



BINAR
ACADEMY

Intent, Bundle, Serializable, dan Parcelable

Silver - Chapter 3 - Topic 4

**Selamat datang di Chapter 3 Topic 4 *online course*
Android Developer dari Binar Academy!**



Hi Teman-Teman 🙋

Pada topik sebelumnya, kita sudah belajar konsep pemrograman dengan fitur **Fragment** dan **Fragment Lifecycle**.

Nah, di **Topik 4** ini kita mendalami fitur lain yang mirip juga, yaitu **Cara Berpindah Activity dengan Intent**.

Yuk, lanjut!



Detailnya, kita bakal bahas hal-hal berikut ini:

- Definisi Intent, Implicit Intent dan Explicit Intent
- Cara mengirim data melalui Intent
- Definisi dan praktik Bundle, Serializable dan Parcelable.

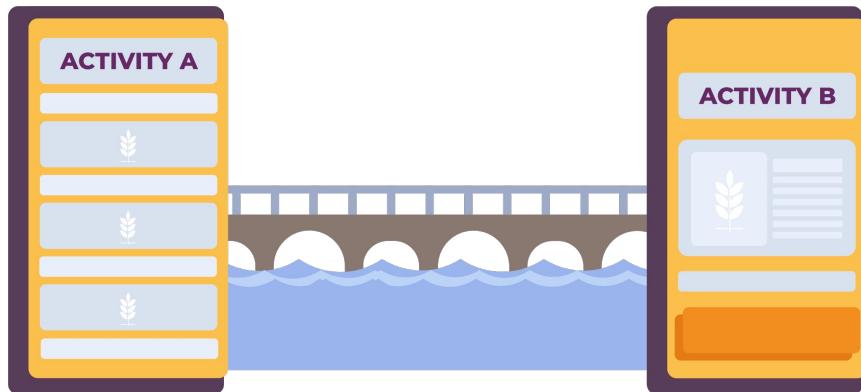




**Intent oh intent.. kedengerennya
masih asing ya di telinga kita. Tapi coba
kamu bayangin sebuah jembatan.**

**Inten itu mirip jembatan yang ada di
imajinasi kamu sekarang.**

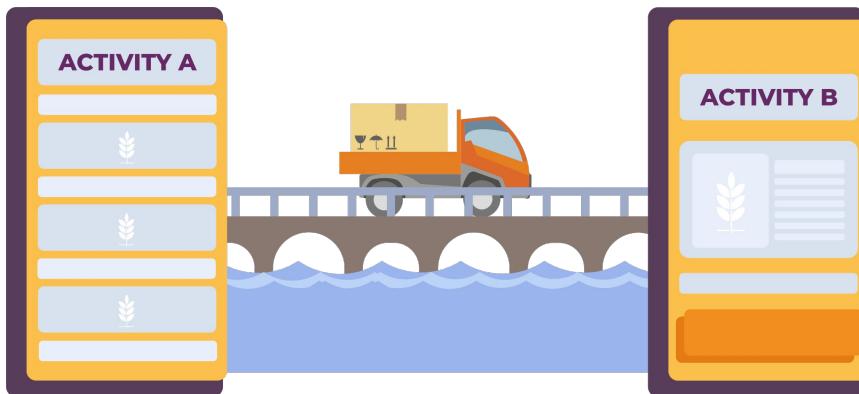
**Tapi kayak gimana sih Intent yang
mirip jembatan ini? Apa sih Intent itu?**



Biar Gampang, kita pakai analogi, ya!

Intent itu diibaratin kayak sebuah **jembatan** yang menghubungkan 2 wilayah, yakni wilayah asal dan wilayah tujuan.

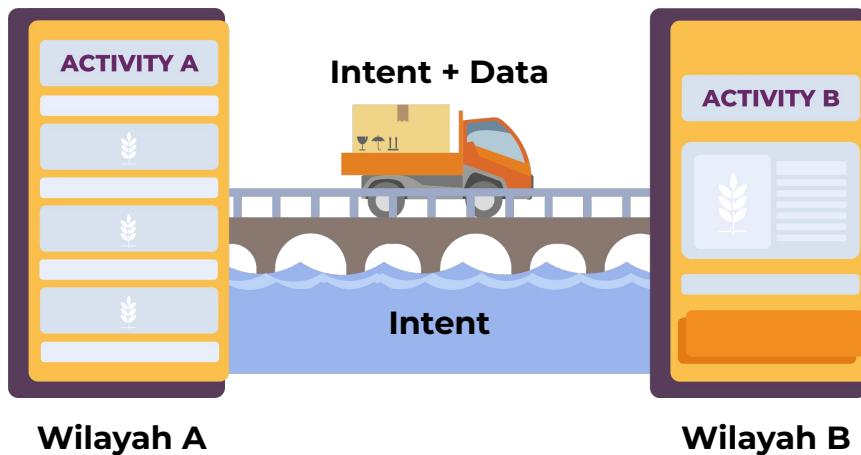
Karena ada si jembatan ini, kita jadi bisa berpindah dong dari satu tempat ke tempat lain.



Tapi apa kita berpindah diri dengan tangan kosong?
Pasti ada barang yang perlu kita bawa.

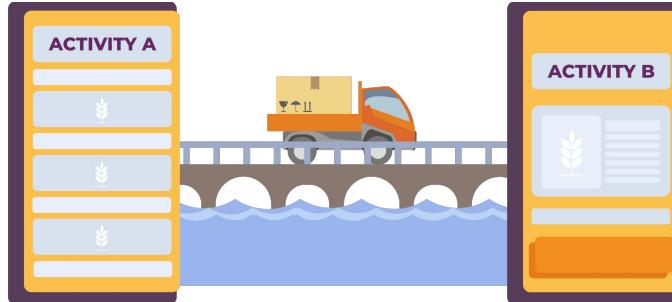
Ketika berpindah antar wilayah ini, kita perlu
memanfaatkan **mobil truk** buat bawa barangnya.

