Uniwersytet Wrocławski

Rafał Lewandków

PROGRAMOWANIE URZĄDZEŃ MOBILNYCH 1

WYKŁAD 0

- PODSTAWOWE INFORMACJE
- TREŚCI PROGRAMOWE
- ZASADY ZALICZENIA

PODSTAWOWE INFORMACJE

Rafał Lewandków
pokój 075
rafal.lewandkow2@uwr.edu.pl

Forma zajęć i liczba godzin:
Wykład 15 godz. /Laboratorium 30 godz.
Materiały do zajęć: https://github.com/RafLew84/ProgUM

Literatura obowiązkowa i zalecana:

- https://kotlinlang.org/docs/home.html
- https://developer.android.com/courses

Nakład pracy studenta:
Praca własna studenta: 30 godz.
Łączna liczba godzin 75
Liczba punktów ECTS: 3

Uniwersytet Wrocławski

ZASADY ZALICZENIA

Warunkiem zaliczenia laboratorium jest uzyskanie pozytywnej oceny z list zadań.

- Przewidzianych jest **6 list zadań**.
- Z każdej listy wystawiana jest **osobna ocena**.
- Dopuszczalne jest **nieoddanie** lub **niezaliczenie** jednej listy. Za tą listę student otrzymuje **ocenę 2.0**.
- Każda lista ma **określony termin zwrotu** i **punktację**. Za każdy tydzień opóźnienia ocena **obniżana jest o 1**.
- Listy oddawane są **podczas** zajęć laboratoryjnych.
- Do list zadań dołączana jest lista pytań, na które student jest **zobowiązany odpowiedzieć ustnie** podczas zajęć laboratoryjnych. Z każdej listy pytań (o ile istnieje) losowane są pytania. Liczba pytań jest zależna od liczby punktów na liście. Przykładowo, jeśli lista ma 10 punktów, a z zadań praktycznych student może otrzymać 5 punktów, losowane jest 5 pytań. Pytania dotyczą zarówno teorii jak i praktyki.

Uniwersytet Wrocławski

ZASADY ZALICZENIA

Warunkiem zaliczenia laboratorium jest uzyskanie pozytywnej oceny z list zadań.

- Student jest **zobowiązany do przekazania kodu źródłowego** w repozytorium GitHub. Link do repozytorium należy podać jako odpowiedź na **zadanie na platformie MS Teams**. Brak linku do repozytorium z rozwiązaniami zadań jest **równoznaczny z brakiem zaliczenia przedmiotu**.
- Ocena końcowa jest **średnią arytmetyczną** ocen z list
  - Ocena 3,0 – średnia 3,0 – 3,24
  - Ocena 3,5 – średnia 3,25 – 3,74
  - Ocena 4,0 – średnia 3,75 – 4,24
  - Ocena 4,5 – średnia 4,25 – 4,74
  - Ocena 5,0 – średnia 4,75 – 5,0
- Dopuszczalne są **trzy nieobecności** nieusprawiedliwione na zajęciach laboratoryjnych. Przy czwartej nieobecności automatycznie wstawiany jest status **nieklasyfikowany.**
- Oceny i punktacja jest **dostępna na bieżąco** w pliku na platformie MS Teams.

Uniwersytet Wrocławski

Treści Programowe

0. Wstęp
1. Typy danych, Wyrażenia, Instrukcje, Pętle
2. Garbage Collection
3. Funkcje
4. Kolekcje
5. Klasy
6. Obiekty i Interfejsy
7. Inicjalizacja i Delegacja
8. Fundamenty Aplikacji. Aktywność, Cykl Życia. Jetpack Compose
9. Podstawy Tworzanie UI, Kompozycja, Rekompozycja, Stan
10. Elementy Struktury UI, Scaffold, Obsługa Kolekcji
11. Nawigacja w Aplikacji. Compose Navigation, Tab Navigation, Drawer
12. Wzorce Projektowe – Strukturalne i Kreacyjne
13. Wzorce Projektowe – Behawioralne
14. Widgety

