KOTLIN BAHASA INDONESIA

By Supriyanto

Daftar Isi

Pengenalan

A. Syntax Dasar

Pengenalan kali ini membahas tentang script dasar dari bahasa Kotlin, berikut ini adalah contoh script dasar bahasa Kotlin.

```
fun main(){

   println("Hello Kotlin")

}
```

Fun adalah sebuah function yang ada di Kotlin, fun ini adalah keyword yang harus di panggil ketika ingin memulai dengan kotlin, misalnya kalian ingin membuat method apapun, begitu juga function yang terdapat di Javascript.

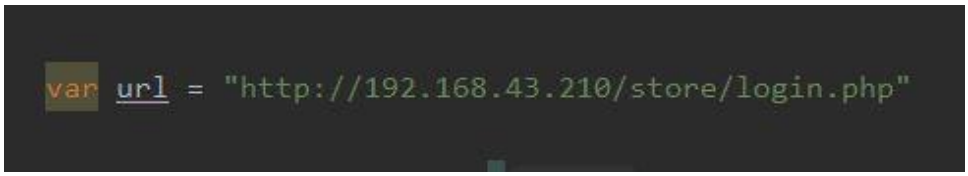
Println adalah sebuah perintah untuk mencetak isi string dengan membuat baris baru nantinya

Pada kotlin tidak perlu menggunakan tanda ; seperti pada bahasa java

Dasar Dasar

A. Tipe Data

Pada kotlin terdapat tipe data seperti halnya di Java, dengan tipe data ini kita bisa membuat sebuah variable dan menentukan dengan tipe data apa yang akan kita gunakan. Namun pada kotlin tanpa harus mendefinisikan terlebih dahulu tipe datanya di awal ketika membuat variable, beda halnya dengan bahasa Java. Contoh pada kotlin:



Pada gambar di atas kita bisa langsung membuat variabel dan mengisikan nilainya, bisa berupa String, Int dan yang lainnya, karena variabel belum didefinisikan untuk tipe datanya maka bebas nilainya.

Tipe	Keterangan
Integer	angka bulat antara -32768 s/d 32767
Shortint	bilangan bulat kecil antara -128 s/d 127
Longint	bilangan bulat dengan nilai sangat besar (9 digit)
Char	karakter tunggal dalam tanda kutip (single quote)
String	deretan karakter dalam tanda kutip (double quote)
Real	bilangan biasa antara -2.9×10^{-39} s/d $1.7 \times 10^{+38}$
Single	bilangan biasa hampir sama dengan real
Double	bilangan biasa yang jauh lebih besar dari real
Extended	bilangan biasa yang sangat besar lebih dari double
Boolean	(True or False)

Diatas adalah gambar untuk tipe data dalam bahasa pemrograman.

B. Var dan Val

Var dan Val adalah sebuah keyword pada kotlin yang digunakan untuk mendeklarasikan sebuah variabel yang akan di gunakan.

Perbedaan Var dan Val dalam bahasa Kotlin:

Var yaitu variabel umum dan itu dikenal sebagai variabel yang bisa berubah di kotlin dan dapat ditugaskan beberapa kali.

Val yaitu variabel Final dan dikenal sebagai tidak berubah di kotlin dan dapat diinisialisasi hanya satu kali

C. Control Flow

1. **If** adalah sebuah pernyataan kondisional jika pernyataan itu benar maka akan melakukan fungsi atau menampilkan informasi
2. **When** adalah keyword terbaru dari Kotlin, when ini menggantikan switch di Java, fungsi **when** mencocokkan argumennya terhadap semua cabang secara berurutan sampai beberapa kondisi cabang terpenuhi.
3. **For** adalah sebuah keyword yang berfungsi melakukan perulangan, keyword ini mirip seperti foreach. Pada for jika kondisi benar dan terpenuhi maka akan dilakukan perulangan terus menerus sampai kondisi salah.
4. **While do While** ini adalah sebuah keyword yang ada pada bahasa kotlin, untuk lebih jelasnya sudah saya jelaskan pada video bagian awal, silahkan di lihat.