Sehingga, dengan adanya mobil truk ini, kita bisa deh
memindahkan **satu** atau **banyak** barang antar
wilayah.



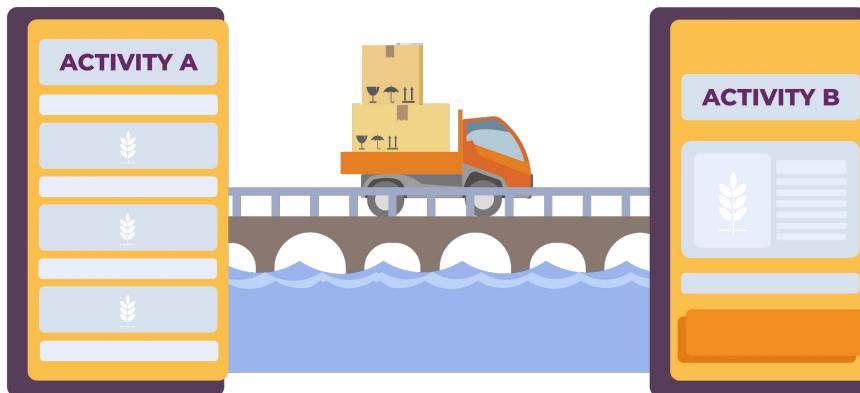
Nah, dari cerita itu, coba kondisinya kamu ganti dengan panduan seperti ini:

- **Wilayah :**
Activity asal maupun Activity tujuan.
- **Jembatan :**
Sebagai Intent yang menavigasi tampilan dari Activity saat ini (asal) ke Activity tujuan.
- **Mobil Truk :**
Sebagai Intent yang membawa data.



Jadi, apa itu Intent?

Intent merupakan **perintah yang dapat menjalankan Activity maupun Services dalam aplikasi kita**, mirip sebuah truk yang sedang lewat jembatan.



Bukan cuma pindah Activity aja, **Intent juga bisa sekalian membawa data ketika berpindah antar Activity.**

Bayangannya tuh kayak truk yang tadi nganter barang dari ujung ke ujung jembatan.

Intent nggak cuma bisa menjalankan Activity yang ada di aplikasi kita aja, tapi juga bisa menjalankan Activity pada aplikasi lain.

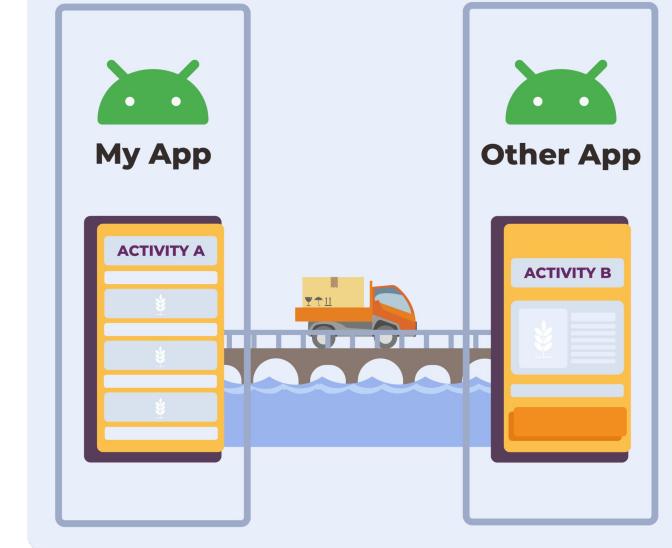


Intent terbagi menjadi dua kategori:

Explicit Intent



Implicit Intent

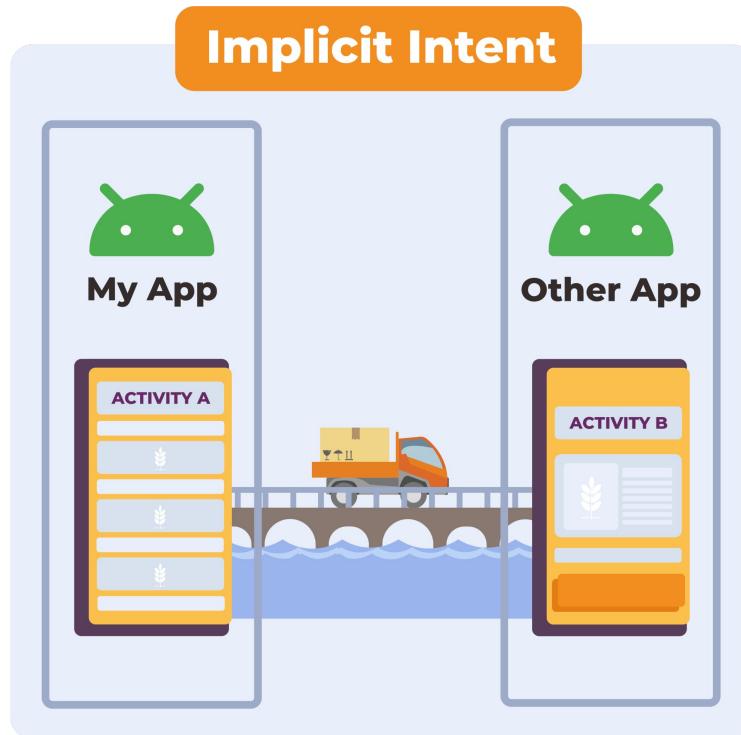




Siap-siap nih kita masuk ke pembahasan dua saudara.

Masuk ke jenis intent yang pertama,
Implicit Intent.

Apa sih maksudnya?



Apa sih Implicit Intent?