Kotlin – Język Przemysłowy

Kotlin (Introduced 2011, Version 1.0: 2016)

- FORTRAN: FORmula TRANslation (1957)
- LISP: LISt Processor (1958)
- ALGOL: ALGOrithmic Language (1958)
- COBOL: COmmon Business-Oriented Language (1959)
- BASIC: Beginners' All-purpose Symbolic Instruction Code (1964)
- Simula 67, the Original Object-Oriented Language (1967)
- Pascal (1970)
- C (1972)
- Smalltalk (1972)
- C++: A Better C with Objects (1983)
- Python: (1990)
- Haskell: Pure Functional Programming (1990)
- Java: Virtual Machines and Garbage Collection (1995)
- JavaScript: (1995)
- C#: (2000)
- Scala: (2003)
- Groovy: (2007)

JVM, JRE, JDK

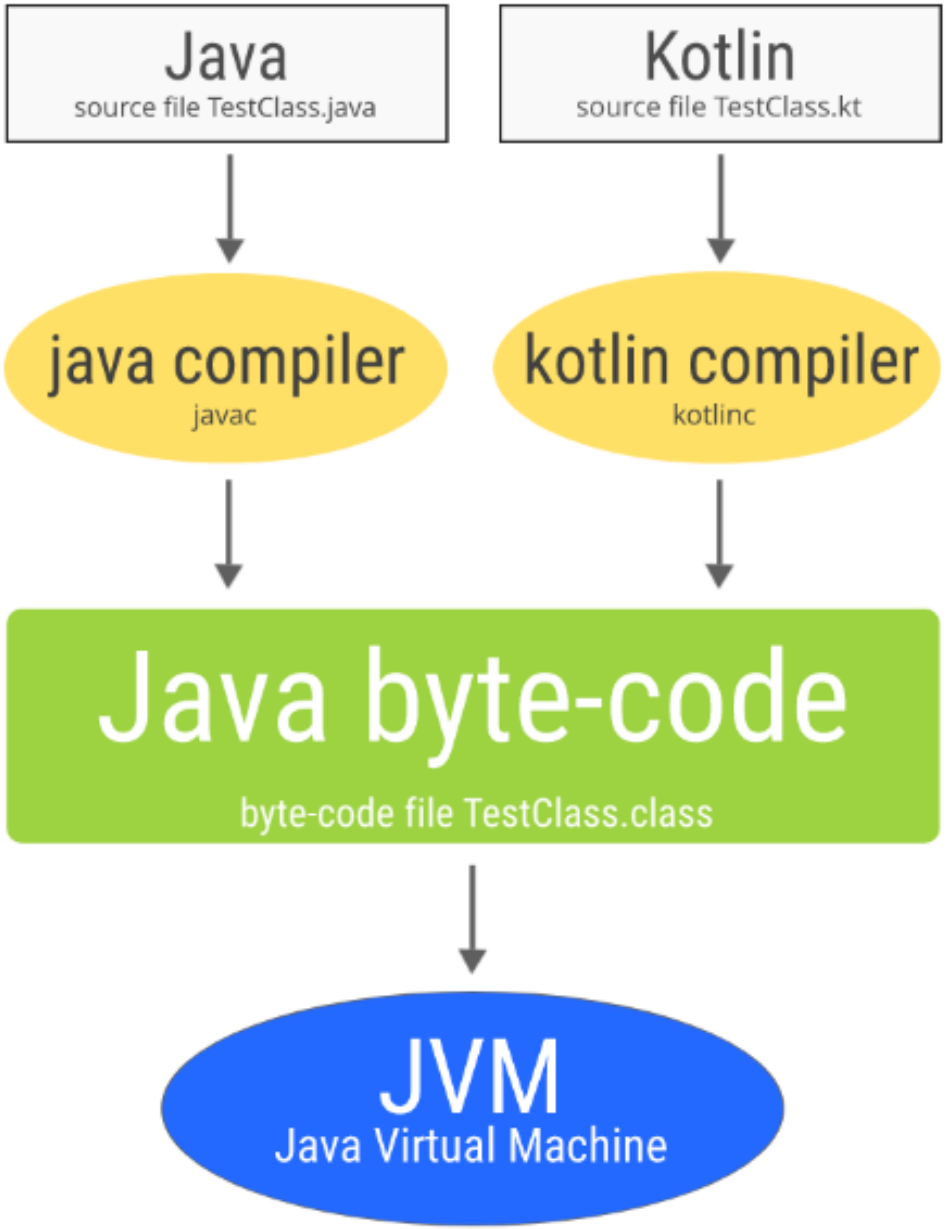
JDK (**Java Development Kit**) - Pakiet Programisty Javy. JDK zawiera Środowisko Uruchomieniowe Javy (tzn. JRE) oraz zestaw narzędzi niezbędnych do wytwarzania oraz kompilowania oprogramowania tworzonego w języku JAVA.