D. Companion Object

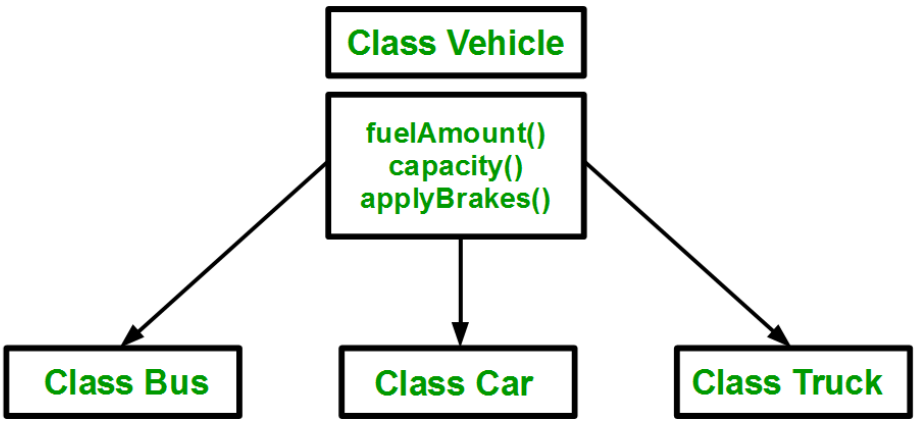
Di Kotlin, jika Anda ingin menulis fungsi atau anggota kelas yang dapat dipanggil tanpa memiliki instance kelas maka Anda dapat menulis yang sama sebagai anggota Companion Object di dalam kelas. Jadi, dengan mendeklarasikan Companion Object , Anda dapat mengakses anggota kelas dengan nama kelas saja.

Kelas dan Object

Sebenarnya kelas dan object sudah saya jelaskan secara detail di pembahasa video, namun akan saya jelaskan lagi di ebook ini biar dapat dipahami lebih detail.

A. Kelas dan Pewarisan

Untuk kelas dan object ini tidak jauh beda dengan bahasa pemrograman lain, sebut saja Java, hanya saja yang membedakan adalah struktru penulisannya saja pada bahasa masing masing. Berikut adalah gambaran kelas dan pewarisan.



Untuk kelas kita bisa langsung melihat ke tulisan atau keyword Class pada gambar diatas, yakan?. Iya itu lah kelas, sekarang kita bahas tentang inheritance atau sifat pewarisan.

Pada gambar diatas yaitu Class Vehicle memiliki property function fuelAmount() dan seterusnya, kemudian ada 3 class lagi atau bisa kita sebut class anak, yang dimana anak ini bisa mewarisi sifat dari class orang tuanya yaitu anggap saja class orang tuanya adalah Class Vehicle. Begitulah class dan pewarisan, untuk lebih detail lihat lagi video saya pada bagian OOP Kotlin.

B. Kelas dan Kelas Bersarang

Kelas adalah template yang digunakan dalam pemrograman berorientasi objek untuk membuat objek, yang merupakan contoh dari kelas itu. Mereka berguna untuk mengatur variabel dan fungsi terkait.

Kelas Bersarang adalah adalah kelas yang dibuat di dalam kelas lain. Di Kotlin, kelas bersarang secara default **statis** , sehingga anggota data dan fungsi anggota dapat diakses tanpa membuat objek kelas. Kelas bersarang tidak dapat mengakses data anggota kelas luar. Contoh:

```
class outerClass{
  //outer class code
   class nestedClass{
    //nested class code
   }
}
```

C. Akses Modifier

Akses Modifier adalah kata kunci yang digunakan untuk menentukan aksesibilitas kelas (atau tipe) dan anggotanya. Pengubah ini dapat digunakan dari kode di dalam atau di luar aplikasi saat ini. Berikut ini adalah list dari Akses Modifier

1. **Private** yaitu hanya dapat mengakses anggota tipe itu, dan karenanya aksesibilitas terbatas pada tipe saat ini
2. **Public** kode dari mana saja di dalam rakitan saat ini, atau rakitan lain yang mereferensikannya, dapat mengakses anggota jenis ini, dan karenanya memungkinkan aksesibilitas dari mana saja
3. **Protected** kode dalam tipe, atau kelas turunannya, dapat mengakses anggota tipe dan karenanya aksesibilitas terbatas pada tipe saat ini dan kelas turunannya
4. **Internal** dalam rakitan saat ini, tetapi tidak dari rakitan lain, dapat mengakses anggota tipe tersebut, karenanya aksesibilitas terbatas pada rakitan saat ini