Implicit Intent adalah Intent untuk menjalankan Activity ataupun Service tertentu sesuai dengan aksi yang didefinisikan pada Intent.

Implicit Intent memungkinkan aplikasi kita untuk menggunakan fitur dari aplikasi lain yang terinstall di ponsel.



Gampangnya gini deh.. Implicit Intent itu ibarat kita mau jajan di warung abang-abang.

Karena cuma abang warungnya aja yang tau letak jajannya dimana, jadinya kita sebagai pelanggan, cukup ngelempar permintaan aja :

“Bang, saya mau jajan Permen.”



Nanti si-Abang akan cariin permen itu di dalam warungnya.

Nah, proses pencarian permen oleh si abang-abang ini kita sebut Implicit Intent.

Abangnya ini namanya “Android Systems”, sedangkan jajannya adalah “Aplikasi internal systems”.



Contohnya, kalo Implicit Intent didefinisiin untuk melakukan aksi pengambilan foto, maka bakal tampil tuh daftar aplikasi kita yang menyediakan fitur untuk mengambil foto, seperti :

- aplikasi kamera bawaan dari ponsel milikmu,
- aplikasi Camera 360 favoritmu,
- aplikasi 'Open Camera',
- Dan fitur sejenis lainnya.

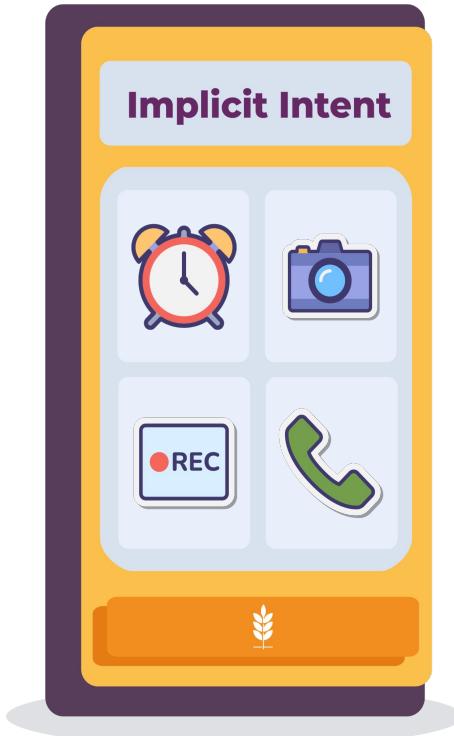


```
● ● ●  
val sendIntent = Intent().apply {  
    action = Intent.ACTION_SEND  
    putExtra(Intent.EXTRA_TEXT, textMessage)  
    type = "text/plain"  
}  
  
// Verify that the intent will resolve to an activity  
if (sendIntent.resolveActivity(packageManager) != null) {  
    startActivity(sendIntent)  
}
```

Cara Deklarasi Implicit Intent, Gimana?

Contoh code Implicit Intent yang melakukan **ACTION_SEND**, yaitu **mengirimkan pesan teks**, kayak gambar disamping.

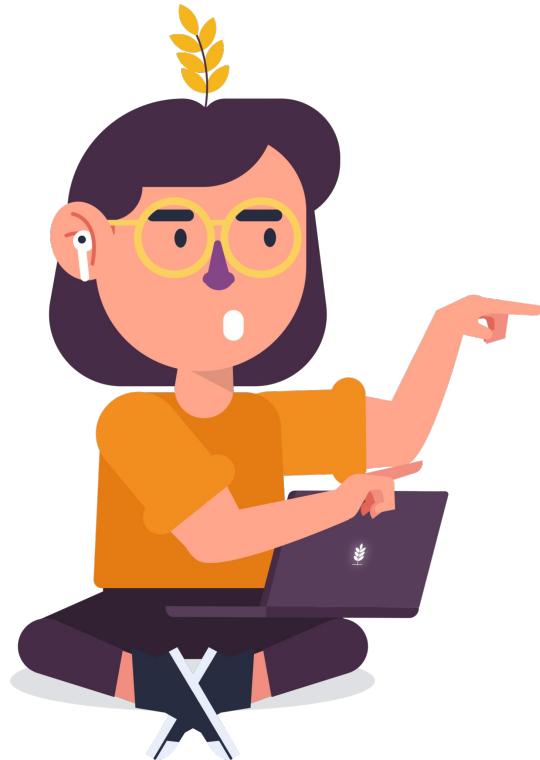
Kira-kira selain Intent **ACTION_SEND**, ada apa lagi?



Contoh Implicit Intent, ada yang lain lagi?

Buanyaak! Ini beberapa di antaranya:

1. **ACTION_SET_ALARM** - Mengatur Alarm
2. **ACTION_SET_TIMER** - Mengatur Timer
3. **ACTION_IMAGE_CAPTURE** - Mengambil gambar
4. **ACTION_VIDEO_CAPTURE** - Mengambil Video
5. **ACTION_DIAL** - Membuka Dialler Telepon
6. **ACTION_CALL** - Melakukan Panggilan



Selain enam contoh di atas, masih banyak lagi contoh lainnya yang bisa kamu liat dengan lengkap melalui dokumentasi resmi Android di link berikut ini:

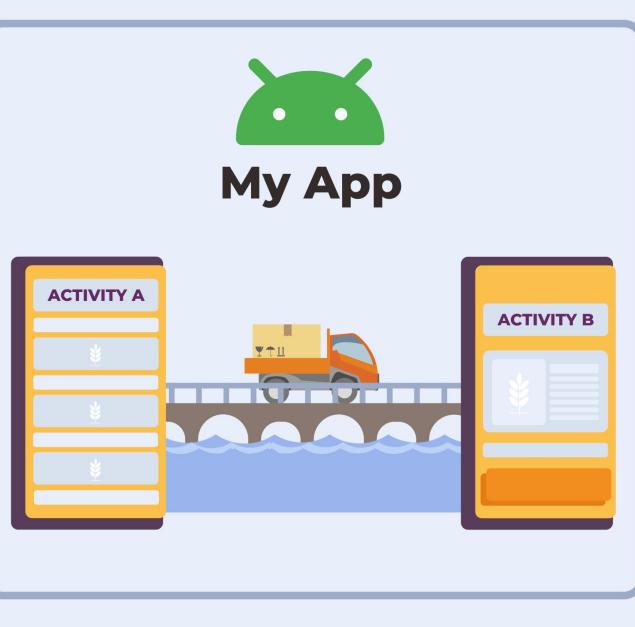
[Intent | Android Developers](#)



Si kakaknya udah kita bahas nih, lanjut ke adiknya yaitu **Explicit Intent**.