JRE (**Java Runtime Environment**) - Środowisko Uruchomieniowe Javy. W skład JRE wchodzi Wirtualna Maszyna Javy (JVM) + zbiór klas oraz narzędzi wymaganych do uruchomienia aplikacji wytworzonych w języku JAVA.

JVM (Java Virtual Machine) - Wirtualna Maszyna Javy. Środowisko zdolne do wykonywania skompilowanego kodu aplikacji (kod bajtowy Javy).

Uniwersytet Wrocławski

Interoperacyjność



Kotlin

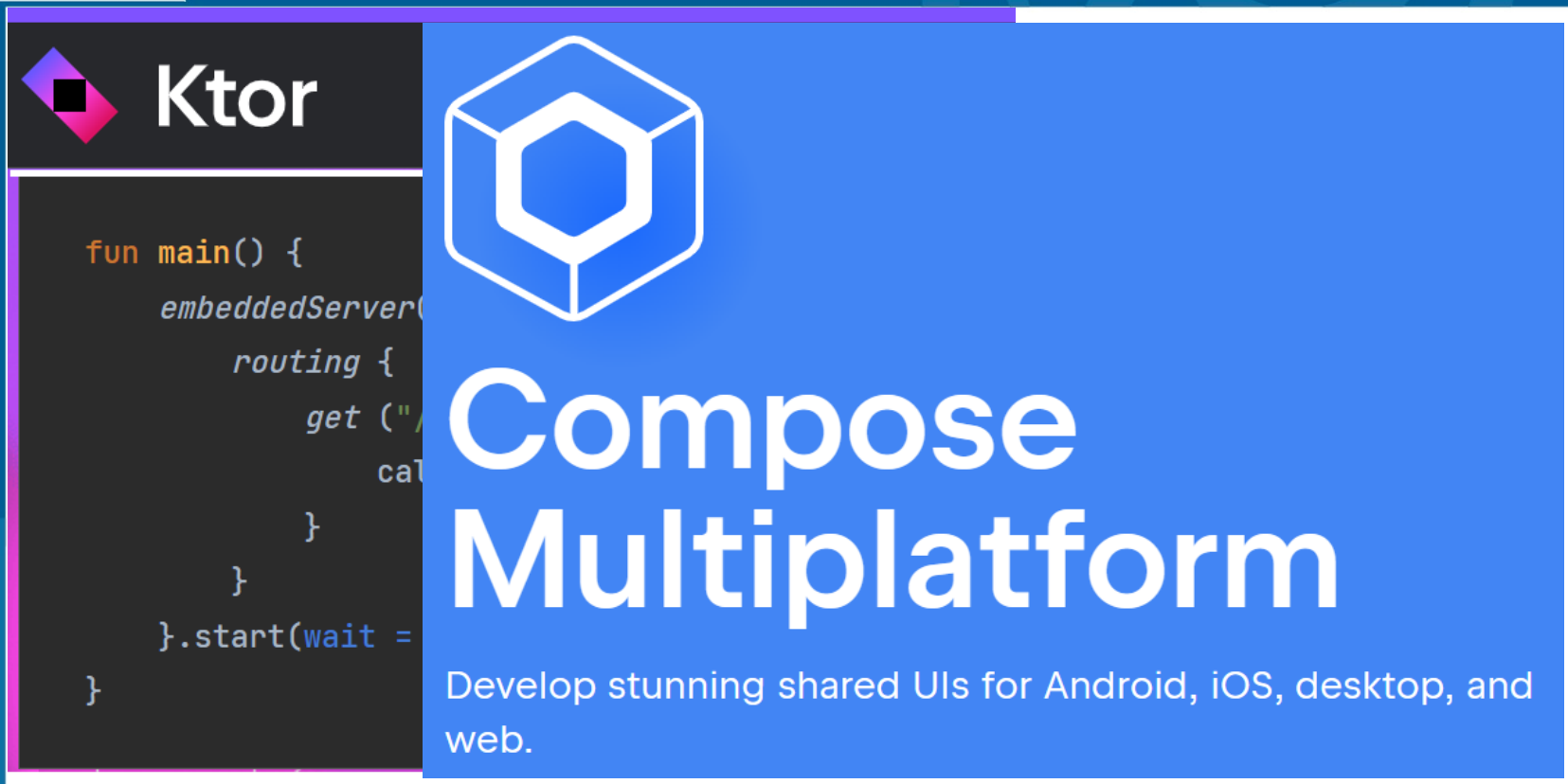
- **Statyczne typowanie danych** – na etapie kompilacji typy wyrażeń są znane a kompilator sprawdza czy istnieją pola i metody w obiektach do których odwołujemy się w kodzie
- **Domniemanie typów** – typ danych jest określany przez kompilator na podstawie kontekstu
- Typy *nullowalne* - **NullPointerException**
- **Programowanie Funkcyjne**
- **OOP**
- **Można łączyć programowanie obiektowe i funkcyjne**