D. Enum

Enum adalah tipe data yang memiliki seperangkat konstanta. Enum didefinisikan dengan menambahkan pengubah enumdi depan kelas seperti yang ditunjukkan di bawah ini. Contoh sebagai berikut:

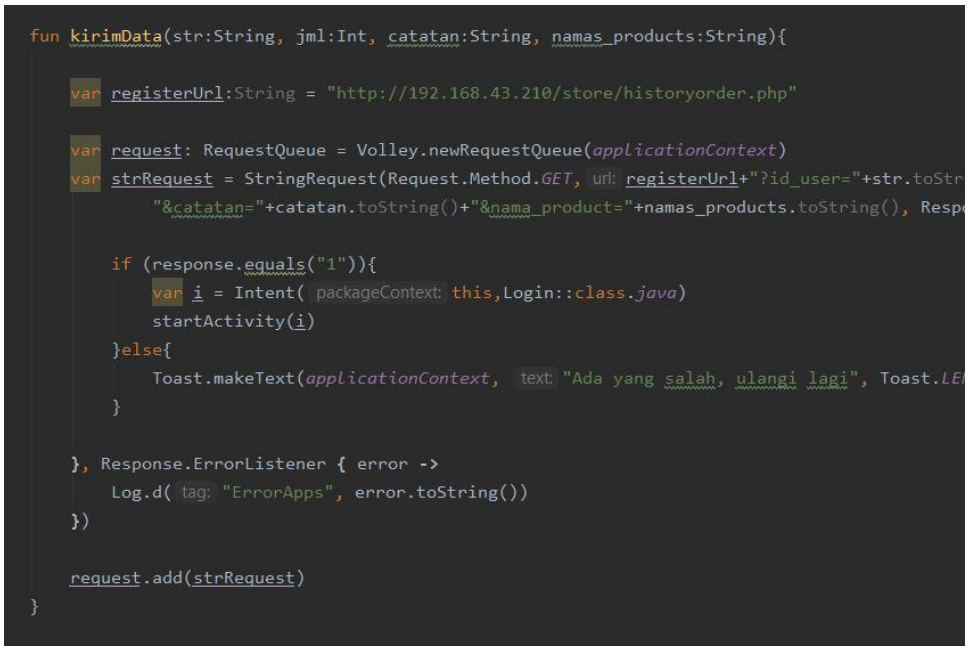
```
*/

fun main(){
    println(myEnum.COLOR)
    println(myEnum.WARNA)
}

enum class myEnum(s:String){
    WARNA( s: "HIJAU"), MOTOR( s: "VARIO"), COLOR( s: "BLUE")
}
```

Gambar diatas pada class enum memiliki properti warna, motor dan color yang bisa kita tulis tanpa harus menggunkan keyword var atau val terlebih dahulu, untuk lebih jelasnya bisa lihat lagi video tentang enum.

Fungsi dan Lambda

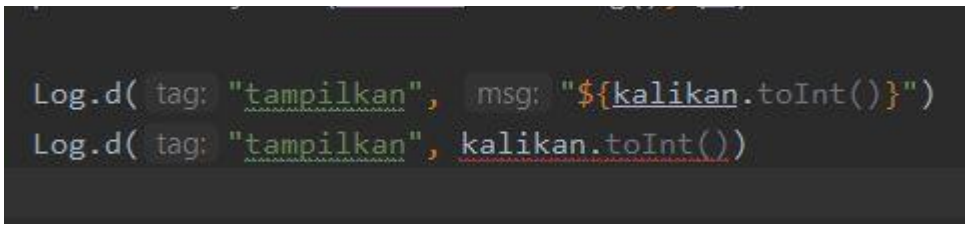
Perlu di ketahui untuk fungsi di Kotlin menggunakan keyword fun sebagaimana di JavaScript menggunakan keyword function

1. **Fungsi** adalah blok kode yang terorganisir dan dapat digunakan kembali yang digunakan untuk melakukan tindakan tunggal yang terkait. Mari lihat contoh gambar di bawah ini:



Kita lihat contoh gambar diatas bahwa saya memiliki funcion dengan nama yaitu KirimData, yang dimana dalam function tersebut saya mengisikan beberapa parameter yaitu (str:String, jml:Int, catatan:String, namas_products:String). Dalam definsi function yaitu bisa melakukan tindakan tunggal yang terkait, itu maksudnya adalah bahwa funtion ini hanya bisa bekerja dalam tunggal saja yang dimana didalamnya bisa mengeksekusi program program yang terkait seperti halnya saya menuliskan seperti gambar diatas.

