 fun output(){  
 *println*("$name\n$lesson\n$date\n$grade")  
 if(grade==2){  
 army="yes"  
 *println*("Army:$army")  
 }  
 else{  
 army="no"  
 *println*("Army:$army")  
 }  
 }  
}

**Тестовые ситуации**

****