

LAPORAN PRAKTIKUM PENGEMBANGAN APLIKASI BERGERAK

Pengenalan Kotlin 2 - WEEK 2



Disusun oleh:

Nama : Alfath Roziq Widhayaka

Nim : L0122012

Kelas : A

PROGRAM STUDI INFORMATIKA

FAKULTAS TEKNOLOGI INFORMASI DAN SAINS DATA

UNIVERSITAS SEBELAS MARET


2024

1. Screenshot Source Code

Source code ini merupakan implementasi program sederhana yang digunakan dalam praktikum minggu kedua. Program ini bertujuan untuk mengelola stok mobil di sebuah dealer dengan fitur tambah stok mobil, tampilkan stok mobil, dan cari mobil berdasarkan tahun produksi. Materi yang digunakan pada source code dibawah ini terdapat Control flow (when expression dan range), Inheritance, Interface, Collections, Lambda Expression, dan Lateint Lazy Property.

A. Control Flow

▪ When Expression

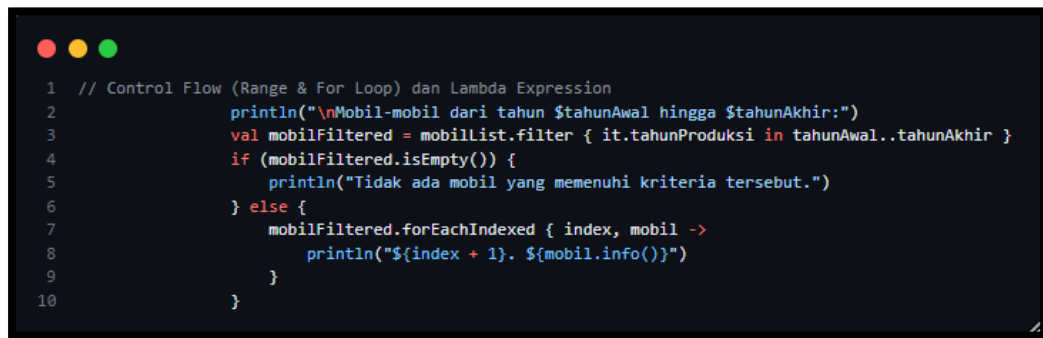


```
1 // Control flow (when expression)
2 when (scanner.nextInt()) {
3     1 -> {
4         println("\nTambahkan stok mobil:")
5         print("Merek      : ")
6         val merek = scanner.next()
7         print("Model      : ")
8         val model = scanner.next()
9         print("Tahun Produksi : ")
10        val tahun = scanner.nextInt()
11        print("Jumlah Pintu  : ")
12        val pintu = scanner.nextInt()
13        print("Nomor Mesin   : ")
14        val nomorMesin = scanner.next()
15
16        val mobil = Mobil(merek, model, tahun, pintu)
17        mobil.nomorMesin = nomorMesin
18
19        mobilList.add(mobil)
20
21        println("\n!!Mobil berhasil ditambahkan!!")
22    }
23    2 -> {
24        println("\nDaftar Mobil di Dealer:")
25        if (mobilList.isEmpty()) {
26            println("Stok mobil kosong.")
27        } else {
28            println("-----")
29            println("| No |      Merek      |      Mod
```

Gambar 1 when expression

Pada kode di atas, **when expression** digunakan untuk menentukan tindakan yang akan diambil berdasarkan input yang diterima dari pengguna melalui scanner pada tampilan menu. Dari kode diatas program aslinya memiliki 4 pilihan pada menu namun, dikarenakan terbatasnya ruang maka yang terlihat hanya sampai 2 pilihan. Setelah pengguna melakukan input maka program akan mengevaluasi input dari pengguna (yang didapat dari scanner.nextInt()) dan akan mengeksekusi blok kode yang sesuai dengan nilai input tersebut.