Apakah sih bedanya dengan **Implicit Intent**?

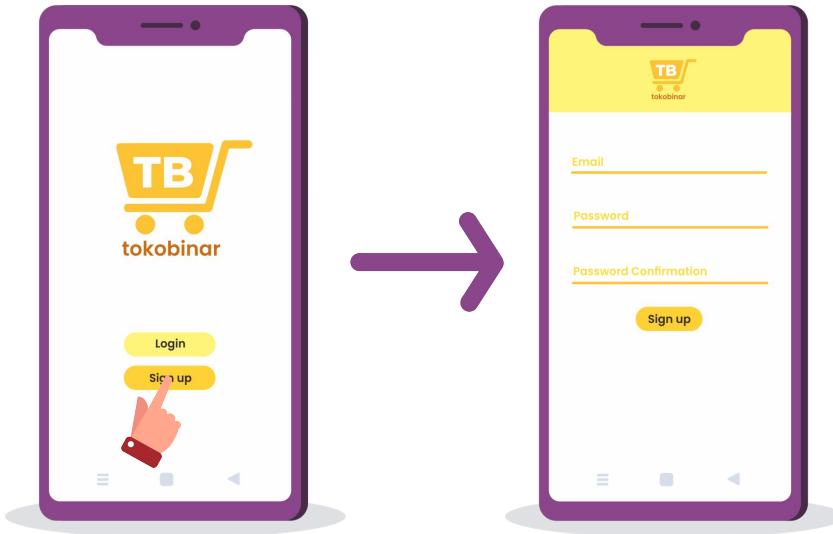
Explicit Intent



Apa itu Explicit Intent?

Explicit Intent adalah Intent untuk menjalankan ataupun berpindah Activity dalam aplikasi.

Selain itu, Explicit Intent juga bisa menjalankan Service tertentu yang ada di dalam aplikasi kita.



Jadi bedanya gimana?

Beda di fungsi dan code. Kayak gini... Implicit Intent tuh manggil **fungsi yang udah ada di internal Systems Android**. Misalnya kamera dan dial number. Makanya tinggal "lempar" permintaan aja.

Sedangkan Explicit Intent itu lebih ke **mengaktifkan komponen dalam satu aplikasi yang sama**, misalnya kayak berpindah screen (Activity) dan menjalankan service aplikasi.

Perbedaan ini juga termasuk dengan penulisan codenya.



Apa aja contoh Explicit Intent?

- Contoh Code yang digunakan untuk **berpindah antar Activity**:

```
// Untuk berpindah antar Activity  
val Intent = Intent(this, SecondActivity::class.java)  
startActivity(Intent)
```

Dengan catatan bahwa **SecondActivity** adalah class Activity tujuan.

- Contoh code yang digunakan untuk **menjalankan Service**:

```
// Untuk menjalankan Service  
val downloadIntent = Intent(this, DownloadService::class.java).apply {  
    data = Uri.parse(fileUrl)  
}  
startService(downloadIntent)
```

Dengan catatan bahwa **DownloadService** adalah class service, bukan class Activity.



Siip, udah paham ya Implicit intent dan Explicit intent perilakunya kayak gimana.

Nah, sekarang kita lanjut yuk ke pemahaman cara ngirim data lewat Intent.



Mengirim data menggunakan Explicit Intent

Pertama Explicit Intent dulu. Kalo di Explicit Intent memungkinkan untuk mengirimkan data melalui method `Intent.putExtra()`.

Contohnya bisa kamu lirik pada kode di samping.



```
val intent = Intent(this, SecondActivity::class.java)
intent.putExtra("name", "Pakde")
startActivity(intent)
```



Lewat kode tersebut, kita akan mengirimkan data sebuah String dengan key : name dan value : Pakde.

Lalu pada SecondActivity, kita dapat menerima data yang "Dilempar" dengan cara :

```
val name = Intent.getStringExtra("name")
```

Pada code di atas, kita menggunakan method **Intent.getStringExtra()** karena tipe data yang kita lemparkan dari Activity sebelumnya adalah String.



```
val intent = Intent(this, SecondActivity::class.java)
intent.putExtra("name", "Pakde")
startActivity(intent)
```



