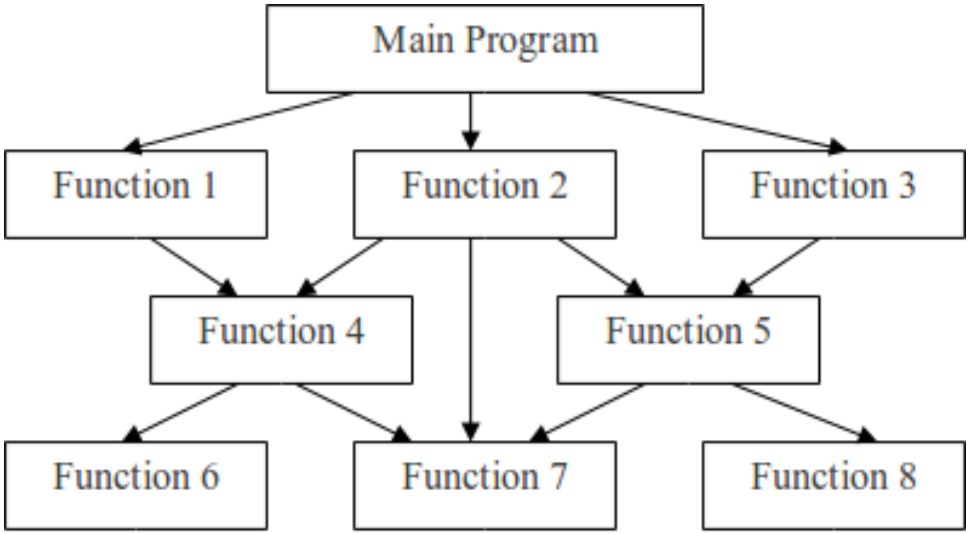
Pemrograman Berorientasi Obyek

Pertemuan 2: Pemrograman Terstruktur VS Pemrograman Berorientasi Obyek

Pemrograman Terstruktur

- Disebut juga Pemrograman Berorientasi Prosedur
- Permasalahan ditinjau berdasarkan urutan kegiatan penyelesaian (misal: membaca data, menghitung data, mencetak data)
- Sejumlah prosedur/fungsi akan dibuat untuk melakukan urutan kegiatan tersebut, sehingga fokus utamanya adalah pada FUNGSI



Pemrograman Terstruktur

- Terdiri atas serangkaian instruksi yang harus dilakukan komputer dalam bentuk FLOWCHART
- Tidak terlalu memperhatikan data, bagaimana data terpengaruhi oleh prosedur/fungsi yang menggunakannya