- Range

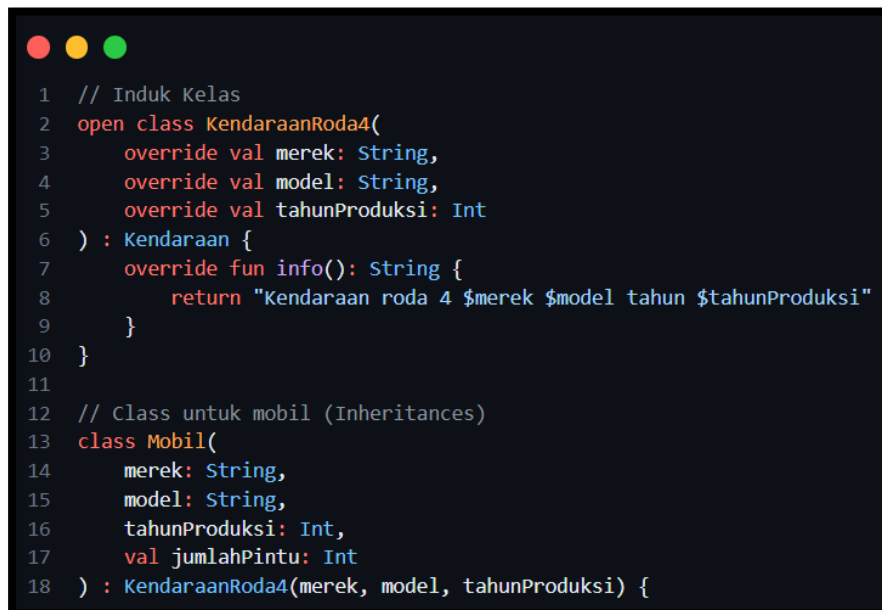


```
1 // Control Flow (Range & For Loop) dan Lambda Expression
2 println("\nMobil-mobil dari tahun $tahunAwal hingga $tahunAkhir:")
3 val mobilFiltered = mobilList.filter { it.tahunProduksi in tahunAwal..tahunAkhir }
4 if (mobilFiltered.isEmpty()) {
5     println("Tidak ada mobil yang memenuhi kriteria tersebut.")
6 } else {
7     mobilFiltered.forEachIndexed { index, mobil ->
8         println("${index + 1}. ${mobil.info()}")
9     }
10 }
```

Gambar 2 range

Pada kode di atas, **Range** digunakan dalam proses pencarian mobil berdasarkan tahun produksi yaitu pada kode **val mobilFiltered = mobilList.filter { it.tahunProduksi in tahunAwal..tahunAkhir }**. Pada baris di atas, tahunAwal..tahunAkhir membuat rentang tahun produksi yang diinginkan, sehingga nanti saat ditampilkan pada output maka akan muncul daftar mobil yang tersedia pada pembuatan tahun sekian sampai sekian.

B. Inheritances



```
1 // Induk Kelas
2 open class KendaraanRoda4(
3     override val merek: String,
4     override val model: String,
5     override val tahunProduksi: Int
6 ) : Kendaraan {
7     override fun info(): String {
8         return "Kendaraan roda 4 $merek $model tahun $tahunProduksi"
9     }
10 }
11
12 // Class untuk mobil (Inheritances)
13 class Mobil(
14     merek: String,
15     model: String,
16     tahunProduksi: Int,
17     val jumlahPintu: Int
18 ) : KendaraanRoda4(merek, model, tahunProduksi) {
```

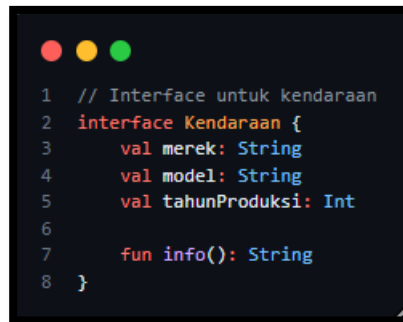
Gambar 3 inheritances

Di kode yang diberikan, **inheritance (pewarisan)** digunakan untuk mewariskan properti dan fungsi dari satu kelas ke kelas lain. Dalam hal ini, terdapat dua kelas yang terlibat dalam pewarisan:

- Kelas KendaraanRoda4 : Kelas ini adalah kelas induk dan mendeklarasikan merek, model, dan tahunProduksi yang diwariskan kepada kelas turunannya. kelas ini memiliki fungsi info() yang akan diimplementasikan oleh kelas turunannya.
- Kelas Mobil : Kelas ini adalah kelas turunan atau subclass dari KendaraanRoda4. Menggunakan inheritance untuk mewarisi merek, model,

dan tahunProduksi dari kelas KendaraanRoda4. Kelas ini juga menambahkan properti jumlahPintu dan nomor mesin yang spesifik untuk mobil.

