



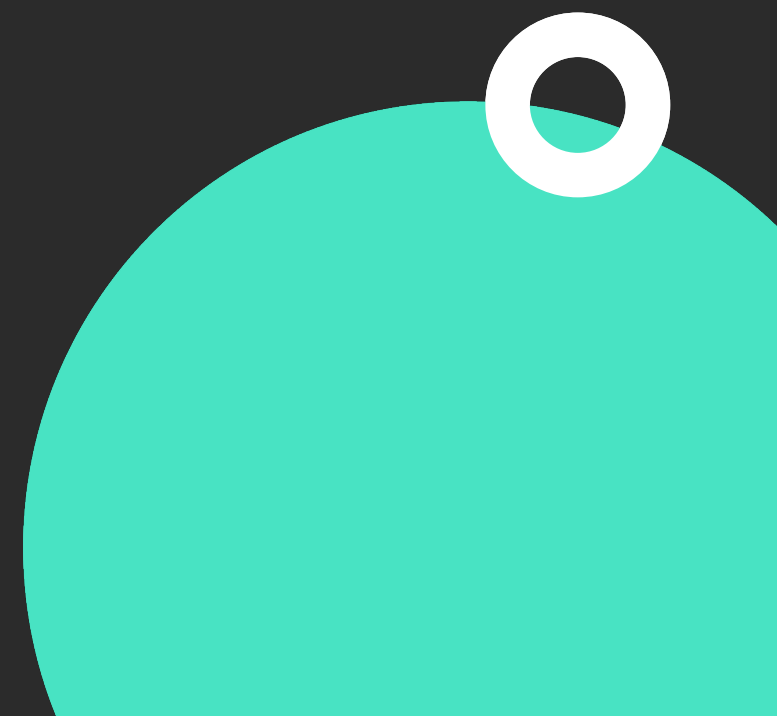
# Defining Relationship Between Objects



Belal Khan

 @probelalkhan

 in/probelalkhan



# Overview

- Learn two approaches to define database relationships: Intermediate data class approach using Embedded objects and multimap return type approach
- Finishing the InvoiceRepository and InvoiceViewModel
- Stitching everything to finish the application



# Code Challenge Recap

- Finishing Customers and Taxes flow.



# Defining Relationship between Objects

- Intermediate Data Class using Embedded Objects
- Multimap return type



# Intermediate Data Class

- Create a new data class that will contain related objects.

```
data class InvoiceWithItems(  
    @Embedded val invoice: Invoice,  
    @Relation(  
        parentColumn = "business_id",  
        entityColumn = "id"  
    )  
    val business: Business,  
    @Relation(  
        parentColumn = "customer_id",  
        entityColumn = "id"  
    )  
    val customer: Customer,  
    @Relation(  
        parentColumn = "tax_id",  
        entityColumn = "id"  
    )  
    val tax: Tax,  
    @Relation(  
        parentColumn = "id",  
        entityColumn = "invoice_id"  
    )  
    val items: List<InvoiceItem>  
) {  
    val invoiceAmount: Double  
        get() = totalAmount + taxAmount  
  
    val taxAmount: Double  
        get() = totalAmount * tax.value / 100  
  
    val totalAmount: Double  
        get() = items.sumOf { it.qty * it.price }  
}
```



## Multimap return type

- We can return the value as a Map.

```
@Query("SELECT * FROM invoices " +  
        "JOIN invoice_items " +  
        "ON invoices.id = invoice_items.invoice_id")  
fun getInvoiceWithItems(): Map<Invoice, List<InvoiceItem>>
```



# InvoiceDao

- CRUD for Invoice Entity
- CRUD for InvoiceItem Entity
- Update Paid Status of an Invoice
- Get Invoices with Items using intermediate data class



# Code Challenge

- Create Invoice Repository.





# Invoice UI Flow



# Summary

- Understood defining relation between objects using Intermediate class and multimap return type
- Created the Invoice DAO
- Finished the Invoice UI Flow





# Testing Room Database

Up Next

