**PRAKTIKUM PEMROGRAMAN MOBILE**

**Kotlin Dasar**

**Pertemuan Ke-2**

**Disusun Oleh :**

**NAMA : Safrin Suleman**

**NIM : 195410099**

**JURUSAN : Teknik Iformatika**

**JENJANG : S1**

**MODUL 2**

**Kotlin Dasar**

1. **TUJUAN**

Mahasiswa mampu memahami dan mengembangkan aplikasi sederhana dengan bahasa pemrograman Kotlin.

1. **DASAR TEORI**

Penggunaan Kotlin untuk Pengembangan Android. Kotlin/Native memungkinkan developer untuk menggunakannya sebagai bahasa pemrograman dalam pengembangan aplikasi di platform lain seperti embedded system, desktop, macOS, dan iOS. Bahkan tak menutup kemungkinan Kotlin juga bisa digunakan untuk data science dan machine learning. Kotlin sangat cocok untuk mengembangkan aplikasi Android, membawa semua keunggulan bahasa modern ke platform Android tanpa memperkenalkan batasan baru:

1. Compatibility. Kotlin sepenuhnya kompatibel dengan JDK 6. Ini memastikan bahwa aplikasi yang dibangun dengan Kotlin dapat berjalan pada perangkat Android yang lebih lama tanpa ada masalah. Android Studio pun mendukung penuh pengembangan dengan bahasa Kotlin.
2. Performance. Dengan struktur bytecode yang sama dengan Java, aplikasi yang dibangun dengan Kotlin dapat berjalan setara dengan aplikasi yang dibangun dengan Java. Terdapat juga fitur seperti inline function pada Kotlin yang membuat 19 kode yang dituliskan dengan lambda bisa berjalan lebih cepat dibandingkan kode yang sama dan dituliskan dengan Java.
3. Interoperability. Semua library Android yang tersedia, dapat digunakan pada Kotlin.
4. Compilation Time. Kotlin mendukung kompilasi inkremental yang efisien. Oleh karena itu, proses build biasanya sama atau lebih cepat dibandingkan dengan Java.
5. **PRAKTIK**

Kita akan membuat program kotlin dengan dibandingkan dengan java. Gunakan laman web (<https://try.kotlinlang.org>) untuk mencoba menjalankan program kotlin.

Dikutip dari <https://kotlinlang.org/docs/reference/basic-syntax.html> Defining packages. Package specification should be at the top of the source file:

|  |
| --- |
|  |
| Disaat di run maka tampiannya no project karena belum ada project yang mau dibuat |

It is not required to match directories and packages: source files can be placed arbitrarily in the file system.

Defining functions

|  |
| --- |
| Program |
|  |

Function with an expression body and inferred return type:

|  |
| --- |
| Program |
|  |
| Hasilnya |
|  |

Function with an expression body and inferred return type:

|  |
| --- |
| Program |
| Hasilnya |
|  |

|  |
| --- |
| Program |
|  |
| Hasilnya |
|  |

Unit return type can be omitted:

|  |
| --- |
| Program |
|  |
| Hasilnya |
|  |

Defining variables Read-only local variables are defined using the keyword val. They can be assigned a value only once.

|  |
| --- |
| Program |
| Hasilnya |
|  |

Variables that can be reassigned use the var keyword:

|  |
| --- |
| var x = 5 // `Int` type is inferred x += 1 |
|  |
| Hasilnya |
|  |

Top-level variables:

|  |
| --- |
| Programnya |
|  |
| Hasil setelah dirun |
|  |

|  |
| --- |
| // This is an end-of-line comment /\*  This is a block comment on multiple lines. \*/  Unlike Java, block comments in Kotlin can be nested. |

See Documenting Kotlin Code for information on the documentation comment syntax

Using string templates

|  |
| --- |
|  |
| Programnya |
|  |
| Hasil setelah dirun |

Using conditional expressions

|  |
| --- |
| fun maxOf(a: Int, b: Int): Int {  if (a > b) {  return a  }  else {  return b  }  } |
| Programnya |
|  |
| Hasil setelah dirun |
|  |

Using if as an expression:

|  |
| --- |
| fun maxOf(a: Int, b: Int) = if (a > b) a else b |
| Programnya |
|  |
| Hasil setelah dirun |
|  |

**Using nullable values and checking for null**

A reference must be explicitly marked as nullable when null value is possible. Return null if str does not hold an integer:

|  |
| --- |
| fun parseInt(str: String): Int? {  // ...  } |

**Use a function returning nullable value:**

|  |
| --- |
| Programnya |
|  |
| Hasil setelah dirun |
|  |

Or

|  |
| --- |
| Programnya |
|  |
| Hasil setelah dirun |
|  |

**Using type checks and automatic casts**

The is operator checks if an expression is an instance of a type. If an immutable local variable or property is checked for a specific type, there's no need to cast it explicitly:

|  |
| --- |
| Program |
|  |
| Hasilnya |
|  |

Or

|  |
| --- |
| Program |
|  |
| Hasilnya |
|  |

Or even

|  |
| --- |
| Program |
|  |
| Hasilnya |
|  |

Using a for loop

|  |
| --- |
| Programnya |
|  |
| Hasil setelah dirun |
|  |

Or

|  |
| --- |
| Programnya |
|  |
| Hasilnya |
|  |

Using a while loop

|  |
| --- |
| Programnya |
|  |
| Hasilnya |
|  |

**Using when expression**

|  |
| --- |
| Programnya |
|  |
| Hasilnya |
|  |

**Using ranges**

Check if a number is within a range using in operator:

|  |
| --- |
| Programnya |
|  |
| Hasilnya |
|  |

Check if a number is out of range:

|  |
| --- |
| Programnya |
|  |
| Hasilnya |
|  |

Iterating over a range:

|  |
| --- |
| Programnya |
|  |
| Hasilnya |
|  |

or over a progression:

|  |
| --- |
| Program |
|  |
| Hasilnya |
|  |

**Using collections**

Iterating over a collection:

|  |
| --- |
| Program |
|  |
| Hasilnya |
|  |

Checking if a collection contains an object using in operator:

|  |
| --- |
| Program |
|  |
| Hasilnya |
|  |

Using lambda expressions to filter and map collections:

|  |
| --- |
| Program |
|  |
| Hasilnya |
|  |

Creating basic classes and their instances:

|  |
| --- |
| Program |
|  |
| Hasilnya |
|  |

1. **LATIHAN**
2. Pelajari dan cobalah bahasa pemrograman kotlin lebih lanjut dari laman web kotlinlang.org
3. **TUGAS**
4. Silakan mempelajari bahasa pemrograman kotlin lebih lanjut dari laman web kotlinlang.org
5. **KESIMPULAN**

Mahasiswa mampu memahami dan mengembangkan aplikasi sederhana dengan bahasa pemrograman Kotlin. Kotlin/Native memungkinkan developer untuk menggunakannya sebagai bahasa pemrograman dalam pengembangan aplikasi di platform lain seperti embedded system, desktop, macOS, dan iOS.