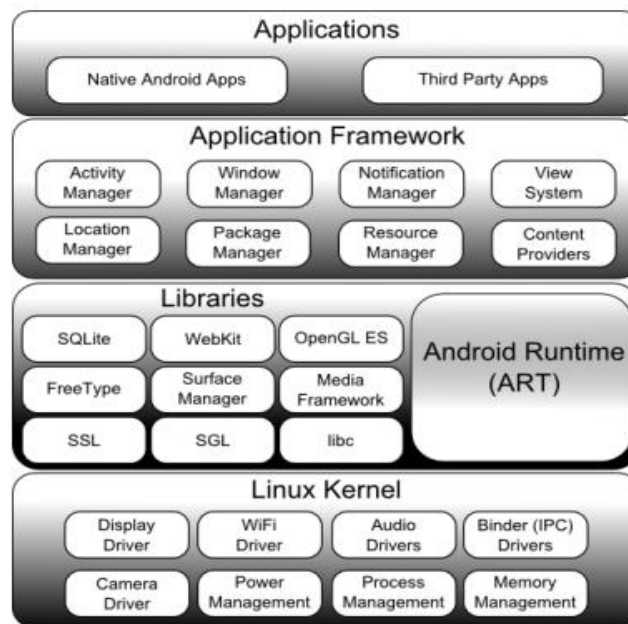


# Pengenalan Android

## 1. Apa itu android?

Android adalah sistem operasi berbasis java yang berjalan pada kernel linux 2.6. sistem ini sangat ringan dan memiliki banyak fitur. Contoh gambar arsitektur dari sistem operasi android ditampilkan pada gambar 1.1



**Gambar 1.1 Arsitektur Android**

Aplikasi android dikembangkan dengan menggunakan bahasa pemrograman java. Sehingga untuk mengembangkan aplikasi android diperlukan Java Development Kit (JDK). Bahasa pemrograman java sangat erat kaitannya dengan pemrograman berorientasi objek (OOP) sehingga untuk dapat mengembangkan aplikasi android dibutuhkan keahlian dalam bidang pemrograman berorientasi objek.

## 2. Install Android

<https://developer.android.com/studio/install>  
<https://developer.android.com/studio>

Kita akan belajar bagaimana menginstall Android Studio dengan lingkungan Google's Android development.

**Yang** diperlukan

- Sebuah komputer dengan sistem operasi Windows atau Linux, atau Mac dengan macOS. Pastikan sudah memenuhi *system* requirement.

- Diperlukan ruang disk beberapa gigabytes untuk install dan menjalankan Android Studio dan SDK yang berhubungan dan emulator images.
- Internet access, atau alternatif lain untuk mendapatkan file instalasi versi terakhir dari Android Studio.

### **Yang perlu dipelajari**

- Bagaimana menginstall dan memulai Android Studio integrated development environment (IDE).

### **System requirements**

#### **Windows**

- Microsoft® Windows® 7/8/10 (32- or 64-bit)  
*The Android Emulator only supports 64-bit Windows.*
- 4 GB RAM minimum, 8 GB RAM recommended
- 2 GB of available disk space minimum, 4 GB Recommended (500 MB for IDE + 1.5 GB for Android SDK and emulator system image)
- 1280 x 800 minimum screen resolution

#### **Mac**

- Mac® OS X® 10.10 (Yosemite) or higher, up to 10.14 (macOS Mojave)
- 4 GB RAM minimum, 8 GB RAM recommended
- 2 GB of available disk space minimum,  
4 GB Recommended (500 MB for IDE + 1.5 GB for Android SDK and emulator system image)
- 1280 x 800 minimum screen resolution

#### **Linux**

- GNOME or KDE desktop  
*Tested on Ubuntu® 14.04 LTS, Trusty Tahr (64-bit distribution capable of running 32-bit applications)*
- 64-bit distribution capable of running 32-bit applications
- GNU C Library (glibc) 2.19 or later
- 4 GB RAM minimum, 8 GB RAM recommended
- 2 GB of available disk space minimum,  
4 GB Recommended (500 MB for IDE + 1.5 GB for Android SDK and emulator system image)
- 1280 x 800 minimum screen resolution