Kotlin

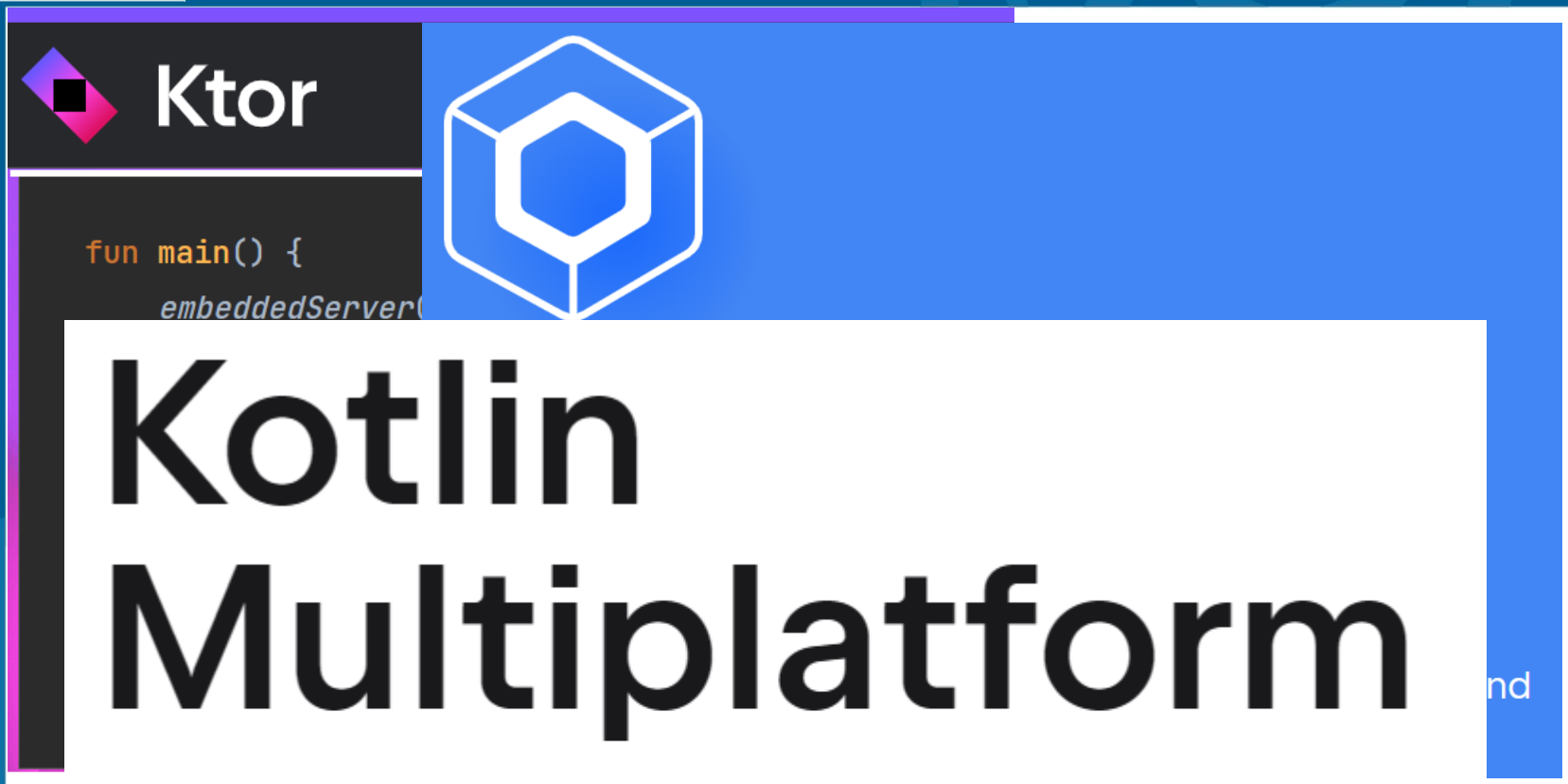
Ktor

```
fun main() {
    embeddedServer(Netty, port = 8000) {
        routing {
            get ("/") {
                call.respondText("Hello, world!")
            }
        }
    }.start(wait = true)
}
```

