

Реализуйте с использованием ООП простейшую консольную базу данных (без красивого интерфейса) в соответствии со своим вариантом. Функции: добавление, изменение, удаление, сортировка, поиск, вывод на экран содержимого. База данных не предполагает сохранении информации между сеансами работы, но подразумевает, что программа не «падает» при (почти) любых действиях пользователя. Исключение: переполнение памяти. Осуществите распределение ваших классов, объектов и интерфейсов по различным файлам, а файлы сгруппируйте по папкам по какому-либо принципу. Обратите внимание на то, что вам следует разделить ваш проект на Model и View, где Model будет без изменений перенесена в проект на Android, а View содержит логику консольного взаимодействия. В ходе реализации строго используйте следующие принципы: • DRY • SOLID



База данных учета доходов и расходов предпринимателя. Поля: дата, тип операции (доход/расход), объем операции, описание, корреспондент.

import java.util.\*

// Класс для представления операции

data class Operation(

val date: Date,

val type: String, // "доход" или "расход"

val amount: Double,

val description: String,

val correspondent: String

)

// Класс, представляющий базу данных операций

class Database {

private val operations = mutableListOf<Operation>()

// Метод для добавления операции

fun addOperation(operation: Operation) {

operations.add(operation)

}

// Метод для удаления операции по индексу

fun deleteOperation(index: Int) {

if (index >= 0 && index < operations.size) {

operations.removeAt(index)

}

}

// Метод для изменения операции по индексу

fun updateOperation(index: Int, updatedOperation: Operation) {

if (index >= 0 && index < operations.size) {

operations[index] = updatedOperation

}

}

// Метод для сортировки операций по дате

fun sortOperationsByDate() {

operations.sortBy { it.date }

}

// Метод для поиска операций по описанию

fun searchOperationsByDescription(keyword: String): List<Operation> {

return operations.filter { it.description.contains(keyword, ignoreCase = true) }

}

// Метод для вывода всех операций на экран

fun printAllOperations() {

operations.forEachIndexed { index, operation ->

println("Индекс: $index, Дата: ${operation.date}, Тип: ${operation.type}, " +

"Объем: ${operation.amount}, Описание: ${operation.description}, " +

"Корреспондент: ${operation.correspondent}")

}

}

}

fun main() {

val database = Database()

// Пример добавления операций

database.addOperation(Operation(Date(), "доход", 1000.0, "Продажа товаров", "Клиент1"))

database.addOperation(Operation(Date(), "расход", 500.0, "Покупка сырья", "Поставщик1"))

// Вывод всех операций

println("Все операции:")

database.printAllOperations()

// Сортировка операций по дате

database.sortOperationsByDate()

println("\nОперации после сортировки:")

database.printAllOperations ()

// Поиск операций по описанию

val keyword = "продажа"

val searchResults = database.searchOperationsByDescription(keyword)

println("\nРезультаты поиска по описанию '$keyword':")

searchResults.forEachIndexed { index, operation ->

println("Индекс: $index, Дата: ${operation.date}, Тип: ${operation.type}, " +

"Объем: ${operation.amount}, Описание: ${operation.description}, " +

"Корреспондент: ${operation.correspondent}")

}

}