Do it! Kotlin Programming

기본 문법부터 안드로이드 활용까지

Copyright Note

- Book: Do it! 코틀린 프로그래밍 / 이지스퍼블리싱
- Author: 황영덕 (Youngdeok Hwang; sean.ydhwang@gmail.com)
- Update Date: 28-June-2019
- Issue Date: 25-Nov-2017
- Slide Revision #: rev02
- Homepage: acaroom.net
- Distributor: 이지스퍼블리싱 (담당: 박현규)

Education Environments

- Host PC
  - Windows 8~10 64bits (4GB~ memory)
  - MAC OS
- Development Environments
  - IntelliJ IDEA Community
  - Android Studio 3.x
  - SourceTree (Git)
  - Virtual Machine for Android
- Target Board
  - Android based machine or Smartphone

Objectives

- 코틀린 프로그래밍 기본 개념
  - 탄생 배경, 언어의 특징, 실습 환경 구축
- 기본 프로그래밍 문법 습득
  - 변수와 타입, 함수, 흐름 제어, 배열과 문자열, 파일 입출력
- 코틀린의 추가된 기법과 활용
  - 함수형 및 OOP 프로그래밍, 추가된 클래스 기법, 람다, 코루틴, 널 처리
- 안드로이드에서 코틀린 확장 및 사용
  - 코틀린 확장, Anko 확장

Contents (1/2)

첫째마당 - 코틀린 기본 익히기

01 코틀린 시작하기

02 변수와 자료형, 연산자

03 함수와 함수형 프로그래밍

04 프로그램의 흐름 제어

둘째마당 - 코틀린 객체지향 프로그래밍

05 클래스와 객체

06 프로퍼티와 초기화

07 다양한 클래스와 인터페이스

Contents (2/2)

셋째마당 - 코틀린 표준 라이브러리의 활용

08 제네릭과 배열

09 컬렉션

10 표준 함수와 파일 입출력

11 코루틴과 동시성 프로그래밍

넷째마당 - 코틀린과 안드로이드 개발

12 안드로이드와 코틀린

13 코틀린 안드로이드 확장

14 Anko 확장 활용

첫째마당 – 코틀린 기본 익히기

01 코틀린 시작하기
02 변수와 자료형, 연산자
03 함수와 함수형 프로그래밍
04 프로그램의 흐름 제어

01장 – 코틀린 시작하기

01 코틀린 언어 소개

01-1 코틀린의 탄생 배경

01-2 실습 환경 구축

01-3 코틀린 프로젝트 시작하기

Kotlin 탄생 배경

- 목표
  - 풀스택 웹 개발, Android와 iOS앱, 그리고 임베디드, IoT등 모든 개발을 다양한 플랫폼에서 개발할 수 있도록 하는 것.
- 특징
  - IDE(Android Studio의 모체)로 유명한 JetBrains에서 개발하고 보급
  - 코드가 간결하고 다재 다능하며 호환성이 높다.
  - 문장 끝에 세미콜론은 옵션이다.
  - Android Studio에서 안드로이드 공식 언어로 추가
  - 변수는 Nullable(널 값 사용 가능)과 NotNull로 나뉘는데, 변수 선언시 '?'를 붙여 Nullable로 만들 수 있다.

다양한 플랫폼

사용 가능한 플랫폼

- Kotlin/JVM - 자바 가상 머신 상에서 동작하는 앱을 만들 수 있다.
- Kotlin/JS - 자바스크립트에 의해 브라우저에서 동작하는 앱을 만들 수 있다.
- Kotlin/Native - LLVM기반의 네이티브 컴파일을 지원해 여러 타깃의 앱을 만들 수 있다.

Kotlin/Native에서의 타깃

- iOS (arm32, arm64, emulator x86_64)
- MacOS (x86_64)
- Android (arm32, arm64)
- Windows (mingw x86_64)
- Linux (x86_64, arm32, MIPS, MIPS little endian)
- WebAssembly (wasm32)

코틀린의 장점

- 자료형에 대한 오류를 미리 잡을 수 있는 정적 언어
- 널 포인터로 인한 프로그램의 중단을 예방할 수 있다.
  - 보통 개발자들은 코틀린의 이런 특징을 'NPE에서 자유롭다'라고 한다.
  - NPE는 Null Pointer Exception을 줄여 말한 것
- 아주 간결하고 효율적
- 함수형 프로그래밍과 객체 지향 프로그래밍이 모두 가능
- 세미콜론을 생략할 수 있다.