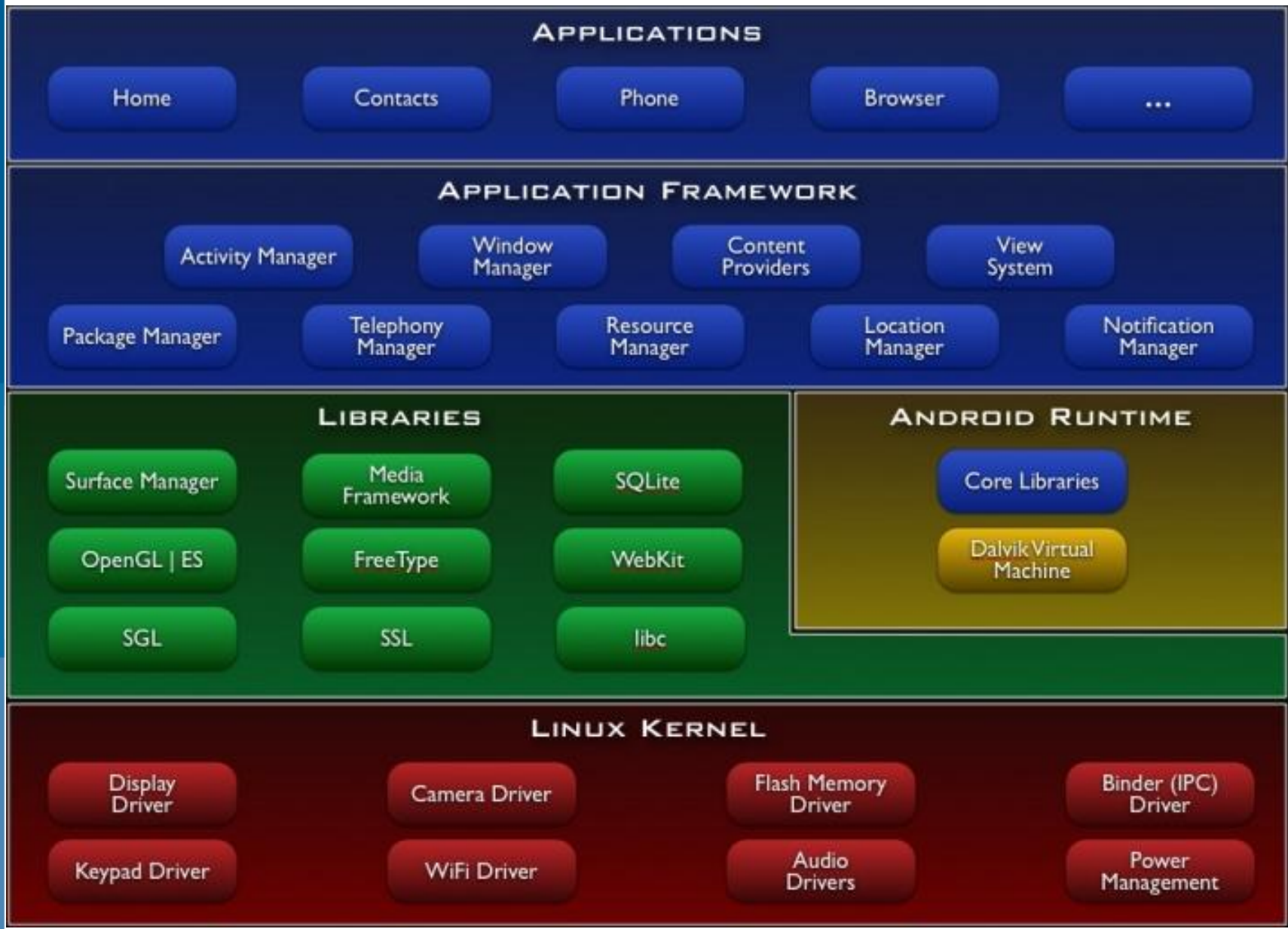
Kotlin


Ktor
fun main() {
embeddedServer(
routing {
get ("/
cal
}
}
}.start(wait =
}
Compose Multiplatform
Develop stunning shared UIs for Android, iOS, desktop, and web.