- `putExtra(String!, Bundle!)` defined in android.content.Intent
- `putExtra(String!, Parcelable!)` defined in android.content.Intent
- `putExtra(String!, Serializable!)` defined in android.content.Intent
- `putExtra(String!, Array<(out) Parcelable!>!)` defined in android.content.Intent
- `putExtra(String!, Array<(out) CharSequence!>!)` defined in android.content.Intent
- `putExtra(String!, Array<(out) String!>!)` defined in android.content.Intent
- `putExtra(String!, Boolean)` defined in android.content.Intent
- `putExtra(String!, BooleanArray!)` defined in android.content.Intent
- `putExtra(String!, Byte)` defined in android.content.Intent
- `putExtra(String!, ByteArray!)` defined in android.content.Intent
- `putExtra(String!, Char)` defined in android.content.Intent
- `putExtra(String!, CharArray!)` defined in android.content.Intent
- `putExtra(String!, CharSequence!)` defined in android.content.Intent
- `putExtra(String!, Double)` defined in android.content.Intent
- `putExtra(String!, DoubleArray!)` defined in android.content.Intent
- `putExtra(String!, Float)` defined in android.content.Intent
- `putExtra(String!, FloatArray!)` defined in android.content.Intent
- `putExtra(String!, Int)` defined in android.content.Intent
- `putExtra(String!, IntArray!)` defined in android.content.Intent
- `putExtra(String!, Long)` defined in android.content.Intent
- `putExtra(String!, LongArray!)` defined in android.content.Intent
- `putExtra(String!, Short)` defined in android.content.Intent
- `putExtra(String!, ShortArray!)` defined in android.content.Intent
- `putExtra(String!, String!)` defined in android.content.Intent

Nah, selain tipe data String, kita juga bisa melempar data dengan tipe data lainnya. Panduannya bisa kamu periksa di gambar ya!

Sebagian besar adalah primitive data type seperti Int, Long, Short, dll.

Namun ada beberapa yang mungkin baru kita lihat yaitu tipe data **Bundle**, **Serializable**, dan **Parcelable**!

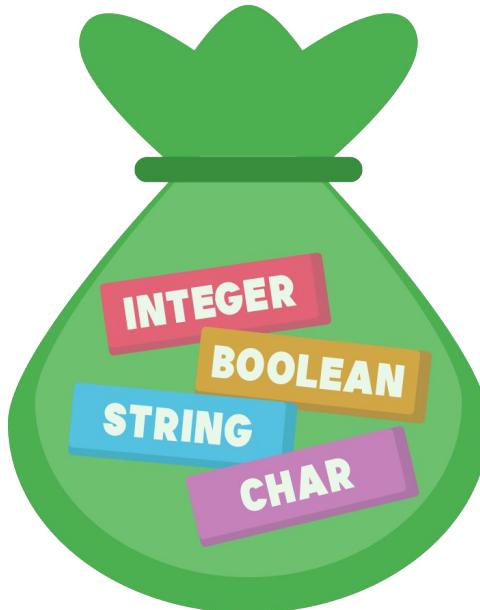
Yuk kita bahas satu per satu!



Katanya, salah satu cara mengirim data pada Intent itu pakai **Bundle.**

Tapi.. apa bundle ini mirip kayak bundle item yang biasa kita beli di e-commerce?

Yuk kita cari tahu dah~



Apa sih Bundle itu?

Android memiliki class yang disebut Bundle. Pada class ini, kita dapat menyimpan data dan **mendukung beberapa tipe data** seperti String, Char, Boolean, Integer, dan sebagainya.

Alih-alih menggunakan "Direct Intent" sebagai wadah data, kita dapat menyimpan data kita langsung ke dalam Bundle dan kemudian menyimpan Bundle ke dalam Intent.



```
...  
val intent = Intent(this@MainActivity, SecondActivity::class.java)  
val bundle = Bundle()  
  
bundle.putString("key", "value")  
intent.putExtras(bundle)  
startActivity(intent)
```

Terus, Cara Deklarasi Intent dengan Bundle?

Proses ini relatif sama dengan Direct Method. Bedanya, dalam kasus ini data disimpan ke dalam Bundle.

Kode kita di MainActivity kurang lebih seperti yang ada pada gambar di sebelah.



```
val Bundle = Intent.extras  
val name = Bundle.getString("key")
```

Di sini, kita membuat Intent dan juga menciptakan Bundle untuk menyimpan data.

Setelah itu, masukkan Bundle ke dalam Intent sehingga Intent dapat membawanya ke SecondActivity.

Kemudian di SecondActivity, kita terima dengan code yang ada di gambar di kiri ini.



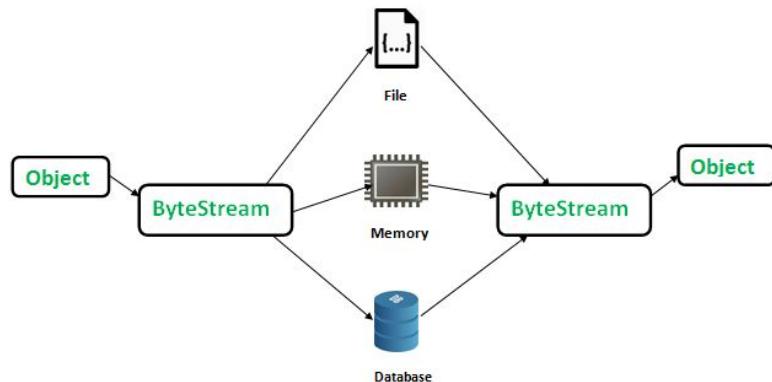
Hmm.. mirip-mirip tipis deh sama bundle item di keranjang e-commerce kita ya.

Lanjut ke **Serializable!**

Waduh.. kalo dari namanya sih millennial bin kekinian banget ya.

Serialization

De-Serialization



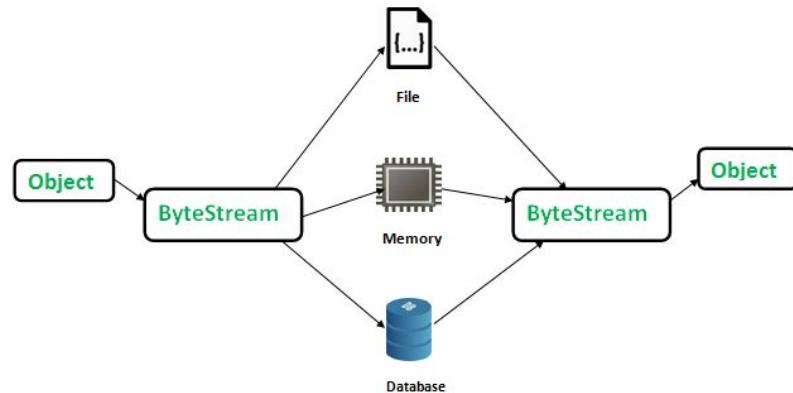
Apa itu Serializable?