C. Interface



```
1 // Interface untuk kendaraan
2 interface Kendaraan {
3     val merek: String
4     val model: String
5     val tahunProduksi: Int
6
7     fun info(): String
8 }
```

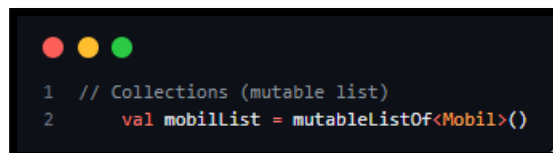
Gambar 4 interface

Dengan menggunakan **interface Kendaraan**, dapat ditentukan bahwa setiap kelas yang mengimplementasikan interface tersebut harus menyediakan properti dan fungsi sesuai dengan yang diberikan oleh interface tersebut. Dalam hal ini, kelas Mobil adalah satu-satunya kelas yang mengimplementasikan interface Kendaraan.

Interface pada kode diatas memiliki tiga pendeklarasian dan satu fungsi yaitu :

- merek: Menyimpan merek kendaraan.
- model: Menyimpan model kendaraan.
- tahunProduksi: Menyimpan tahun produksi kendaraan.
- info(): Fungsi yang memberikan informasi umum tentang kendaraan.

D. Collections

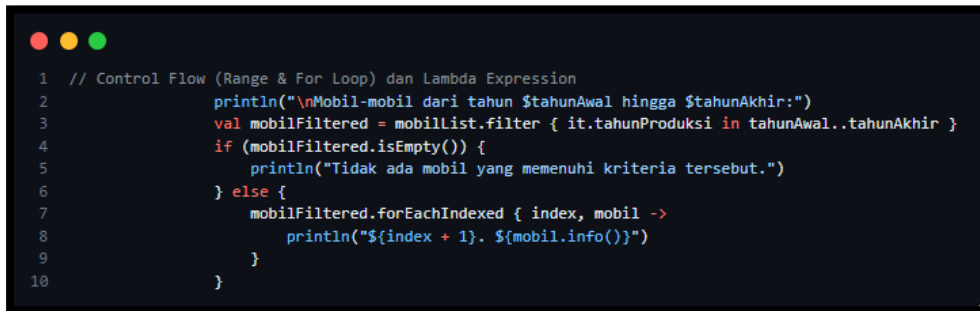


```
1 // Collections (mutable list)
2 val mobilList = mutableListOf<Mobil>()
```

Gambar 5 collections

Collections pada kode di atas digunakan untuk menyimpan dan mengelola data tentang mobil-mobil yang ada di showroom. MutableList digunakan untuk menyimpan objek-objek Mobil yang ditambahkan ke dalam showroom. MutableList memungkinkan untuk menambahkan, menghapus, atau mengubah elemen-elemennya setelah dibuat. Setiap kali pengguna memilih opsi untuk menambahkan stok mobil (Menu: 1), objek Mobil baru akan dibuat dan ditambahkan ke dalam mobilList.

E. Lambda Expression

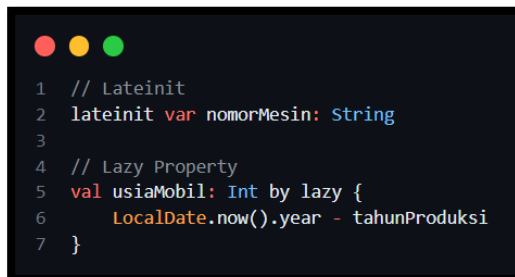


```
1 // Control Flow (Range & For Loop) dan Lambda Expression
2 println("\nMobil-mobil dari tahun $tahunAwal hingga $tahunAkhir:")
3 val mobilFiltered = mobilList.filter { it.tahunProduksi in tahunAwal..tahunAkhir }
4 if (mobilFiltered.isEmpty()) {
5     println("Tidak ada mobil yang memenuhi kriteria tersebut.")
6 } else {
7     mobilFiltered.forEachIndexed { index, mobil ->
8         println("${index + 1}. ${mobil.info()}")
9     }
10 }
```

Gambar 6 lambda expression

Program ini menggunakan **Lambda expression** di dalam fungsi filter. Fungsi filter digunakan untuk menyaring elemen-elemen dalam koleksi (mobilList dalam hal ini) berdasarkan suatu kriteria tertentu yang ditentukan oleh lambda expression. Lambda expression di atas adalah { it.tahunProduksi in tahunAwal..tahunAkhir }. Ini memiliki parameter tunggal disebut it yang mewakili setiap elemen dalam koleksi (Mobil dalam hal ini). Lambda expression ini akan mengembalikan true jika tahunProduksi mobil berada dalam rentang tahun yang ditentukan (tahunAwal..tahunAkhir), sehingga mobil akan disertakan dalam koleksi mobilFiltered.