안드로이드의 공식언어

- 자바와 안드로이드 그리고 코틀린
  - 제임스 고슬링 - 자바언어의 제작자
  - 선 마이크로 시스템즈 → 오라클 이후 자바의 부분 유료 정책 고수

- 자바 언어와 Oracle JDK
  - 자바 언어 자체는 무료지만 SDK인 Oracle JDK는
    특정 기능을 위해 라이선스비 지불

- OpenJDK
  - 오픈소스화된 SDK로 Oracle JDK와 거의 동일하나 몇 가지
    상용기능이 빠진 형태로 제공한다.

01 코틀린 언어 소개

01-1 코틀린의 탄생 배경

01-2 실습 환경 구축

01-3 코틀린 프로젝트 시작하기

자바 JDK 설치 하기

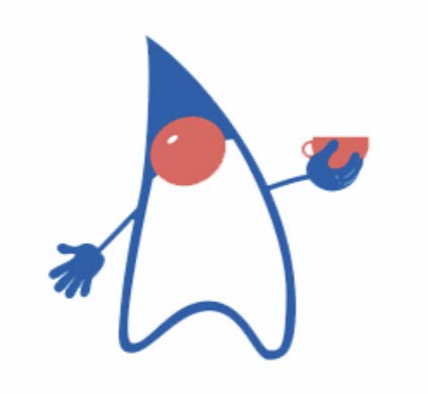
- JDK 설치는 왜?
  - 코틀린을 JVM에서 실행하기 위해
  - 기존 자바와 상호작용할 수 있고 자바 라이브러리를 이용할 수 있다.

- Oracle JDK vs OpenJDK
  - Oracle JDK: 보안 업데이트를 지속적으로 받으려면 '구독' 방식으로 라이선스 구매 필요
  - OpenJDK: 제한 없이 사용 가능. 단 보안 서비스의 의무가 없어 유지보수 어려움