Metode `Serializable` adalah metode universal, yang sudah diimplementasikan pada bahasa pemrograman Java.

Dengan menggunakan `Serializable`, kita dapat **mengirimkan object dari suatu class melalui Intent**.

Serialization

De-Serialization



Kenapa bisa mengirimkan object dari suatu class melalui Intent? Karena metode serializable ini mengubah suatu object menjadi data serial.

Cara menggunakan serializable ini cukup dengan extend class Serializable pada class yang akan di serializable.



Apa bedanya Serializable di Java dan Kotlin?

Pada Java class, isinya hanya memuat data-data biasa. Sedangkan tujuannya hanya untuk menyimpan dan mengambil data yang disebut POJO (Plain Old Java Object).

Pada Kotlin, sudah disederhanakan menjadi Data Class yang lebih ringkas dengan ditiadakannya method setter-getter.



```
● ● ●  
data class Person(val name: String, val email: String)
```



```
● ● ●  
data class Person(val name: String, val email: String): Serializable
```

Sebagai contoh, kita mempunyai Data Class bernama Person:

```
data class Person(val name: String, val email: String)
```

Untuk membuat Data Class tersebut Serializable, cukup membuat class tersebut Extend ke Interface Serializable dari java.io.Serializable menjadi seperti ini:

```
data class Person(val name: String, val email: String): Serializable
```



```
val Intent = Intent(this, SecondActivity::class.java)
val person = Person("Binarian", "binarian@binar.co.id")
Intent.putExtra("AN_OBJECT", person)
startActivity(Intent)
```

Parsing data Serializable
Dengan sintaks **as**

```
val person = Intent.getSerializableExtra("AN_OBJECT") as Person
```

Untuk mengirimkan class yang sudah Serializable melalui Intent, cukup dengan perintah seperti pada kode di samping.

Kita tidak akan mendapatkan error pada baris kode yang memasukkan object dari class **Person** karena Class Person **sudah extend ke Serializable**.



```
val Intent = Intent(this, SecondActivity::class.java)
val person = Person("Binarian", "binarian@binar.co.id")
Intent.putExtra("AN_OBJECT", person)
startActivity(Intent)
```

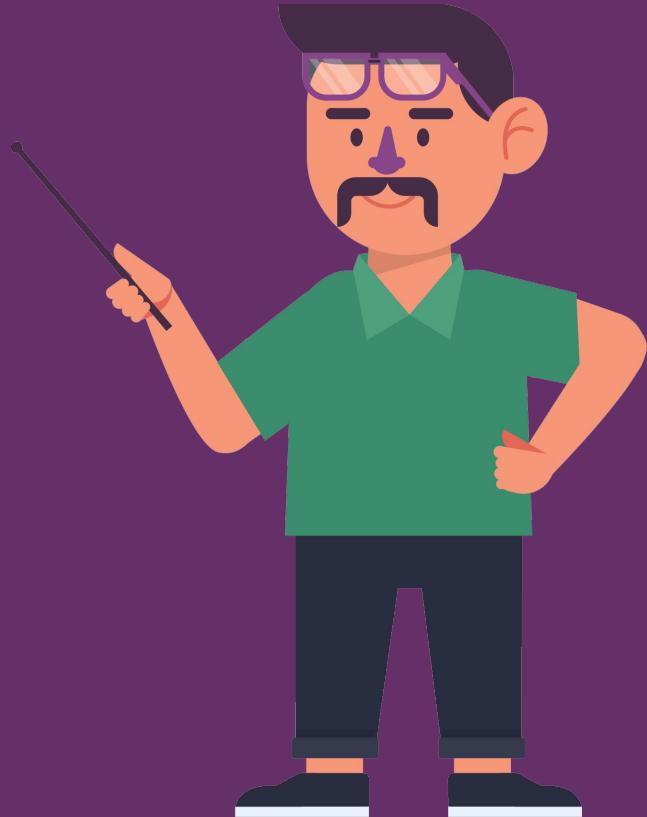
Parsing data Serializable
Dengan sintaks **as**



```
val person = Intent.getSerializableExtra("AN_OBJECT") as Person
```

Untuk menerima object class person pada SecondActivity, kita perlu melakukan Parsing data serializable menjadi object Person kembali dengan syntax **as**. Contohnya kayak dibawah ini:

```
val person =
Intent.getSerializableExtra("AN_OBJECT") as Person
```



Setelah `Serializable`, ada satu lagi nih yaitu `Parcelable`.

Nama yang bagus yaa buat nama olshop, eh tapi bukan itu fokus kita. Kita bahas buat Android. Fiuhh.. stay focus.



Apa itu Parcelable?

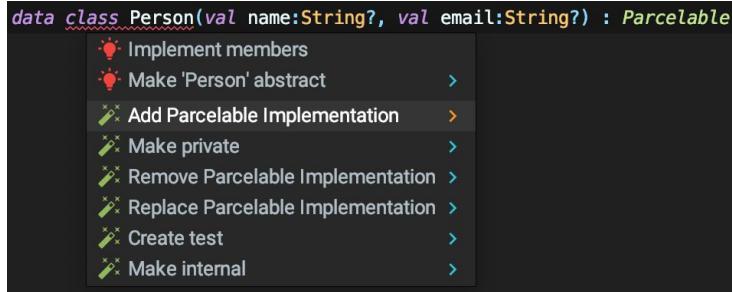
Parcelable hampir mirip dengan Serializable, yaitu memungkinkan kita untuk mengirim Object melalui Intent.