F. Lateinit & Lazy Property



```
1 // Lateinit
2 lateinit var nomorMesin: String
3
4 // Lazy Property
5 val usiaMobil: Int by lazy {
6     LocalDate.now().year - tahunProduksi
7 }
```

Gambar 7 lateinit & lazy property

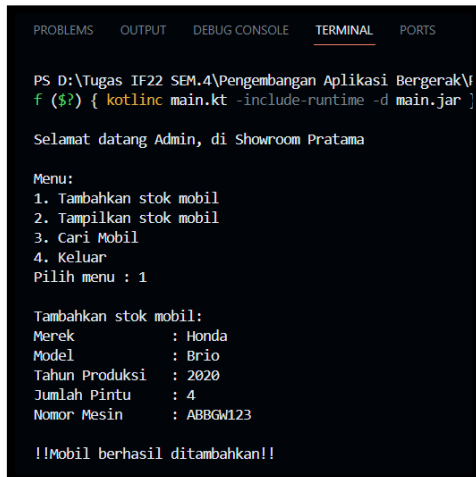
- Program diatas menggunakan **lateinit** untuk menunda inialisasi variabel hingga saat runtime. Di sini, **nomorMesin** adalah properti dari kelas Mobil. Properti ini tidak diinisialisasi pada saat deklarasi karena nilai nomorMesin akan dimasukkan oleh pengguna melalui input. Oleh karena itu, menggunakan lateinit memungkinkan untuk menunda inialisasi hingga nilai sebenarnya dimasukkan oleh pengguna.
- Program diatas menggunakan **lazy property** untuk menunda evaluasi dari suatu ekspresi hingga dibutuhkan untuk pertama kalinya. Di sini, usiaMobil adalah properti yang dihitung berdasarkan tahun produksi mobil. Nilai properti ini dihitung secara lazy, artinya nilai akan dihitung hanya saat properti tersebut pertama kali diakses. Dengan kata lain, ekspresi **LocalDate.now().year - tahunProduksi** tidak akan dievaluasi sampai **usiaMobil** pertama kali diakses dalam program. lazy sangat berguna untuk mengoptimalkan kinerja dan penggunaan memori, terutama ketika properti tersebut mungkin tidak akan digunakan atau diakses sepanjang waktu aplikasi berjalan.

2. Screenshot Terminal

Berikut adalah hasil pada terminal apabila source code dijalankan pada program yang saya miliki ini. Gambar-gambar dibawah ini merupakan hasil yang menerapkan materi-materi yang diminta.

Hasil Output pada terminal :

1. Menu pilihan 1



```
PS D:\Tugas IF22 SEM.4\Pengembangan Aplikasi Bergerak\
f ($?) { kotlinc main.kt -include-runtime -d main.jar }

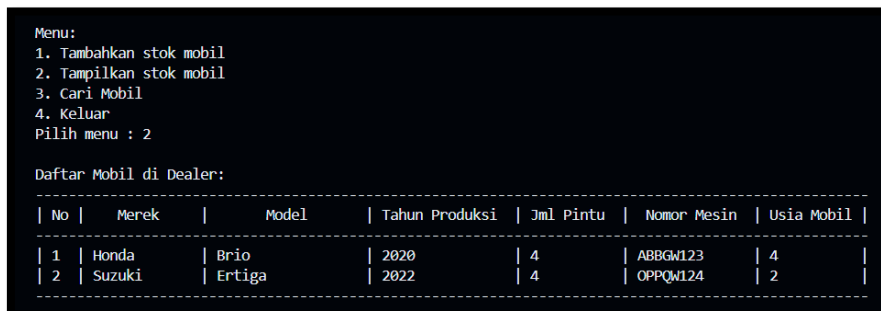
Selamat datang Admin, di Showroom Pratama

Menu:
1. Tambahkan stok mobil
2. Tampilkan stok mobil
3. Cari Mobil
4. Keluar
Pilih menu : 1

Tambahkan stok mobil:
Merek      : Honda
Model      : Brio
Tahun Produksi : 2020
Jumlah Pintu : 4
Nomor Mesin : ABBGW123

!!Mobil berhasil ditambahkan!!
```