- Azul의 Julu
  - TCK 인증을 통과한 OpenJDK를 묶어서 배포하는 제 3의 벤더

JDK 배포판의 선택

- 원하는 배포판의 선택



OpenJDK

https://openjdk.java.net/

Java ORACLE

https://www.oracle.com/technetwork/java/javase/downloads

Red Hat

AZUL SYSTEMS®

https://www.azul.com/downloads/zulu/

AdoptOpenJDK

01 코틀린 언어 소개

01-1 코틀린의 탄생 배경

01-2 실습 환경 구축

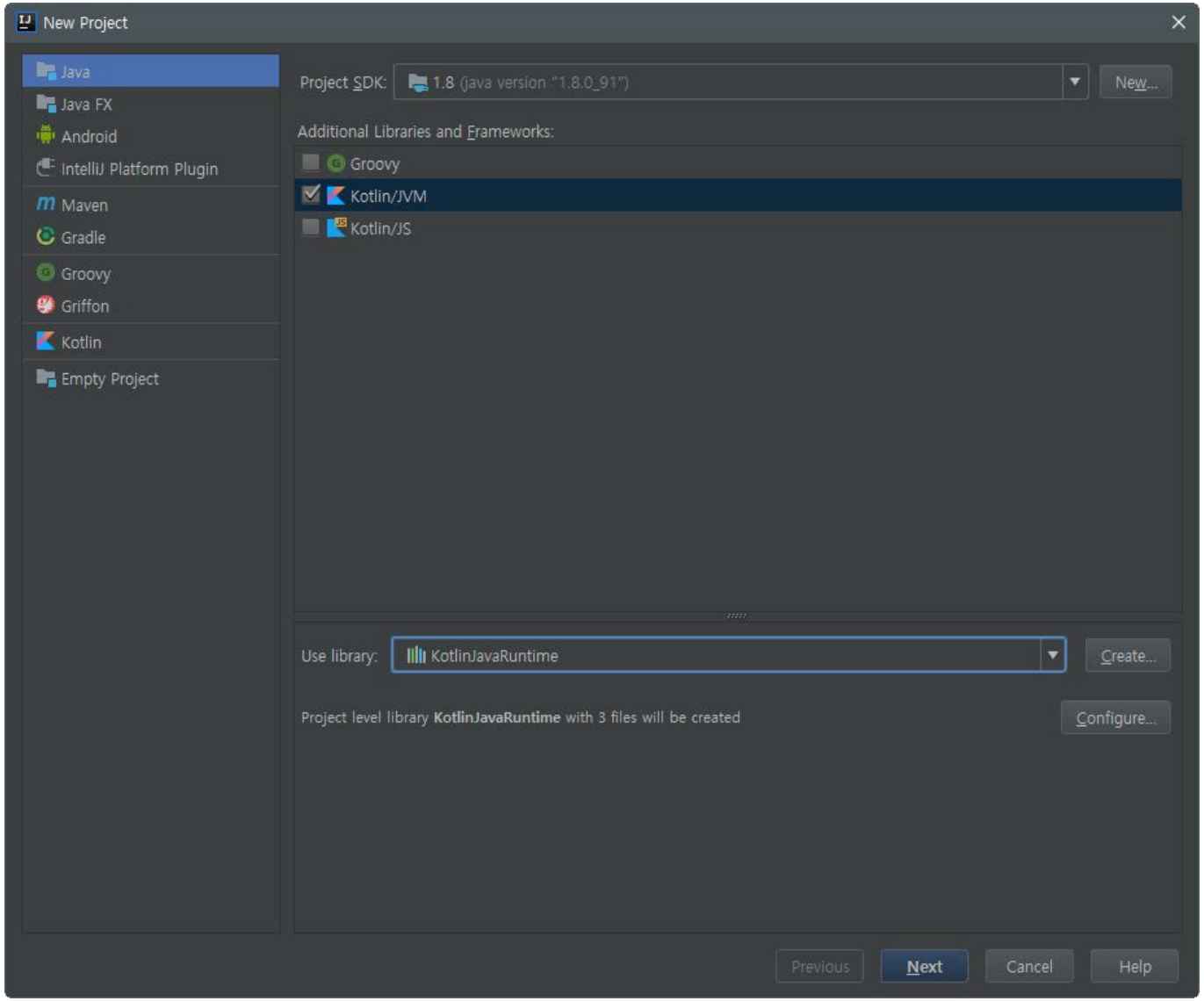
01-3 코틀린 프로젝트 시작하기

IntelliJ

- IntelliJ
  - jetbrains에서 만든 도구로 코틀린언어와 궁합이 잘 맞는다.
  - 개인적인 사용에서 community 버전을 무료로 사용할 수 있다.
    - https://www.jetbrains.com/idea/download