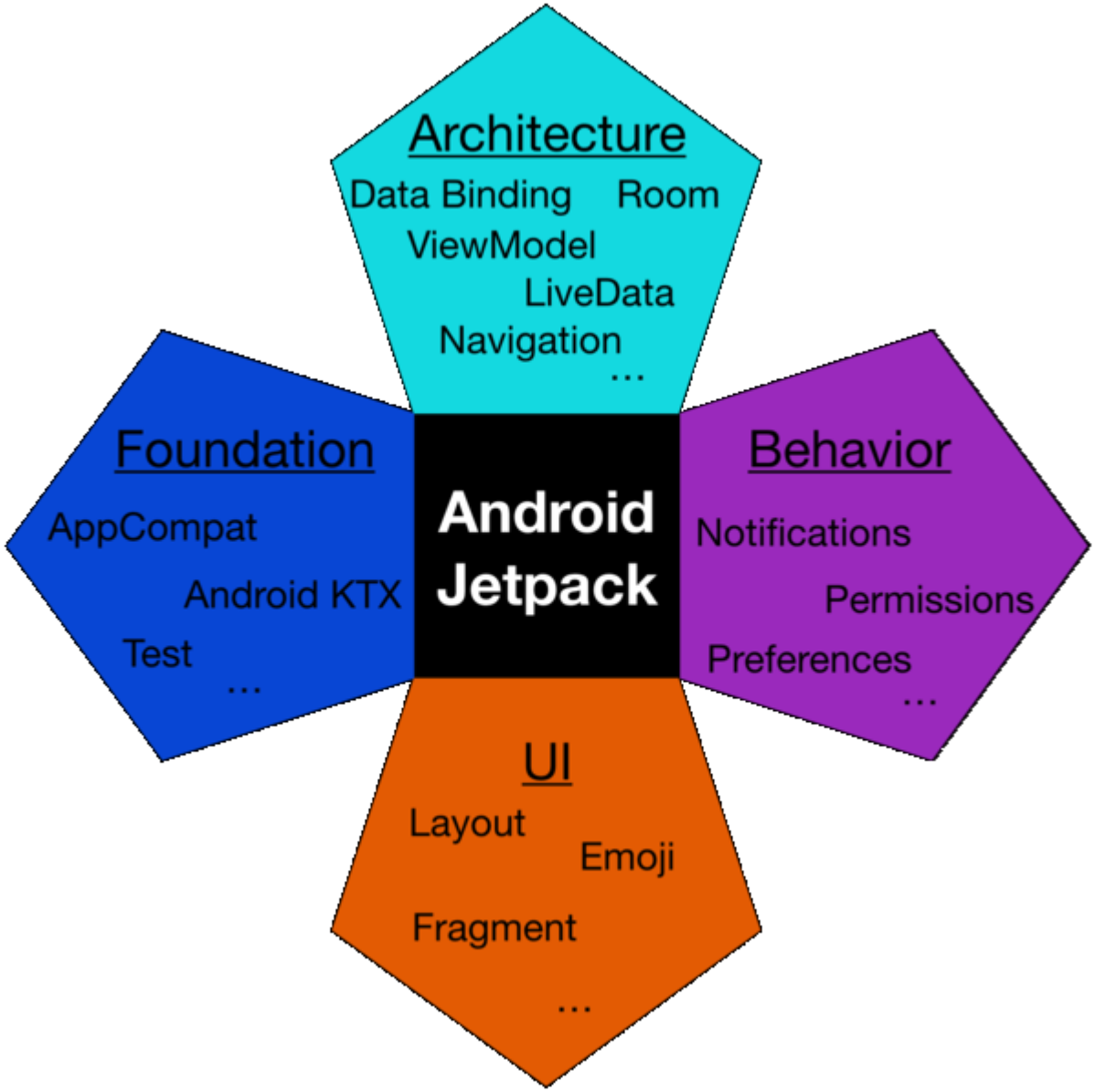
Kotlin


Ktor
fun main() {
embeddedServer(
Kotlin
Multiplatform
Share code on your terms
Reuse Kotlin code across Android, iOS, web, desktop, and
server-side while keeping native code if needed.
nd

Android


APPLICATIONS
Home
Contacts
Phone
Browser
...
APPLICATION FRAMEWORK
Activity Manager
Window Manager
Content Providers
View System
Package Manager
Telephony Manager
Resource Manager
Location Manager
Notification Manager
LIBRARIES
Surface Manager
Media Framework
SQLite
OpenGL | ES
FreeType
WebKit
SGL
SSL
libc
ANDROID RUNTIME
Core Libraries
Dalvik Virtual Machine
LINUX KERNEL
Display Driver
Camera Driver
Flash Memory Driver
Binder (IPC) Driver
Keypad Driver
WiFi Driver
Audio Drivers
Power Management

Android


Architecture
Data Binding
Room
ViewModel
LiveData
Navigation
...
Foundation
AppCompat
Android KTX
Test
...
Android Jetpack
Behavior
Notifications
Permissions
Preferences
...
UI
Layout
Emoji
Fragment
...

Android Studio