



# PROGRAMOWANIE URZĄDZEŃ MOBILNYCH 1

## WYKŁAD 0

- PODSTAWOWE INFORMACJE
- TREŚCI PROGRAMOWE
- ZASADY ZALICZENIA

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

Warunkiem zaliczenia laboratorium jest uzyskanie pozytywnej oceny z list zadań.

- Na zajęcia przewidzianych jest **7 list zadań**.
- Z każdej listy wystawiana jest **osobna ocena**.
- **Nie jest konieczne** zaliczenie wszystkich list aby otrzymać ocenę pozytywną z laboratorium.
- Każda lista posiada informację o **liczbie punktów** wymaganych na konkretną ocenę
- Każda lista posiada **termin zwrotu**.
- Za każdy tydzień opóźnienia otrzymana ocena jest **obniżana o 1**.
- Listy oddawane są **podczas zajęć** laboratoryjnych.
- Do każdego zadania prowadzący **może zadać dodatkowe pytania**.
- Liczba punktów za listę jest przyznawana na podstawie **poprawności wykonania zadań oraz odpowiedzi ustnej**.
- Ocena końcowa jest **średnią arytmetyczną ze wszystkich ocen** z list.
- Na ocenę 3,0 **wymagana jest średnia co najmniej 3,0**.

- 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 Tworzenie UI, Kompozycja, Rekompozycja, Stan**
- 10. Elementy Struktury UI, Scaffold, Obsługa Kolekcji**
- 11. Nawigacja w Aplikacji. Compose Navigation, Tab Navigation, Drawer**
- 12. Wzorce Projektowe**
- 13. Powiadomienia i Upoważnienia w Aplikacji**

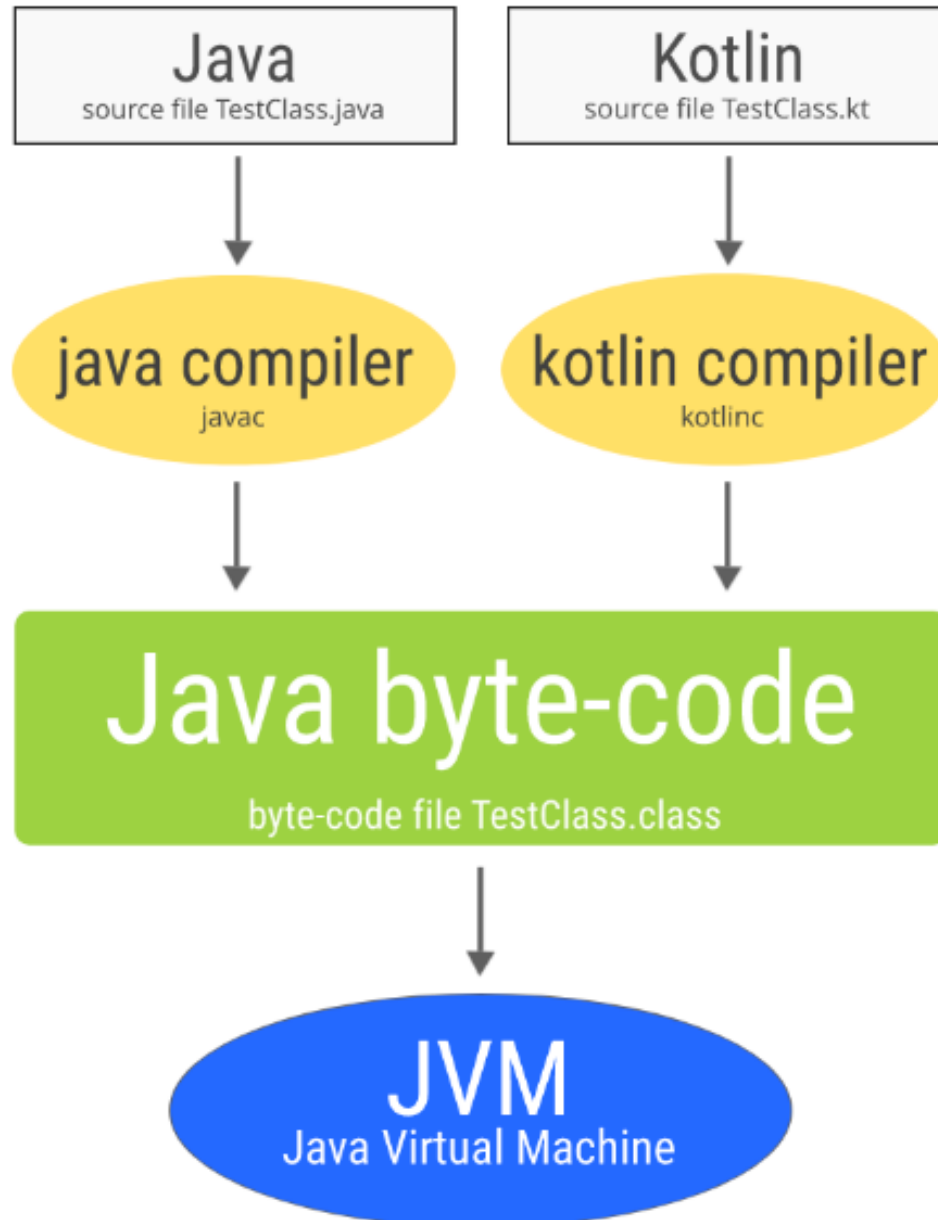
## Kotlin (Introduced 2011, Version 1.0: 2016)