#### **Chrome OS**

- 8 GB RAM or more recommended
- 4 GB of available disk space minimum
- 1280 x 800 minimum screen resolution
- Intel i5 or higher (U series or higher) recommended
- Recommended devices:
- Acer: Chromebook 13/Spin 13, Chromebox CX13
- Lenovo: Yoga C630 Chromebook
- HP: Chromebook x360 14, Chromebox G2
- Dell: Inspiron Chromebook 14
- ASUS: Chromebox 3
- ViewSonic: NMP660 Chromebox
- CTL: Chromebox CBx1

## Install Android Studio

(<https://codelabs.developers.google.com/codelabs/kotlin-android-training-install-studio/index.html?index=..%2F..android-kotlin-fundamentals#1>)

Android Studio menyediakan IDE lengkap, termasuk editor kode canggih dan templat aplikasi. Ini juga berisi alat untuk pengembangan, debugging, pengujian, dan kinerja yang membuatnya lebih cepat dan lebih mudah untuk mengembangkan aplikasi. Anda dapat menggunakan Android Studio untuk menguji aplikasi Anda dengan sejumlah besar emulator yang telah dikonfigurasi sebelumnya, atau pada perangkat seluler Anda sendiri. Anda juga dapat membuat app dan mempublikasikan aplikasi di Google Play Store.

*Catatan: Android Studio terus ditingkatkan. Untuk informasi terbaru tentang persyaratan sistem dan instruksi pemasangan, lihat halaman unduhan Android Studio. Android Studio tersedia untuk komputer yang menjalankan Windows atau Linux, dan untuk Mac yang menjalankan macOS. OpenJDK (Java Development Kit) terbaru dibundel dengan Android Studio.*

Instalasi serupa untuk semua platform. Setiap perbedaan dicatat di bawah ini.

1. Navigasikan ke halaman unduh Android Studio (<https://developer.android.com/studio/>) dan ikuti instruksi untuk mengunduh dan menginstal Android Studio (<https://developer.android.com/studio/install.html>)
2. Terima konfigurasi default untuk semua langkah, dan pastikan bahwa semua komponen dipilih untuk instalasi.
3. Setelah penginstalan selesai, wizard pengaturan akan mengunduh dan menginstal komponen tambahan, termasuk Android SDK. Bersabarlah, karena proses ini mungkin memakan waktu, tergantung pada kecepatan internet Anda.
4. Ketika instalasi selesai, Android Studio dimulai, dan Anda siap untuk membuat proyek pertama Anda.

Pemecahan masalah: Jika Anda mengalami masalah dengan instalasi Anda, lihat catatan rilis [Android Studio release notes](#) or [Troubleshoot Android Studio](#)..

## Install Android Studio di Windows

Pastikan sudah [download versi terakhir dari Android Studio](#).

Untuk menginstal Android Studio di Windows, lanjutkan sebagai berikut:

1. Jika Anda mengunduh file .exe (disarankan), klik dua kali untuk meluncurkannya. Jika Anda mengunduh file .zip, membongkar ZIP, salin folder android-studio ke folder Program Files, lalu buka android-studio> folder bin dan jalankan studio64.exe (untuk mesin 64-bit) atau studio.exe (untuk mesin 32-bit).
2. Ikuti wizard pengaturan di Android Studio dan instal paket SDK apa pun yang direkomendasikan.

## **Komponen Pemrograman Android**

Dalam mengembangkan aplikasi android dibutuhkan berbagai macam komponen yang saling berkaitan. Dengan menggunakan komponen-komponen ini pengembang akan dimudahkan karena setiap komponen memiliki perannya masing-masing. Komponen yang dibutuhkan dan perlu dipelajari untuk pengembangan aplikasi android diantaranya :