2. **Lambda** pada dasarnya adalah blok kode yang dapat ditugaskan ke variabel. Contoh penulisan lambda pada bahasa kotlin adalah sebagai berikut:



Mari kita lihat contoh lambda pada gambar diatas, kita bisa lihat bahwa ada 2 baris kode yang dimana baris pertama menggunakan fungsi lambda sedangkan yang ke 2 tidak menggunannya.

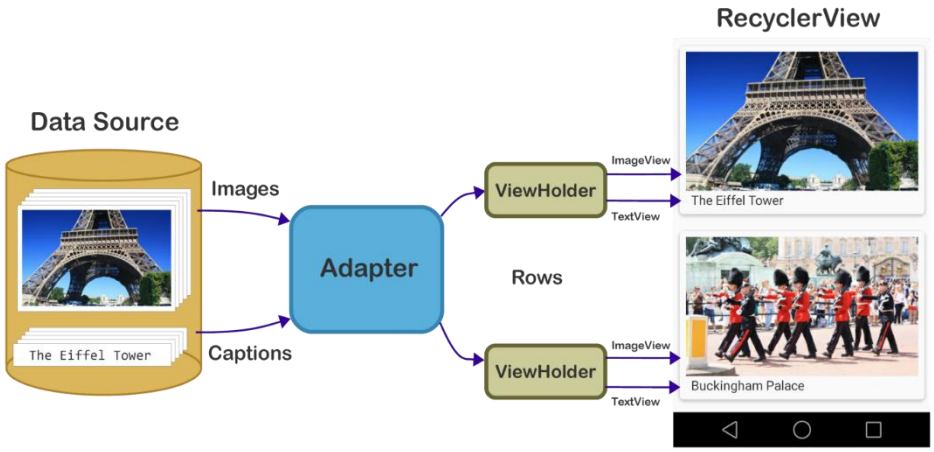
Gambar diatas baris ke 2 error karena pada Log.d seharusnya berisi String bukan Int, sehingga muncullah pesan itu, kenapa kita tidak buat toString() saja? Kalau kita buat

toSting nanti variable kalikan akan crash sehingga kita bisa memanfaatkan keyword lambda untuk mengatsi errer tersebut, sehingga variable dengan tipe Int dapat di panggil dengan lambda di dalam string.

Adapter

Dalam bab Adapter ini kita akan membahas mengenai RecyclerView, RecyclerView.Adapter, Recyclerview.ViewHolder dan masih banyak lagi lainnya.

A. **RecyclerView** merupakan versi lanjutan dari ListView dan GridView yang lebih canggih dan ringan untuk digunakan sehingga tidak memakan memori di aplikasi kita. RecyclerView ini digunakan untuk menampilkan dataset atau menampilkan jumlah data yang sangat besar.
B. **RecyclerView.Adapter** yaitu digunakan untuk penghubung antara DataSource dengan RecyclerView atau tampilan screen android kita. Mari kita lihat gambar di bawah ini.

