Fast Track to Kotlin Native Training

Duration: **2 Day**

BİLGİNÇ IT Academy
info@bilginc.com
bilginc.com

Overview

Kotlin is a superb choice for JVM development, and now the recommended language for Android. With Kotlin Native you can bring the power and simplicity of Kotlin to platforms where Virtual Machines are not applicable or desirable. Kotlin Native uses LLVM to produce native instructions that execute directly on the target platform e.g. MacOS, ARM and WebAssembly.

In this course you will learn how to use Kotlin alongside native applications and libraries written in C, C++, Objective-C and Swift. You will learn how to generate Kotlin library wrappers using headers from these other languages. With the generated wrappers you will write Kotlin that easily calls into native API's and makes marshalling data back and forth trivial. You will also learn how to consume Kotlin Native libraries from other environments.

Finally, you will see how this fits into the wider Kotlin ecosystem by creating multi-platform projects that target the JVM, Android, JS and Native platforms.

Prerequisites

Delegates must be confident Kotlin developers. Prior exposure to C/C++ terminology and concepts is very helpful but not essential.

What You Will Learn

- Understand where Kotlin Native fits in the ecosystem
- Use Kotlin to create native apps, libraries and frameworks
- Know which libraries are available on what platforms
- Be comfortable consuming C and Objective-C libraries
- Have confidence manually managing C data type memory
- Create hierarchical Gradle multi-platform projects

Outline

Introduction to Kotlin Native

- Expanding Kotlin beyond the JVM
- Understanding the LLVM Compiler Tools

Fast Track to Kotlin Native Training

Duration: **2 Day**

BİLGİNÇ IT Academy
info@bilginc.com
bilginc.com

- Platforms supported by Kotlin Native
- JVM functionality not available in Native
- Choosing between the CLion and IntelliJ IDEs
- Deconstructing a simple native project

Interoperating with C Libraries

- Working with the provided platform libraries
- Importing headers from external C libraries
- Making sense of Kotlin types for native data
- The role of CVariables, CPointers and CValues
- Exchanging string data between Kotlin and C
- Representing C data structures using CStructs
- Memory management via alloc, free & memScoped
- Invoking Kotlin callacks from C and vice versa
- Using *StableRef* to extend object lifetimes

Interoperating with Objective-C / Swift

- Working with the provided platform libraries
- Importing headers from Objective-C / Swift
- What Swift features are not supported
- How Swift APIs are mapped to Kotlin

Multithreading and Coroutines

- The dangers of shared mutable state
- Constraints on sharing data across threads
- Using *@SharedImmutable* and *@ThreadLocal*
- Initialising the runtime when creating native threads
- Utilising Kotlin's built-in *Worker* type
- Consuming results via *Future* objects
- Freezing data and *DetachedObjectGraph* objects
- Breaking the rules using the *Atomic* types
- Transferring object ownership between threads
- Limits on using coroutines in Kotlin Native

Creating Multiplatform Gradle Projects

- Choosing between the Groovy and Kotlin DSLs
- The Kotlin Multiplatform Gradle Plugin
- Targeting different native platforms
- The default folder naming conventions
- Specifying repos and project dependencies
- Controlling compiler and linker options
- Creating hierarchies and sub-projects