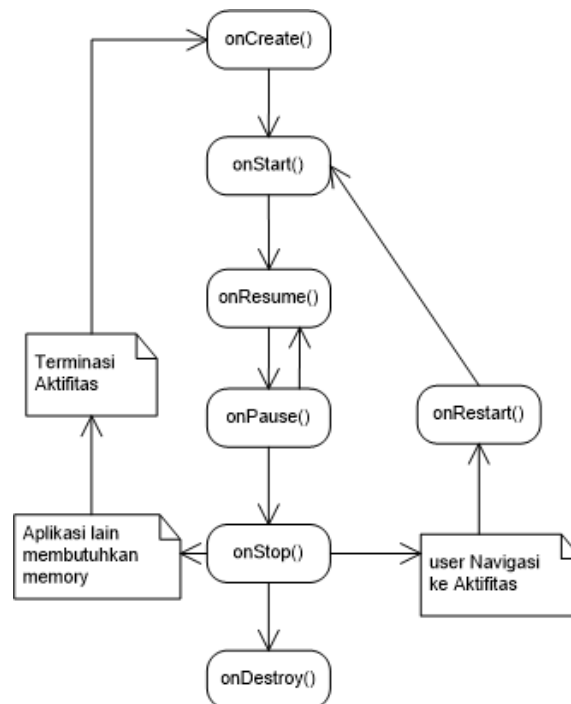
### **a) Aktivitas**

Aktivitas dimaksudkan sebagai komponen, yang dapat digunakan kembali dan dapat dipertukarkan, dan bisa dibagi di antara aplikasi yang berbeda. Sebuah aplikasi email yang ada, misalnya, mungkin berisi Aktivitas khusus untuk membuat dan mengirim pesan email. Seorang pengembang mungkin menulis sebuah aplikasi yang juga memiliki persyaratan untuk mengirim pesan email. Daripada mengembangkan Aktivitas komposisi email khusus untuk aplikasi baru, pengembang hanya dapat menggunakan Aktivitas dari aplikasi email yang ada.

Aktivitas diciptakan sebagai subclass dari kelas Activity Android dan harus dieksekusi sehingga menjadi terpisah sepenuhnya dari Aktivitas lain dalam aplikasi. Dengan kata lain, Aktivitas bersama tidak bisa dipanggil langsung dalam program (karena aplikasi lain dapat menggunakan Aktivitas) dan satu Aktivitas tidak bisa langsung memanggil metode atau mengakses data Aktivitas lain. Sebagai gantinya, untuk mencapai tujuan ini, dengan menggunakan Intents dan Content Providers. Secara default, suatu Aktivitas tidak dapat memberikan hasil dengan aktivitas yang dipanggil. Jika fungsi ini diperlukan, Aktivitas harus secara khusus dimulai sebagai sub-aktivitas.

### **Siklus hidup Aktifitas**

Siklus hidup Aktivitas adalah proses yang digunakan semua Aktivitas untuk dipanggil, jalan logika, dan penyelesaian dengan cara yang terstruktur dan dapat diandalkan. Ini membantu sistem menjaga stabilitas dan mengelola sumber daya sistem. Beberapa metode, seperti finish(), dapat dipanggil dalam Aktivitas untuk memaksa Aktivitas ditutup; Namun, disarankan agar Anda membiarkan sistem Android mengelola kapan harus menyelesaikan atau menghancurkan Aktivitas.



**Gambar 1.2 Android lifecycle**

## b) Intent

Intent digunakan sistem untuk memulai sesuatu aktivitas. Ketika menggunakan intent terdapat dua tipe intent yaitu explicit intent dan implicit intent.

Untuk membuat intent dibutuhkan nama intent yang akan digunakan kemudian instansiasi dari intent yang akan dibuat, jika intent bersifat explicit maka diperlukan nama aktivitas yang akan dituju.

## c) Intent Filter

Intent filter dibuat dengan menambahkan kode `<intent-filter>` pada file manifest. Jika dibuat intent filter, maka harus disertakan turunan elemennya yaitu `<action>`, `<data>` dan `<category>`.

## d) Broadcast Receiver

Bila Intent dibuat dan dikirim, aplikasi memerlukan cara untuk mengambilnya kembali. Hal ini dilakukan dengan membuat broadcast receiver. Ini adalah proses dua bagian: Pertama kali membuat Broadcast Receiver dalam file kelas dan kemudian mendaftarkan file kelas di dalam manifest XML yang dibuat, dengan `<receiver>` yang berisi elemen turunan `<intent-filter>`.

## 1. Struktur pemrograman Android

### a) Manifest

File manifest dibuat android studio ketika pembuatan proyek pertama kali. Contoh cari file manifest yaitu.