- **FORTTRAN: 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)**

**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).



- **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**





## Ktor

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



Ktor

```
fun main() {  
    embeddedServer(  
        routing {  
            get ("/  
            cat  
        }  
    }.start(wait =  
}
```



# Compose Multiplatform

Develop stunning shared UIs for Android, iOS, desktop, and web.



Ktor

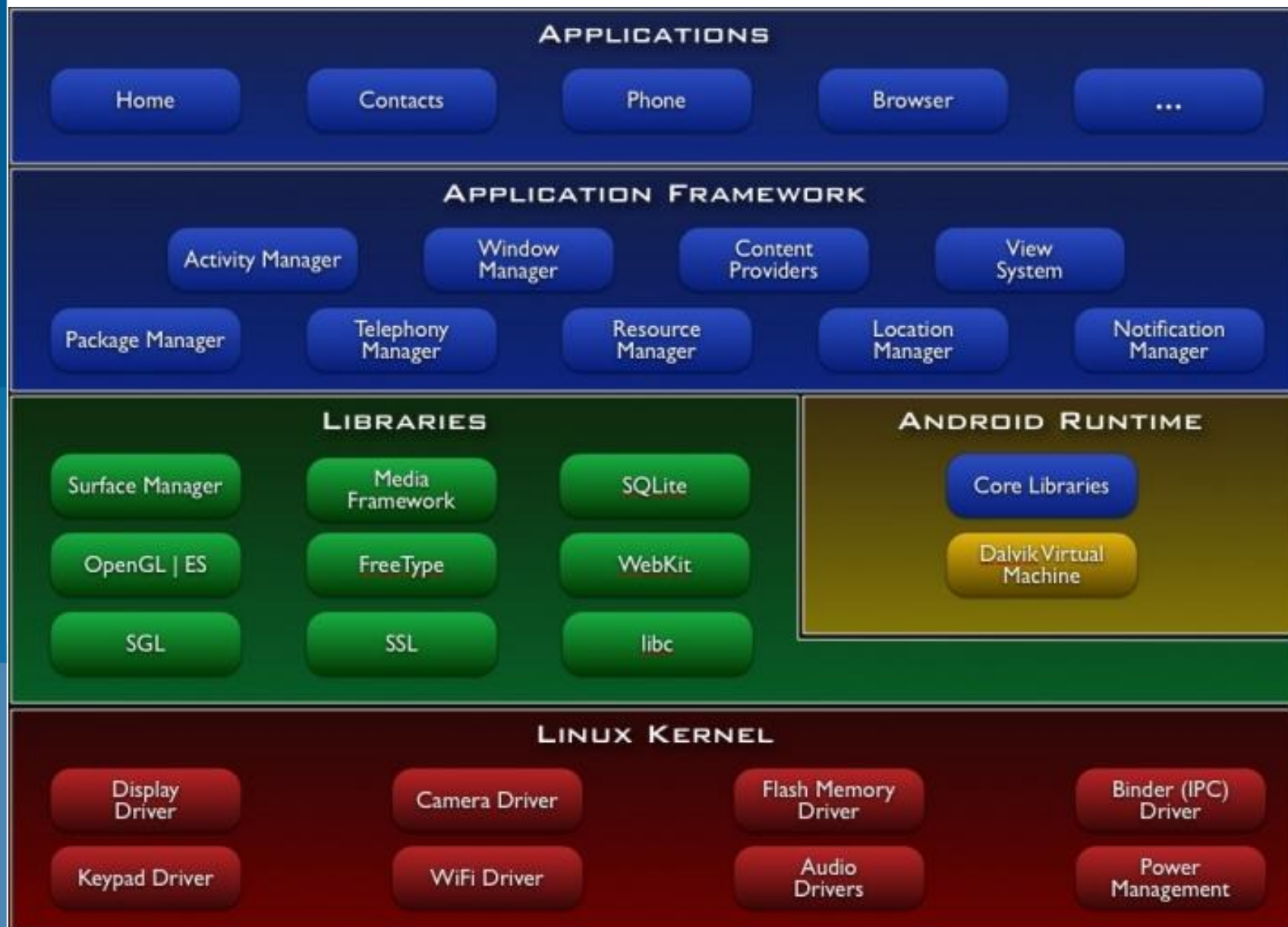


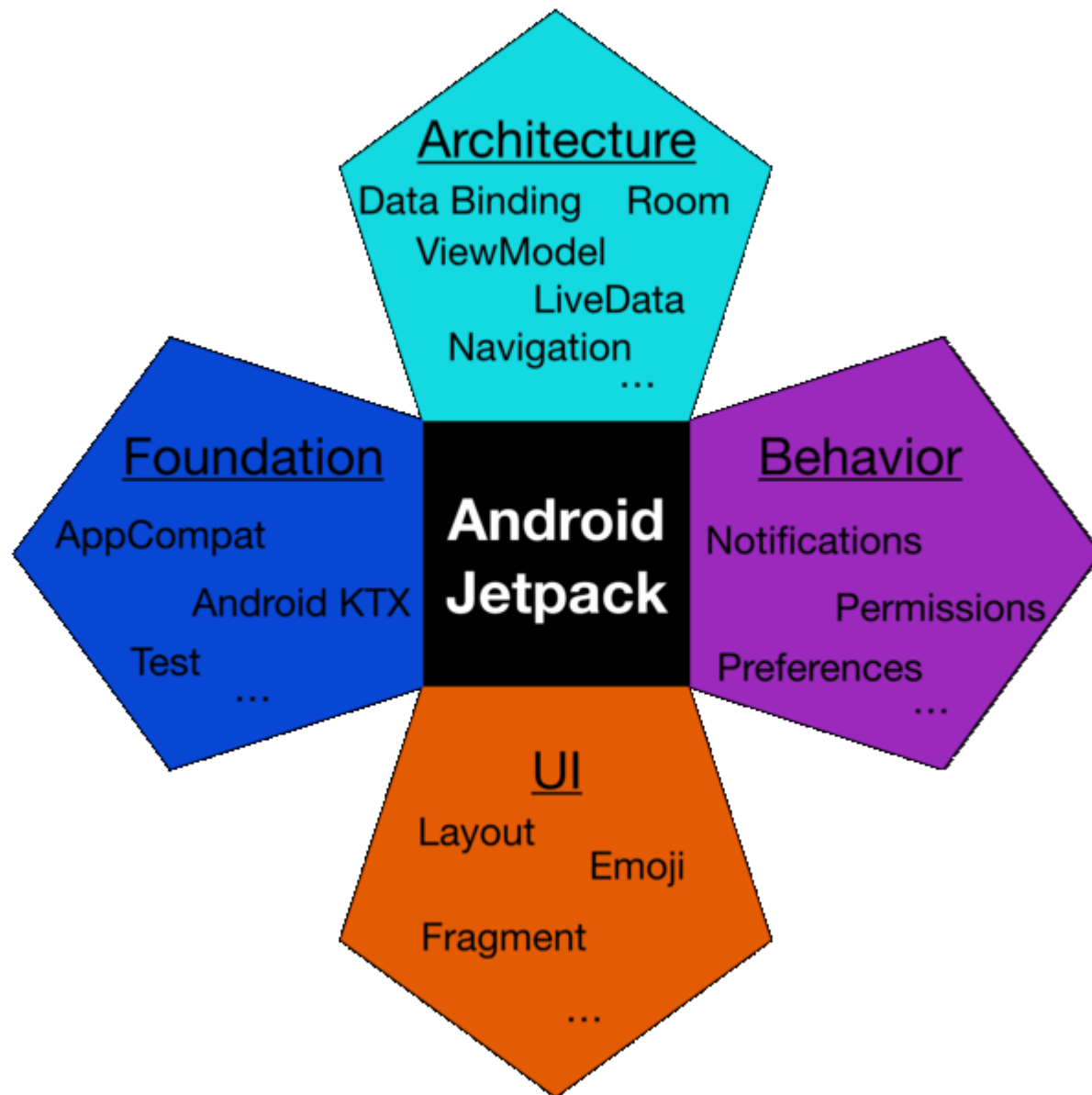
```
fun main() {  
    embeddedServer()
```

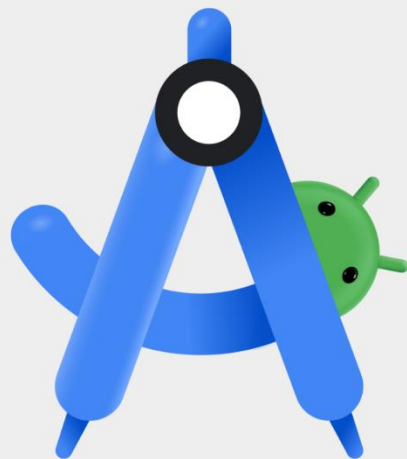
# Kotlin Multiplatform

Share code on your terms

Reuse Kotlin code across Android, iOS, web, desktop, and server-side while keeping native code if needed.







# Android Studio