Cara menggunakan parcelable ini dengan extend class Parcelable pada class yang akan di serializable.



Cara Deklarasi Parcelable?

Untuk mengimplementasikan **Parcelable**, cukup tekan **Alt+Enter**, lalu pilih Add **Parcelable Implementation**:



Lalu, secara mejik, Android Studio akan meng-generate code Parcelable untuk class yang kita buat:

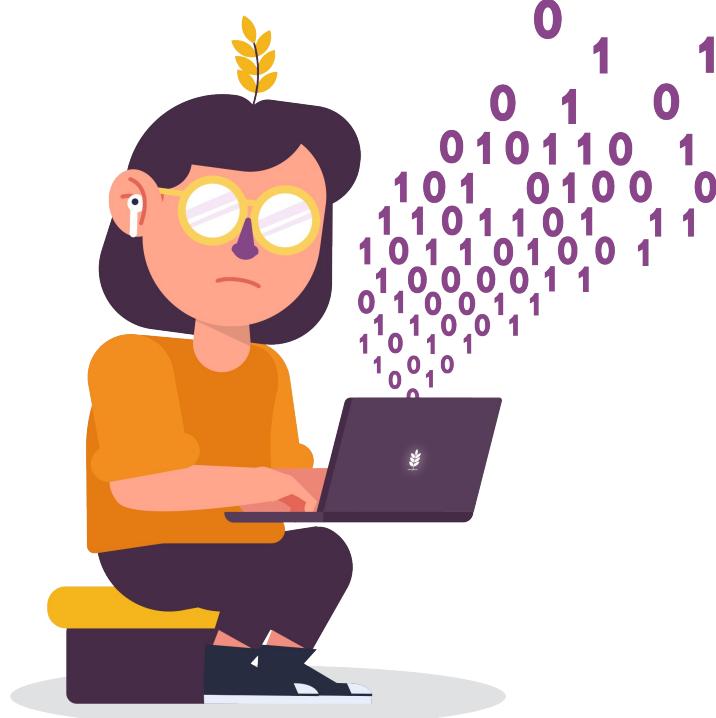
```
data class Person(val name:String?, val email:String?) : Parcelable {
    constructor(parcel: Parcel) : this(
        parcel.readString(),
        parcel.readString()
    )

    override fun writeToParcel(parcel: Parcel, flags: Int) {
        parcel.writeString(name)
        parcel.writeString(email)
    }

    override fun describeContents(): Int {
        return 0
    }

    companion object CREATOR : Parcelable.Creator<Person> {
        override fun createFromParcel(parcel: Parcel): Person {
            return Person(parcel)
        }

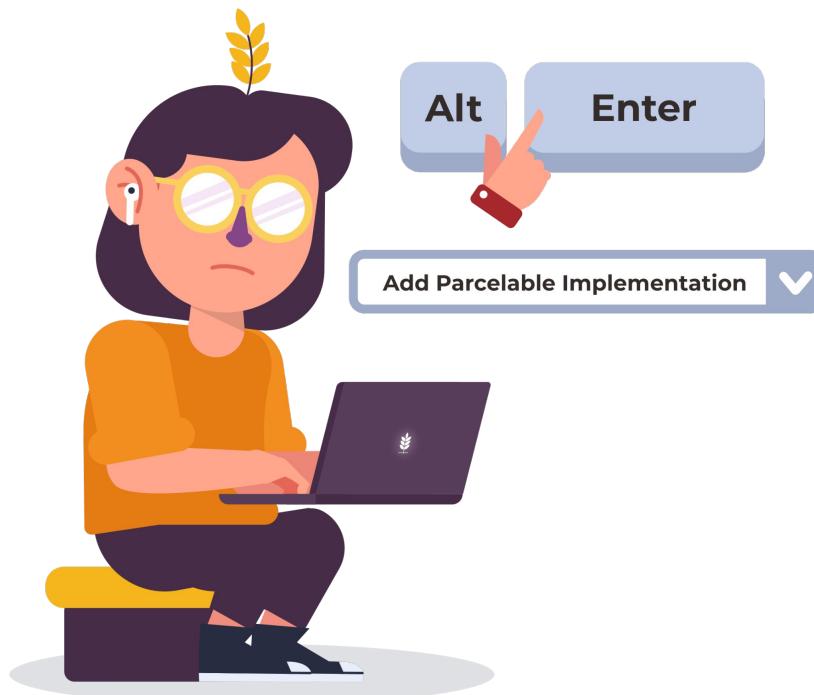
        override fun newArray(size: Int): Array<Person?> {
            return arrayOfNulls(size)
        }
    }
}
```



Pada Serializable, object hanya di-extend ke class serializable tanpa adanya code-code tambahan.

Nah, dalam Parcelable, kita harus **menambahkan beberapa code pada Data Class**.

Tapi, tenang! Android Studio sudah menyediakan tools khusus untuk melakukannya secara otomatis.

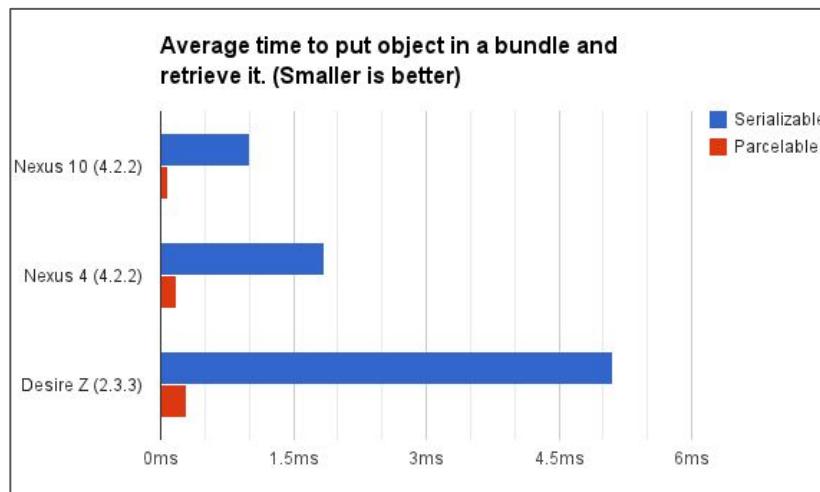


Caranya, cukup **alt+enter** error pada data class yang sudah extend ke Parcelable dan pilih **Add Parcelable Implementation**.