#### Listing 1.1 Default Manifest dari Android Studio

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="dev.udinsi.ofishmap">

    <application
        android:allowBackup="true"
        android:icon="@drawable/icon"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity
            android:name=".SplashScreen"
            android:theme="@style/AppTheme.gayaku">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

File manifest biasanya dibuat dengan menggunakan XML, sehingga ketika melihat file manifest akan dipastikan ada file dengan awalan elemen `<?xml version="1.0" encoding="utf-8"?>`. Setelah elemen tersebut akan ada elemen lain yaitu `<manifest>`. `<manifest>` memiliki beberapa atribut seperti nama paket(package) dan name space android. Contoh dari penggunaan atribut namespace android dan package yaitu.

```
<manifest
    xmlns:android="http://schemas.android.com/apk/res/android"
    package="dev.udinsi.ofishmap">
```

Dalam element manifest ada beberapa child element (elemen turunan) yang dapat digunakan. Element turunan yang dapat digunakan pada `<manifest>` ditampilkan pada

#### Listing 1.2 Child Element Manifest

```
<manifest>
    <uses-permission />
    <permission />
    <permission-tree />
    <permission-group />
    <instrumentation />
```

```

<uses-sdk />
<uses-configuration />
<uses-feature />
<supports-screens />
<compatible-screens />
<supports-gl-texture />
<application>
    <activity>
        <intent-filter>
            <action />
            <category />
            <data />
        </intent-filter>
        <meta-data />
    </activity>
<activity-alias>
    <intent-filter>
</intent-filter>
    <meta-data />
</activity-alias>
<service>
    <intent-filter>
</intent-filter>
    <meta-data/>
</service>
<receiver>
    <intent-filter>
</intent-filter>
    <meta-data />
</receiver>
<provider>
    <grant-uri-permission />
    <meta-data />
    <path-permission />
</provider>
<uses-library />
</application>
</manifest>

```

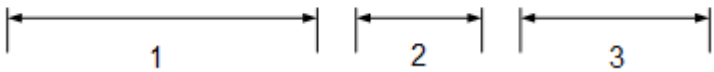
## b) Java

Folder java digunakan untuk meletakkan semua file java yang dibuat oleh pengembang. Dalam folder java dapat dibuat sub folder sesuai dengan yang dibutuhkan. Hal ini bertujuan agar pengembang dapat membedakan folder untuk meletakkan kelas-kelas dari masing masing kebutuhan. Sebagai contoh, apabila akan dibuat kelas untuk mengatur database, maka dalam folder java dapat dibuat sub folder 'database'. Dengan demikian maka pengembang dapat dengan mudah menemukan file yang harus di edit atau yang perlu diperbaharui. Untuk menggunakan kelas yang dibuat pada sub folder harus dilakukan import kelas yang dibutuhkan, sebagai contoh, jika kita akan menggunakan file MyDatabase dari package database, maka perlu dilakukan

```
import dev.udinsi.ofishmap.database.MyDatabase;
```

Penjelasan dari contoh kode yaitu :

```
import dev.udinsi.ofishmap.database.MyDatabase;
```



**Gambar 1.3 Contoh Import Class dalam Java**

Keterangan :

1. Nama package (paket) saat membuat aplikasi
2. Sub folder yang dibuat manual
3. Nama kelas yang akan digunakan.

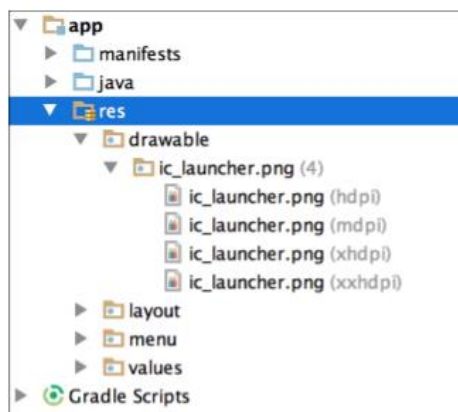
### c) Res (Resources)

Folder res berisi controller layout, media, menu dan constant yang tersimpan pada values yang akan digunakan pada aplikasi.

#### a) Drawable

Folder drawable berisi semua media visual dan resource yang dibutuhkan aplikasi.

