Kotlin ile Modern Programlama Temelleri

Emre Özcan
Native Android Developer

Bugün Neler Öğreneceğiz?

- Kotlin Dili Hakkında
- Temel Yapılar ve Programlama Mantığı
- Fonksiyonlar
- Nesne Yönelimli Programlama (OOP)
- Null Safety ile Güvenli Kodlama
- Scope Functions ile Temiz ve Etkili Kod
- Lambda ve Higher-Order Yapılar
- Extension Functions

Altın Yıldız

Android Developer mülakatlarında karşınıza sıkça çıkan konularda yer alacak!

Kotlin Dili Hakkında

- JetBrains
- JVM
- Google I/O 2017 – Resmi Dil
- Modern Yaklaşım
- Birlikte Çalışabilirlik

Temel Kotlin Yapıları

- val / var / lateinit
- Primitive Types
- if / when
- Loops
- Collections

Fonksiyonlar

- fun
- Parameters
- return
- Default Parameters
- Named Arguments

Nesne Yönelimli Programlama (OOP)

- Real World Modeling
- Properties & Behaviors
- Object Instances
- Reusability

Dört Temel OOP Prensibi

Encapsulation

Inheritance

Polymorphism

Abstraction

Encapsulation

- Protect Internal Data
- Use Access Modifiers (private, protected, internal, public)
- Getter / Setter
- Controlled Access

Encapsulation

Inheritance

- Code Reuse
- Superclass / Subclass
- open and override

Inheritance

Polymorphism

- One Interface, Many Implementations
- Method Overriding
- Behavior Substitution
- override Keyword
- Interface or Superclass Reference

Polymorphism

Abstraction

- Hide Implementation Details
- Define What, Not How
- abstract class
- interface

data class

- Immutable Data Holder
- Auto-generated: equals(), hashCode(), toString()
- copy() Function

enum

- Predefined constant list
- Ideal for roles, types, options
- .name, .ordinal, values()
- Lightweight, no state
- Used in when blocks

sealed class

- Restricted class hierarchy
- Each subclass is known at compile-time
- Safer when expressions (exhaustive)
- Can hold data (like data class)
- Used in UI states, errors, navigation

Hangi durumda hangisi?

Durum / Amaç	enum class	sealed class
Her öğenin aynı davranışa sahip olması	✅	❌
Her öğeye özel veri ve davranış	❌	✅
when ifadesinde güvenli ayrım	⚠️ (eksik)	✅ (exhaustive check)
UI State, API State gibi durumlar	❌	✅

Null Safety ile Güvenli Kodlama

- Nullable Types: String?
- Safe Call: ?.
- Elvis Operator: ?:
- Not-null Assert: !!

Scope Functions ile Temiz ve Etkili Kod

- let – null check & transformation
- apply – configure object
- also – side effects
- run – expression result
- with – context object

Hangi durumda hangisi?

Function	Context	Returns	Use Case
let	it	lambda result	null check, data transform
apply	this	object itself	object config/init (builder)
also	it	object itself	log/side-effect/debug
run	this	lambda result	execute block, return single value
with	this	lambda result	extension on existing object

Lambda ve Higher-Order Yapılar

- Functions as Values
- Lambda Expression { it -> ... }
- Function Types ⟶ (Int) -> String, () -> Unit
- Pass Functions as Parameters

Extension Functions

- Add Function to Existing Type
- Cleaner Utility Methods
- Syntax: Type.functionName()
- Used Everywhere in Android

Teşekkürler!

Emre Özcan

Native Android Developer

Emre Özcan

emre-ozcan

emreozcandev