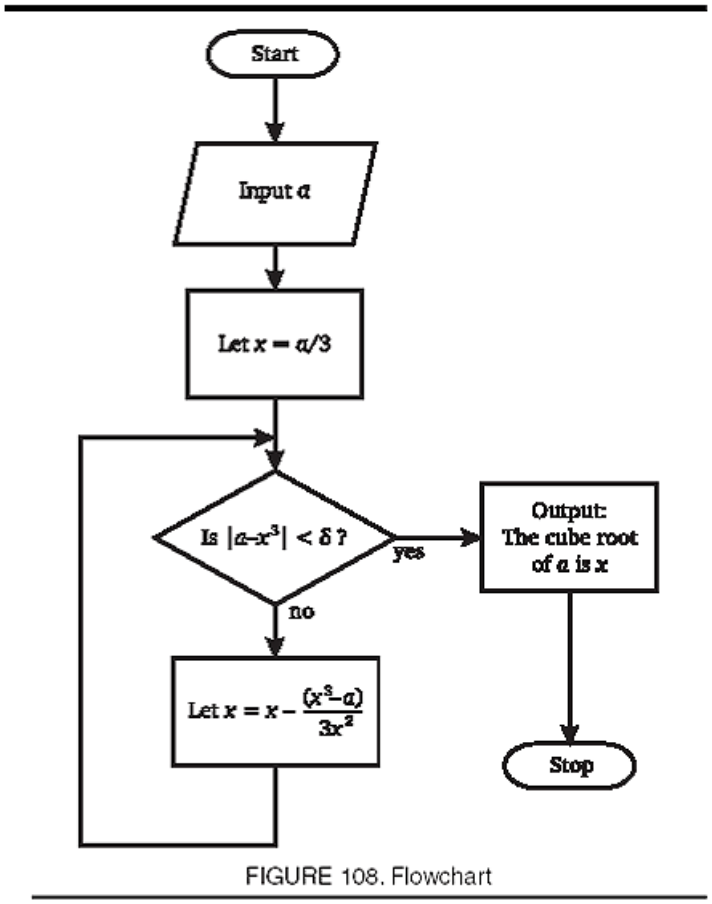
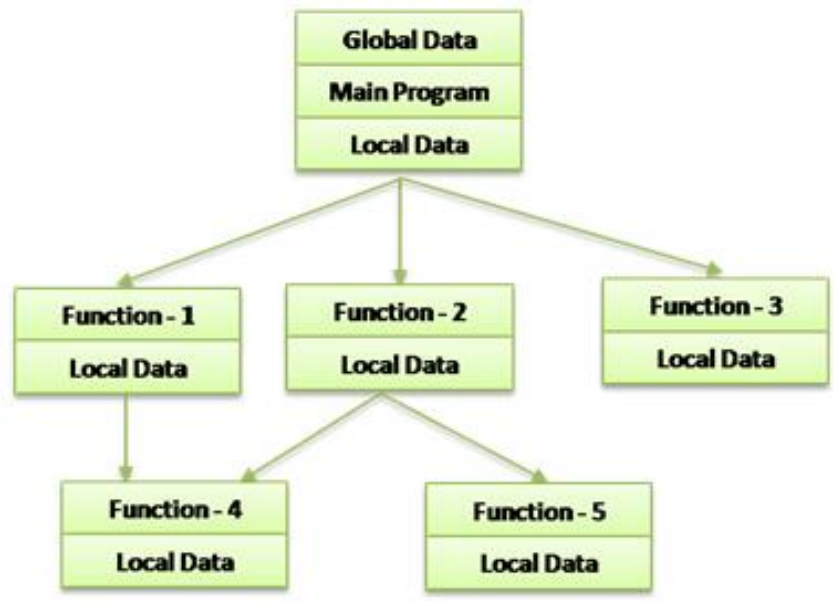


FIGURE 108. Flowchart

Pemrograman Terstruktur

- Dalam pemrograman terstruktur yang memiliki banyak fungsi, banyak data penting yang ditempatkan sebagai data *global* (variabel global) sehingga dapat diakses oleh semua fungsi
- Setiap fungsi juga dapat memiliki data lokal (variabel lokal) yang hanya berlaku di dalam fungsi tersebut
- Variabel global lebih rentan terhadap perubahan yang dilakukan dengan ceroboh oleh fungsi –fungsi yang mengaksesnya. Apalagi dalam program berskala besar.
- Pemrograman terstruktur juga tidak dapat memodelkan permasalahan dunia nyata dengan baik, karena prosedur/fungsi adalah berorientasi *action*, dan tidak berhubungan dengan elemen permasalahan