Saat bekerja dengan Android Studio, perhatikan bahwa tidak semua folder dapat ditampilkan pada resource. Pada sistem file mungkin memiliki folder drawable, drawable-hdpi, drawable-mdpi, dan drawable-xhdpi yang terpisah, dengan masing-masing berisi resource yang diberi nama sama namun harus digunakan secara khusus untuk kepadatan layar yang berbeda. Di Android Studio, sumber ini akan ditampilkan di folder yang dapat digambar sebagai folder yang bisa diperluas, dengan sumber daya yang akan digunakan dengan tanda kurung. Gambar 3.2 menunjukkan bagaimana Android Studio menampilkan resource dengan nama yang sama di folder yang bergantung kepadatan piksel.





## Gambar 1.4 Struktur dalam Folder Drawable

### b) layout

Folder layout menampung file XML yang digunakan untuk layout aplikasi. Berkas layout default dinamai sesuai aktivitas yang dibuat, jika membuat proyek baru di Android Studio dan memilih setelan default adalah `activity_main.xml`.

File ini digunakan untuk menyiapkan tata letak untuk aktivitas dan digunakan untuk penyejajaran dasar tata letak, komponen, widget, dan aset sejenis yang digunakan untuk UI (User interface) aplikasi.

Tata letak juga dapat didukung berdasarkan kepadatan layar berdasarkan titik-per-inci (dpi) dengan menggunakan folder yang dinamai berdasarkan kepadatan, seperti yang tercantum:

**Table 1.1 Jenis Kepadatan Pixel**

| Jenis kepadatan | Jumlah kepadatan (dpi)   |
|-----------------|--|
| Ldpi            | 120  |
| Mdpi            | 160  |
| Hdpi            | 240  |
| Xhdpi           | 320  |
| Xxhdpi          | 480  |
| Xxxhdpi         | 640  |
| Nodpi           | Sumber daya di sini akan digunakan untuk semua kerapatan.        |
| Tvdpi           | Digunakan pada layar berukuran antara mdpi dan hdpi, sekitar 213 |

### c) Menu

Untuk membuat atau menambahkan menu ke aplikasi, XML yang mendefinisikan menu berada dalam folder ini. pilihan untuk membuat nama apa pun yang diinginkan untuk menu yang akan dibuat, namun membuat proyek baru di Android Studio dan menggunakan opsi default, akan mendapati menu telah diberi nama `menu_main.xml`

Konvensi penamaan ini sebenarnya cukup membantu karena mengidentifikasi file XML apa untuk "menu" dan aktivitas apa yang ditugaskan ke "main."

#### d) values

Folder values digunakan untuk melacak nilai yang akan digunakan di aplikasi. Untuk membuat aplikasi dengan siklus perawatan yang lebih mudah, sangat disarankan untuk tidak lagi memasukkan hard code ke kode yang dibuat. Sebagai gantinya, tempatkan nilai pada file XML di dalam folder values.

Contoh penggunaan coding

#### Listing 1.3 Perbedaan Hard Code dan Penggunaan Values (soft code)

```
// Hard-coding a resource
<TextView android:text="Hello Android!"
android:layout_width="wrap_content"
android:layout_height="wrap_content" />

// Using a value from /res/values/strings.xml
<TextView android:text="@string/hello_android"
android:layout_width="wrap_content"
android:layout_height="wrap_content" />
```

Pada contoh sebelumnya, nilai yang ditunjukkan pada TextView akan berubah berdasarkan string yang dimasukkan ke dalam string.xml. Berikut ini adalah contoh file strings.xml:

#### Listing 1.4 contoh string.xml pada folder values

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <string name="hello_android">Hello Android!</string>
</resources>
```

Ketika proyek android pertama kali dibuat dengan android studio, maka akan tercipta file XML secara otomatis di antaranya :

- dims.xml
- string.xml
- style.xml
- colors.xml
- arrays.xml

#### e) gradle

Untuk membantu mengatur pembangunan file, android studio menambahkan sebuah sesi dengan skrip gradle ke dalam proyek yang sedang dibuat. Skrip Gradle akan menampilkan konfigurasi pengembangan aplikasi seperti file properti dan pengaturan file yang akan digunakan pada pembangunan aplikasi. Listing 1.5 akan menampilkan contoh skrip gradle

## Listing 1.5 build.gradle pada Aplikasi Android