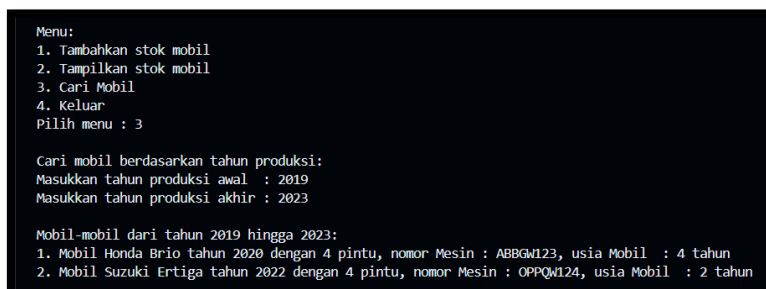
2. Menu pilihan 2



```
Menu:
1. Tambahkan stok mobil
2. Tampilkan stok mobil
3. Cari Mobil
4. Keluar
Pilih menu : 2

Daftar Mobil di Dealer:
-----
| No | Merek | Model | Tahun Produksi | Jml Pintu | Nomor Mesin | Usia Mobil |
-----
| 1 | Honda | Brio | 2020 | 4 | ABBGW123 | 4 |
| 2 | Suzuki | Ertiga | 2022 | 4 | OPPQW124 | 2 |
-----
```

3. Menu pilihan 3

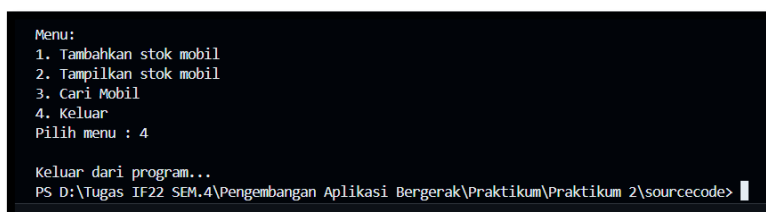


```
Menu:
1. Tambahkan stok mobil
2. Tampilkan stok mobil
3. Cari Mobil
4. Keluar
Pilih menu : 3

Cari mobil berdasarkan tahun produksi:
Masukkan tahun produksi awal : 2019
Masukkan tahun produksi akhir : 2023

Mobil-mobil dari tahun 2019 hingga 2023:
1. Mobil Honda Brio tahun 2020 dengan 4 pintu, nomor Mesin : ABBGW123, usia Mobil : 4 tahun
2. Mobil Suzuki Ertiga tahun 2022 dengan 4 pintu, nomor Mesin : OPPQW124, usia Mobil : 2 tahun
```

4. Menu pilihan 4



```
Menu:
1. Tambahkan stok mobil
2. Tampilkan stok mobil
3. Cari Mobil
4. Keluar
Pilih menu : 4

Keluar dari program...
PS D:\Tugas IF22 SEM.4\Pengembangan Aplikasi Bergerak\Praktikum\Praktikum 2\sourcecode>
```

Penjelasan masing-masing materi :

A. Control Flow

- When Expression

```
Selamat datang Admin, di Showroom Pratama

Menu:
1. Tambahkan stok mobil
2. Tampilkan stok mobil
3. Cari Mobil
4. Keluar
Pilih menu :
```

Kegunaan **when expression** disini terlihat pada gambar diatas yaitu untuk menampilkan fitur menu 1 sampai 4.

- Range

```
Menu:
1. Tambahkan stok mobil
2. Tampilkan stok mobil
3. Cari Mobil
4. Keluar
Pilih menu : 3

Cari mobil berdasarkan tahun produksi:
Masukkan tahun produksi awal : 2019
Masukkan tahun produksi akhir : 2023

Mobil-mobil dari tahun 2019 hingga 2023:
1. Mobil Honda Brio tahun 2020 dengan 4 pintu, nomor Mesin : ABBGW123, usia Mobil : 4 tahun
2. Mobil Suzuki Ertiga tahun 2022 dengan 4 pintu, nomor Mesin : OPPQW124, usia Mobil : 2 tahun
```

Kegunaan **range** disini digunakan untuk mencari mobil berdasarkan tahun produksi sekian sampai sekian, misal pengguna ingin mencari mobil tahun produksi 2019 sampai 2023. Maka hasil yang ditampilkan akan sesuai yang diminta.