Pemrograman Terstruktur

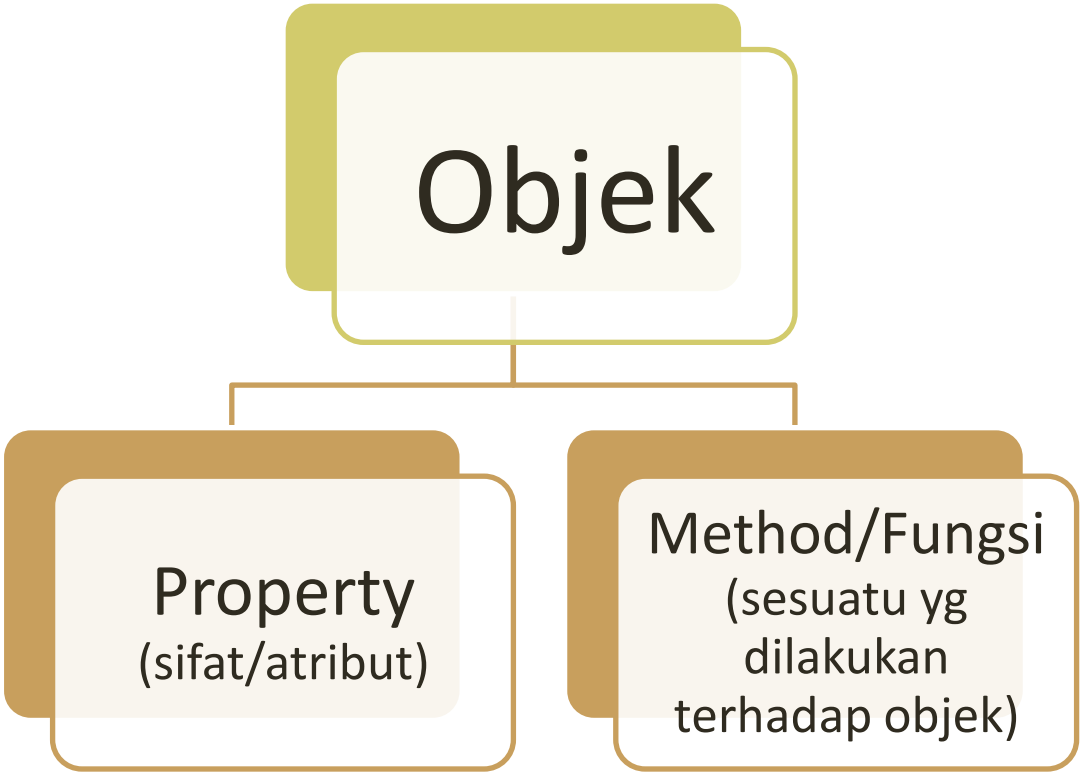


- Karakteristik Pemrograman Terstruktur:
  - Penekanan pada urutan langkah
  - Program besar dibagi menjadi beberapa program kecil yg disebut fungsi
  - Sebagian besar fungsi berbagi data global
  - Data bergerak bebas dari satu fungsi ke fungsi lainnya
  - Fungsi mentransformasi data dari satu bentuk ke bentuk lain
  - Menerapkan pendekatan *top-down* dalam desain program