Maka, code-code yang diperlukan secara ajaib sudah tercipta.

Untuk menerimanya di SecondActivity, tidak terlalu berbeda dengan cara Serializable. Contohnya kayak dibawah ini :

```
Intent.getParcelableExtra("AN_OBJECT") as Person
```



Lalu, apa yang membedakan Serializable dengan Parcelable?

Jawabannya adalah **performa**.

Seperti yang kita ketahui sebelumnya, bahwa Serializable bersifat universal, sehingga semua aplikasi yang dibuat oleh Java bisa menerapkan Serializable. Jadi lebih **mengutamakan Compatibility daripada Performance**

Lain halnya dengan Parcelable. Parcelable diperuntukkan memang khusus platform Android, sehingga **performa dapat dimaksimalkan**.

Saatnya kita Quiz!





1. Berikut ini adalah pernyataan yang benar terkait intent, kecuali

- A. Implicit intent akan menghubungkan Activity dengan fitur-fitur bawaan sistem Android.
- B. Explicit intent akan mengaktifkan service atau komponen dalam satu aplikasi yang sama.
- C. Implicit intent adalah metode untuk mendeklarasikan fitur perpindahan antar screen.

1. Berikut ini adalah pernyataan yang benar terkait intent, kecuali

- A. Implicit intent akan menghubungkan Activity dengan fitur-fitur bawaan sistem Android.
- B. Explicit intent akan mengaktifkan service atau komponen dalam satu aplikasi yang sama.
- C. Implicit intent adalah metode untuk mendeklarasikan fitur perpindahan antar screen.

Hayoo jangan sampai tertukar ya antara Explicit dan Implicit Intent ya! Intent yang menavigasi hanya dalam aplikasi adalah Explicit Intent.



2. Intent mana yang memungkinkan untuk menavigasi ke luar aplikasi sesuai dengan Action Intent?

- A. Implicit Intent
- B. Explicit Intent
- C. Absolute Intent

2. Intent mana yang memungkinkan untuk menavigasi ke luar aplikasi sesuai dengan Action Intent?

- A. Implicit Intent
- B. Explicit Intent
- C. Absolute Intent

Nah, sekarang yang tepat adalah Implicit Intent karena ia bisa menavigasi ke luar aplikasi sesuai Action Intent yang ditentukan.

Jika Action Intent yang ditentukan adalah untuk mengambil foto, maka akan tampil daftar aplikasi yang bisa mengambil foto. Begitu pula jika Intent actionnya untuk mengirim pesan, maka akan tampil aplikasi perpesanan seperti SMS, WhatsApp, Telegram, dll.



3. Objek mana yang memungkinkan untuk membungkus beberapa tipe data tunggal yang bisa dikirimkan melalui Intent?

- A. Bundle
- B. Parcelable
- C. Serializable

3. Objek mana yang memungkinkan untuk membungkus beberapa tipe data tunggal yang bisa dikirimkan melalui Intent?

- A. Bundle
- B. Parcelable
- C. Serializable

Bundle dapat membungkus beberapa data tunggal seperti Boolean, String, Integer, Char.



4. Cara terbaik untuk mengirimkan Object dari Data Class adalah menggunakan?

- A. Bundle
- B. Parcelable
- C. Serializable

4. Cara terbaik untuk mengirimkan Object dari Data Class adalah menggunakan?

- A. Bundle
- B. Parcelable
- C. Serializable

Yaps, seperti yang kita ketahui, bahwa mengirimkan object melalui Intent cara terbaiknya adalah dengan menggunakan Parcelable.

Parcelable lebih unggul daripada Serializable dari segi performa, namun kita perlu menambahkan beberapa code tambahan untuk menerapkan Parcelable. Tidak seperti Serializable yang hanya cukup memerlukan extend ke class Serializable.



5. Aplikasi dapat membuka galeri foto yang ada di device termasuk bentuk penerapan dari?

- A. Implicit Intent
- B. Explicit Intent
- C. Absolute Intent



5. Aplikasi dapat membuka galeri foto yang ada di device termasuk bentuk penerapan dari

- A. Implicit Intent
- B. Explicit Intent
- C. Absolute Intent

Membuka Galeri dari dalam aplikasi kita sama dengan membuka aplikasi lain. Sehingga membuka Galeri termasuk dalam Implicit Intent yes!

Referensi dan bacaan lebih lanjut~

1. [Fragment | Android Developers](#)
2. [Serializable | Android Developers](#)
3. [Android Intents - Tutorial](#)
4. [Passing data between Activities | Medium](#)
5. [Parcelable in Kotlin? Here comes Parcelize | by Joao Alves | Medium](#)
6. [Parcelable vs Serializable | AndroidPub](#)
7. [Introduction to Intent | Academia.edu](#)
8. [Mengenal Apa Itu Explicit Intent dan Implicit Intent pada Android Studio - Naskahkode](#)



Nah, selesai sudah pembahasan kita di Chapter 3 Topic 4 ini.

Selanjutnya, kita bakal bahas tentang Navigation Component.

Penasaran kayak gimana? Cus langsung ke topik selanjutnya~



Terima Kasih!



Next Topic

loading...