```
apply plugin: 'com.android.application'
android {
    compileSdkVersion 25
    buildToolsVersion "25.0.0"
    defaultConfig {
        multiDexEnabled true
        applicationId "dev.udinsi.ofishmap"
        minSdkVersion 21
        targetSdkVersion 25
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner
            "android.support.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles
                getDefaultProguardFile('proguard-android.txt'),
                'proguard-rules.pro'
        }
    }
    dependencies {

        compile fileTree(dir: 'libs', include: ['*.jar'])
        androidTestCompile('com.android.support.test.espresso:
espresso-core:2.2.2', {
            exclude group: 'com.android.support',
                module: 'support-annotations'
        })
        compile 'com.android.support:appcompat-v7:25.1.0'
        compile 'com.android.support:design:25.1.0'
        compile 'com.android.support:gridlayout-v7:25.1.0'
        compile 'com.android.support:recyclerview-v7:25.1.0'
        compile 'net.gotev:uploadservice:2.1'
        compile 'com.android.support.constraint:constraint-layout:1.0.2'
        compile 'com.mcxiaoke.volley:library-aar:1.0.0'
        compile 'de.hdodenhof:circleimageview:2.1.0'
        testCompile 'junit:junit:4.12'
    }
}
```

## 3. Kotlinlang

### a. 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 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.

#### b. Memulai kotlin

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:

```
package my.demo

import java.util.*

// ...
```

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

See [Packages](#).

#### Defining functions

Function having two Int parameters with Int return type:

```
fun sum(a: Int, b: Int): Int {
    return a + b
}
```

Target platform: JVMRunning on kotlin v. 1.3.41

Function with an expression body and inferred return type:

```
fun sum(a: Int, b: Int) = a + b
```

Target platform: JVMRunning on kotlin v. 1.3.41

Function returning no meaningful value:

```
fun printSum(a: Int, b: Int): Unit {
    println("sum of $a and $b is ${a + b}")
}
```

Target platform: JVMRunning on kotlin v. 1.3.41

Unit return type can be omitted:

```
fun printSum(a: Int, b: Int) {
    println("sum of $a and $b is ${a + b}")
}
```

Target platform: JVMRunning on kotlin v. 1.3.41

See [Functions](#).

#### Defining variables

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

```
val a: Int = 1 // immediate assignment
val b = 2 // `Int` type is inferred
val c: Int // Type required when no initializer is provided
c = 3 // deferred assignment
```

Target platform: JVMRunning on kotlin v. 1.3.41

Variables that can be reassigned use the **var** keyword:

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

Target platform: JVMRunning on kotlin v. 1.3.41

Top-level variables:

```
val PI = 3.14
var x = 0

fun incrementX() {
    x += 1
}
```

Target platform: JVMRunning on kotlin v. 1.3.41

See also Properties And Fields.

Comments

Just like Java and JavaScript, Kotlin supports end-of-line and block comments.

```
// 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**

```
var a = 1
// simple name in template:
val s1 = "a is $a"

a = 2
// arbitrary expression in template:
val s2 = "${s1.replace("is", "was")}, but now is $a"
```

Target platform: JVMRunning on kotlin v. 1.3.41

See [String templates](#).

**Using conditional expressions**

```
fun maxOf(a: Int, b: Int): Int {
    if (a > b) {
        return a
    } else {
        return b
    }
}
```

Target platform: JVMRunning on kotlin v. 1.3.41

Using **if** as an expression:

```
fun maxOf(a: Int, b: Int) = if (a > b) a else b
```

Target platform: JVMRunning on kotlin v. 1.3.41

See [if-expressions](#).

### 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:

```
fun printProduct(arg1: String, arg2: String) {  
    val x = parseInt(arg1)  
    val y = parseInt(arg2)  
  
    // Using `x * y` yields error because they may hold nulls.  
    if (x != null && y != null) {  
        // x and y are automatically cast to non-nullable after null check  
        println(x * y)  
    }  
    else {  
        println("either '$arg1' or '$arg2' is not a number")  
    }  
}
```

Target platform: JVMRunning on kotlin v. 1.3.41

or

```
// ...  
if (x == null) {  
    println("Wrong number format in arg1: '$arg1'")  
    return  
}  
if (y == null) {  
    println("Wrong number format in arg2: '$arg2'")  
    return  
}  
  
// x and y are automatically cast to non-nullable after null check  
println(x * y)
```

Target platform: JVMRunning on kotlin v. 1.3.41

See [Null-safety](#).