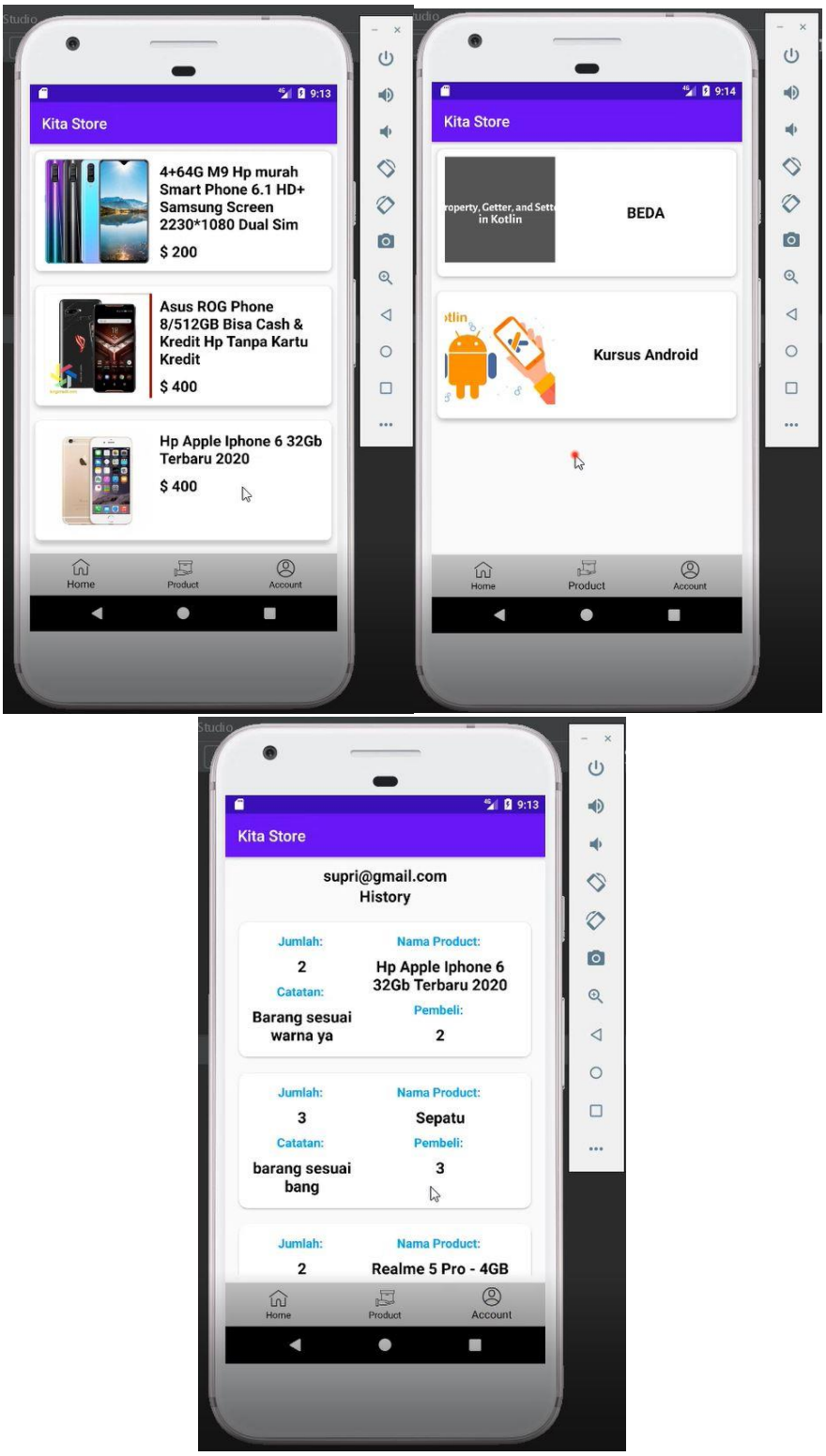


Mari kita lihat contoh gambar diatas, kita lihat bahwa Adapter diatas berposisi sebagai penghubung antara DataSource dengan RecyclerView (tampilan android screen kita), kemudian disebelah kanan dari Adapter ada ViewHolder yang berfungsi menampilkan DataSource melalui Adapter.

C. **RecyclerView.ViewHolder** adalah keyword yang berada dalam RecyclerView, keyword ini berguna atau bertugas untuk menampilkan DataSource ke screen aplikasi kita yang sebelumnya sudah di ambil melalui Adapter
D. **Constructor** adalah method khusu yang digunakan untuk menginisialisasi objek dan dipanggil ketika objek kelas dibuat.