Sumber gambar: http://www.ustudy.in/sites/default/files/Procedure.png

Pemrograman Berorientasi Objek

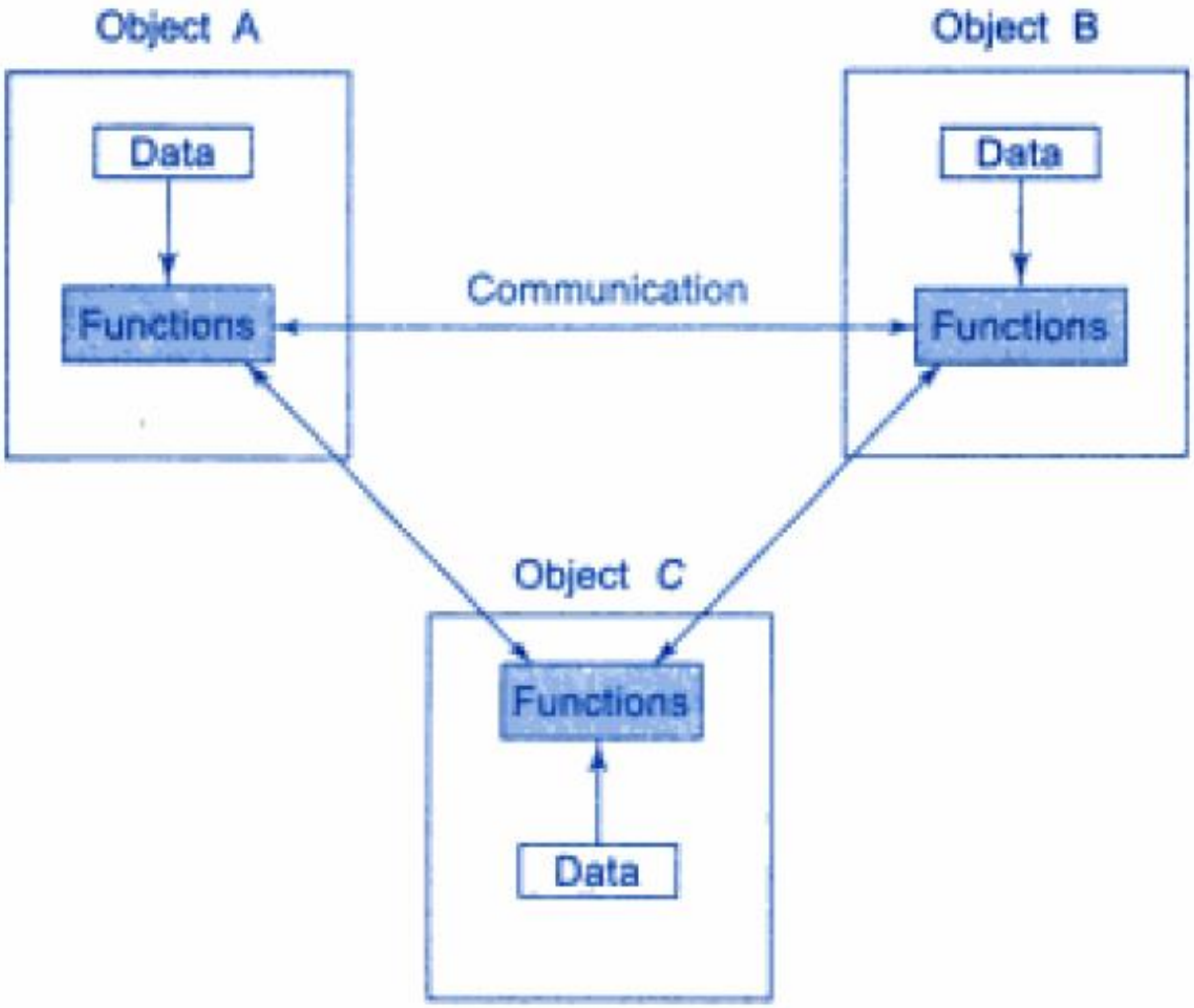
- Sebuah paradigma (cara pandang) pemrograman berdasarkan pada konsep “OBJEK” yang merupakan struktur data dalam bentuk “PROPERTY” dan “METHOD”



Pemrograman Berorientasi Objek

- Memperlakukan data sebagai elemen penting dalam program dan tidak memperbolehkannya bergerak dengan bebas di dalam sistem
- Data diikat di dalam fungsi yang beroperasi terhadap data tersebut dan melindunginya dari fungsi luar/lain yang tidak berkepentingan
- Data sebuah objek hanya dapat diakses melalui fungsi yang berasosiasi dengan objek tersebut, tetapi fungsi dari sebuah objek dapat mengakses fungsi pada objek yang lain

Data dan Fungsi dalam PBO



Pemrograman Berorientasi Objek

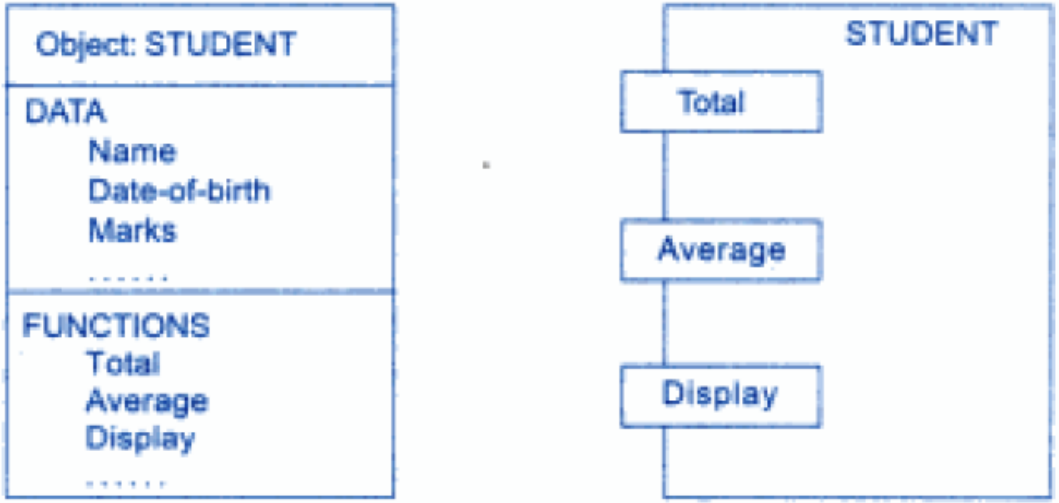
- Fitur utama dari PBO:
  - Penekanan pada data, bukan prosedur
  - Program dibagi dalam sejumlah objek
  - Struktur data dirancang agar memberikan ciri pada objek
  - Fungsi yang beroperasi pada data sebuah objek “diikat bersama” dalam struktur data
  - Data bersifat tersembunyi dan tidak dapat diakses oleh fungsi eksternal
  - Objek-objek dapat saling berkomunikasi satu sama lain dengan menggunakan fungsi
  - Data dan fungsi baru dapat ditambahkan dengan mudah
  - Menggunakan pendekatan *bottom-up* dalam desain software