### 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:

```
fun getStringLength(obj: Any): Int? {  
    if (obj is String) {  
        // `obj` is automatically cast to `String` in this branch  
        return obj.length  
    }  
  
    // `obj` is still of type `Any` outside of the type-checked branch  
    return null  
}
```

Target platform: JVMRunning on kotlin v. 1.3.41

or

```
fun getStringLength(obj: Any): Int? {  
    if (obj !is String) return null  
  
    // `obj` is automatically cast to `String` in this branch  
    return obj.length  
}
```

Target platform: JVMRunning on kotlin v. 1.3.41

or even

```
fun getStringLength(obj: Any): Int? {  
    // `obj` is automatically cast to `String` on the right-hand side of `&&`  
    if (obj is String && obj.length > 0) {  
        return obj.length  
    }  
  
    return null  
}
```

Target platform: JVMRunning on kotlin v. 1.3.41

See [Classes](#) and [Type casts](#).

### Using a for loop

```
val items = listOf("apple", "banana", "kiwifruit")  
for (item in items) {  
    println(item)  
}
```

Target platform: JVMRunning on kotlin v. 1.3.41

or

```
val items = listOf("apple", "banana", "kiwifruit")  
for (index in items.indices) {  
    println("item at $index is ${items[index]}")  
}
```

Target platform: JVMRunning on kotlin v. 1.3.41

See [for loop](#).

### Using a while loop

```
val items = listOf("apple", "banana", "kiwifruit")  
var index = 0  
while (index < items.size) {  
    println("item at $index is ${items[index]}")  
    index++  
}
```

Target platform: JVMRunning on kotlin v. 1.3.41

See [while loop](#).

### Using when expression

```
fun describe(obj: Any): String =  
    when (obj) {  
        1      -> "One"  
        "Hello" -> "Greeting"  
        is Long  -> "Long"  
        !is String -> "Not a string"  
        else    -> "Unknown"  
    }
```

Target platform: JVMRunning on kotlin v. 1.3.41

See [when expression](#).

### Using ranges

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

```
val x = 10  
val y = 9  
if (x in 1..y+1) {  
    println("fits in range")  
}
```

Target platform: JVMRunning on kotlin v. 1.3.41

Check if a number is out of range:

```
val list = listOf("a", "b", "c")  
  
if (-1 !in 0..list.lastIndex) {  
    println("-1 is out of range")  
}
```

```
if (list.size !in list.indices) {  
    println("list size is out of valid list indices range, too")  
}
```

Target platform: JVMRunning on kotlin v. 1.3.41

Iterating over a range:

```
for (x in 1..5) {  
    print(x)  
}
```

Target platform: JVMRunning on kotlin v. 1.3.41

or over a progression:

```
for (x in 1..10 step 2) {  
    print(x)  
}  
println()  
for (x in 9 downTo 0 step 3) {  
    print(x)  
}
```

Target platform: JVMRunning on kotlin v. 1.3.41

See [Ranges](#).

## Using collections

Iterating over a collection:

```
for (item in items) {  
    println(item)  
}
```

Target platform: JVMRunning on kotlin v. 1.3.41

Checking if a collection contains an object using *in* operator:

```
when {  
    "orange" in items -> println("juicy")  
    "apple" in items -> println("apple is fine too")  
}
```

Target platform: JVMRunning on kotlin v. 1.3.41

Using lambda expressions to filter and map collections:

```
val fruits = listOf("banana", "avocado", "apple", "kiwifruit")  
fruits  
    .filter { it.startsWith("a") }  
    .sortedBy { it }  
    .map { it.toUpperCase() }  
    .forEach { println(it) }
```

Target platform: JVMRunning on kotlin v. 1.3.41

See [Higher-order functions and Lambdas](#).

## Creating basic classes and their instances:

```
val rectangle = Rectangle(5.0, 2.0) //no 'new' keyword required  
val triangle = Triangle(3.0, 4.0, 5.0)
```

Target platform: JVMRunning on kotlin v. 1.3.41

See [classes](#) and [objects and instances](#).