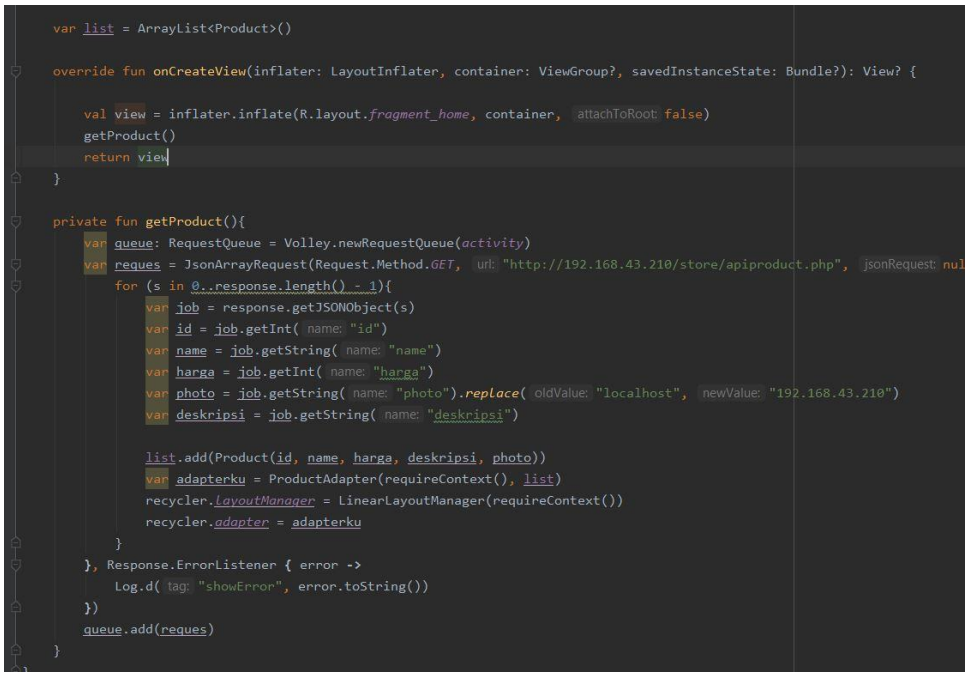
Fragment

Fragment adalah bagian dari aktivitas, juga dikenal sebagai sub-aktivitas. Mungkin ada lebih dari satu fragmen dalam suatu aktivitas. Fragmen mewakili beberapa layar dalam satu aktivitas. Kita lihat contoh Fragment pada gambar di bawah ini.



Pada gambar diatas kita hanya mempunyai 1 aktivity, namun kita bisa menambahkan 3 fragment dengan cara mengreplace, ketika bottom sheet di tekan akan membuka fragment berdasarkan idnya sebagaimana yang sudah kita bahas dalam video.

Pada aplikasi yang sudah kita buat, kita membuat fragment yang dimana didalamnya terdapat keyword seperti pada gambar di bawah ini.



```
var list = ArrayList<Product>()

override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?, savedInstanceState: Bundle?): View? {

    val view = inflater.inflate(R.layout.fragment_home, container,  attachToRoot: false)
    getProduct()
    return view
}

private fun getProduct(){
    var queue: RequestQueue = Volley.newRequestQueue(activity)
    var reques = JsonArrayRequest(Request.Method.GET,  url: "http://192.168.43.210/store/apiproduct.php",  jsonRequest: nul
        for (s in 0..response.length() - 1){
            var job = response.getJSONObject(s)
            var id = job.getInt( name: "id")
            var name = job.getString( name: "name")
            var harga = job.getInt( name: "harga")
            var photo = job.getString( name: "photo").replace( oldValue: "localhost",  newValue: "192.168.43.210")
            var deskripsi = job.getString( name: "deskripsi")

            list.add(Product(id, name, harga, deskripsi, photo))
            var adapterku = ProductAdapter(requireContext(), list)
            recycler.layoutManager = LinearLayoutManager(requireContext())
            recycler.adapter = adapterku
        }
    }, Response.ErrorListener { error ->
        Log.d( tag: "showError", error.toString())
    })
    queue.add(reques)
}
```

A. **LayoutInflanter** digunakan untuk mengambil isi file XML layout ke objek View yang sesuai yang terletak di onCreateView. Untuk contoh ada pada gambar diatas.
B. **ViewGroup** bisa kita katakan gabungan view antara parent dengan child yang biasanya kita gunakan di dalam class fragment dengan attachroot false.

Pada gambar diatas kita isikan untuk paramter pada inflate menjadi 3 parameter, yaitu FragmentHome, container dan false (LayoutInflanter, ViewGroup dan AttachRoot) attachroot adalah bagian xml yang ada di xml fragment_home, ketika kita buat menjadi false maka tidak terjadi error, jika kita buat menjadi true maka aplikasi akan crash disebabkan berbenturan antara xml parent dengan child.

C. **Bundle (savedInstanceState)** yaitu digunakan untuk menyimpan data hanya untuk sementara atau aplikasi Data disimpan dalam memori hanya sampai aplikasi hidup, dengan kata lain data ini hilang ketika aplikasi ditutup.