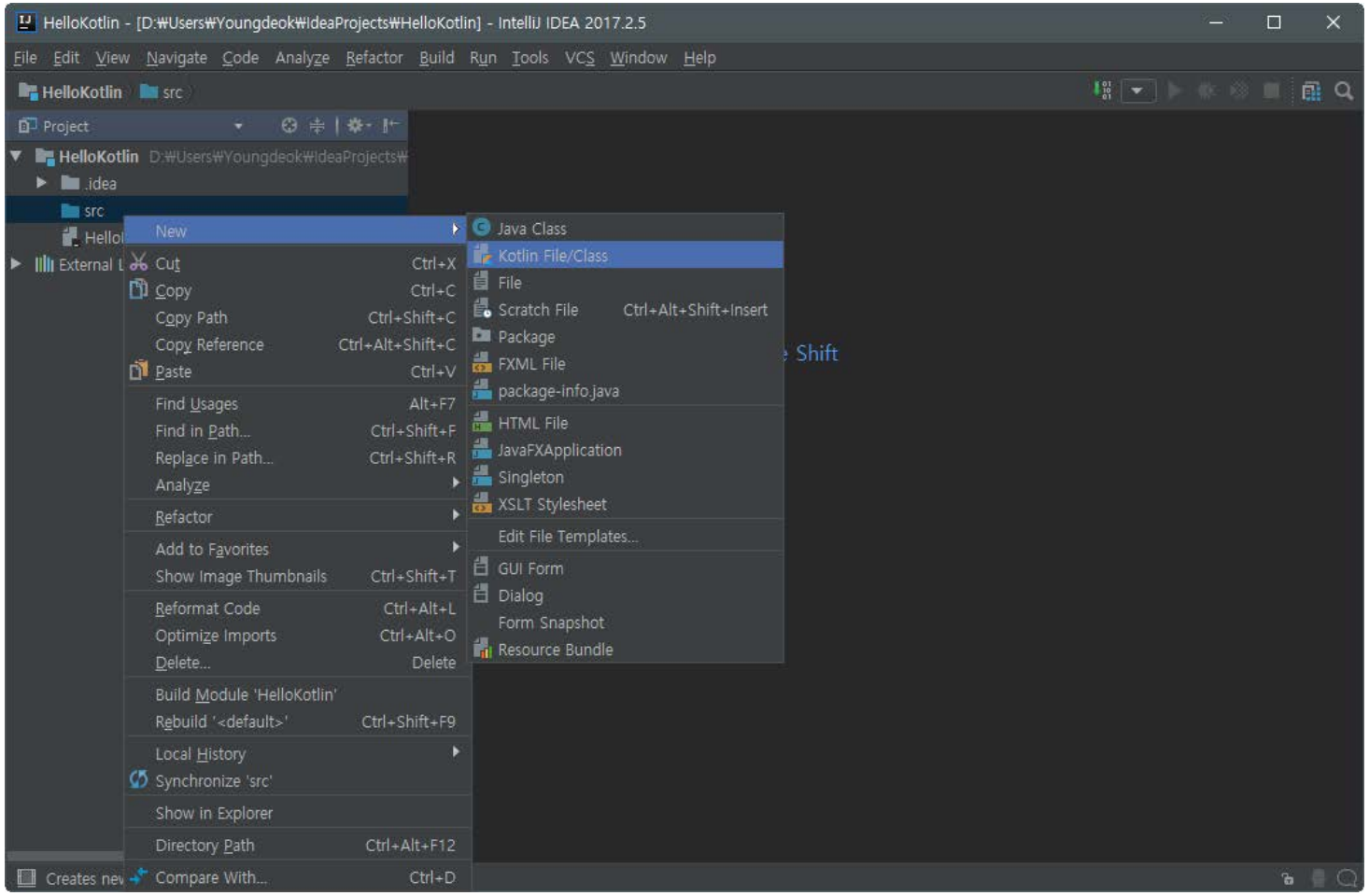
Create a new project

- New Project
  - Java 선택
  - SDK 선택
    - Java 1.6 이상 선택
  - Use library 선택
  - Porject Name
    - HelloKotlin



Create a new project

- src에서 Kotlin 추가
  - Win: Alt + Insert키 혹은 마우스 우측 메뉴에서 선택
  - Kotlin File/Class
    - app 입력 후 Enter



Create a new project

- Coding
  - main을 타이핑하고 tab키를 눌러보자.

```
fun main() {

}
```

  - main 함수가 코드 어시스트에 의해 자동 완성 된다.

```
fun main() {
    println("Hello Kotlin!")
}
```

- 실행 단축키(Win)
  - Ctrl + Shift + F10

Hello Kotlin 분석

- 분석
  - class가 없는데 main 메서드 하나로 println을 콘솔에 실행하고 있다.
  - 실제로는 main 메서드는 파일명을 기준으로 자동으로 클래스가 생성된다.
    - Tools > Kotlin > Show Kotlin Bytecode 에서 Decompile을 해본다.

```
public final class HelloKt {
   public static final void main(@NotNull String[] args) {
      Intrinsics.checkParameterIsNotNull(args, "args");
      String var1 = "Hello Kotlin";
      System.out.println(var1);
   }
}
```

main( )에서 매개변수를 사용하는 경우

- 코드 보조에서 maina로 선택하는 경우
  - main(args: Array<String>)

```
fun main(args: Array<String>) {
    println(args[0]) // 외부의 첫번째 인자
    println(args[1]) // 외부의 두번째 인자
    println(args[2]) // 외부의 세번째 인자
}
```

  - 프로그램 외부에서 인자를 받아들인다.

HelloKotlin의 분석

- main()은 최상위 함수로 실행 진입점이다.
  - 자바와 같은 객체 지향 언어는 프로그램을 실행하기 위해 클래스와 그 안에 main( )필요
  - 코틀린은 클래스 없이 main( )함수 하나로 실행 가능
- Decompile된 HelloKotlin.kt 소스

```
public final class HelloKotlinKt {
   public static final void main() {
      String var0 = "Hello Kotlin!";
      boolean var1 = false;
      System.out.println(var0);
   }

   // $FF: synthetic method
   public static void main(String[] var0) {
      main();
   }
}
```

동적 메모리 영역

- 프로그램의 일반적인 메모리 영역