B. Inheritance

```
Selamat datang Admin, di Showroom Pratama

Menu:
1. Tambahkan stok mobil
2. Tampilkan stok mobil
3. Cari Mobil
4. Keluar
Pilih menu :
```

Inheritance dalam program diatas dihasilkan pada output tambahkan stok mobil, tampilkan stok mobil atau pada pengelolaan mobil. Konsep inheritances ini diperlihatkan dengan kelas **Mobil** yang mewarisi sifat-sifat dari kelas **KendaraanRoda4**

C. Interface

Dengan menggunakan **interface Kendaraan**, dapat ditentukan bahwa setiap kelas yang mengimplementasikan interface tersebut harus menyediakan properti dan fungsi sesuai dengan yang diberikan oleh interface tersebut.

D. Collections

```
Menu:
1. Tambahkan stok mobil
2. Tampilkan stok mobil
3. Cari Mobil
4. Keluar
Pilih menu : 2

Daftar Mobil di Dealer:
-----
| No | Merek | Model | Tahun Produksi | Jml Pintu | Nomor Mesin | Usia Mobil |
-----
| 1 | Honda | Brio | 2020 | 4 | ABBGW123 | 4 |
| 2 | Suzuki | Ertiga | 2022 | 4 | OPPQM124 | 2 |
-----
```

Dengan menggunakan **collections**, program menggunakan sebuah **mutableListOf<Mobil>()** untuk menyimpan daftar stok mobil.

E. Lambda Expression

```
Menu:
1. Tambahkan stok mobil
2. Tampilkan stok mobil
3. Cari Mobil
4. Keluar
Pilih menu : 3

Cari mobil berdasarkan tahun produksi:
Masukkan tahun produksi awal : 2019
Masukkan tahun produksi akhir : 2023

Mobil-mobil dari tahun 2019 hingga 2023:
1. Mobil Honda Brio tahun 2020 dengan 4 pintu, nomor Mesin : ABBGW123, usia Mobil : 4 tahun
2. Mobil Suzuki Ertiga tahun 2022 dengan 4 pintu, nomor Mesin : OPPQM124, usia Mobil : 2 tahun
```

Pada bagian "Cari mobil berdasarkan tahun produksi", program menggunakan **Lambda Expression** untuk melakukan filtering terhadap mobil-mobil yang memenuhi kriteria tahun produksi yang dimasukkan pengguna.

F. Lateinit & Lazy Property

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS D:\Tugas IF22 SEM.4\Pengembangan Aplikasi Bergerak\>
f ($?) { kotlinc main.kt -include-runtime -d main.jar }

Selamat datang Admin, di Showroom Pratama

Menu:
1. Tambahkan stok mobil
2. Tampilkan stok mobil
3. Cari Mobil
4. Keluar
Pilih menu : 1

Tambahkan stok mobil:
Merek      : Honda
Model      : Brio
Tahun Produksi : 2020
Jumlah Pintu   : 4
Nomor Mesin  : ABBGW123

!!Mobil berhasil ditambahkan!!
```

Lateinit digunakan pada variabel **nomorMesin** yang ada di kelas **Mobil**. Properti ini diperlukan karena nilai dari **nomorMesin** akan diinisialisasi setelah objek **Mobil** dibuat.


```
Menu:
1. Tambahkan stok mobil
2. Tampilkan stok mobil
3. Cari Mobil
4. Keluar
Pilih menu : 2
```

Daftar Mobil di Dealer:

No	Merek	Model	Tahun Produksi	Jml Pintu	Nomor Mesin	Usia Mobil
1	Honda	Brio	2020	4	ABBGM123	4
2	Suzuki	Ertiga	2022	4	OPPQM124	2

usiaMobil di kelas Mobil dideklarasikan sebagai **lazy property**. Ini berarti nilai dari **usiaMobil** tidak akan dihitung sampai properti tersebut pertama kali diakses.