Konsep Dasar PBO

- Objek
- Class
- Abstraksi data dan enkapsulasi
- Pewarisan (*Inheritance*)
- *Polymorphism*
- *Dynamic binding*
- *Message passing*

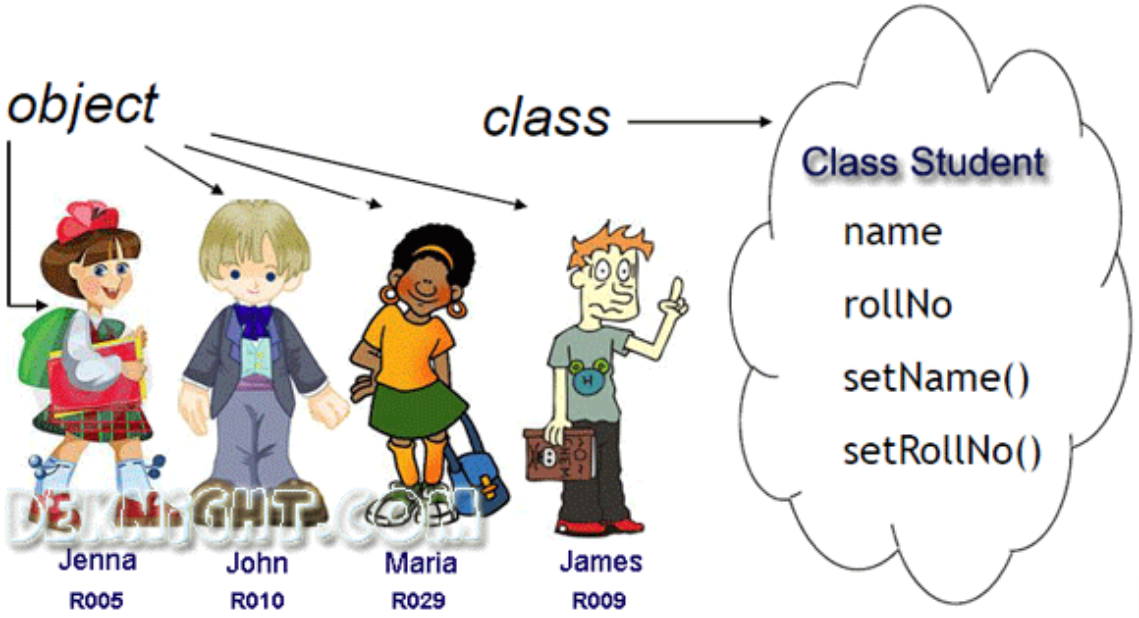
Konsep Dasar PBO: Objek

- Entitas paling mendasar, dapat mewakili orang, tempat, rekening bank, tabel data, vector, list, dan waktu
- Objek dalam program dibuat sedemikian rupa sehingga menyerupai objek dunia nyata
- Objek saling berinteraksi dengan mengirimkan pesan.
  - Contoh: Jika terdapat objek “customer” dan objek “rekening”, maka objek customer dapat mengirim pesan ke objek rekening untuk *request* saldo rekening
- Objek dapat saling berinteraksi tanpa perlu mengetahui detil data/program dari masing-masing objek



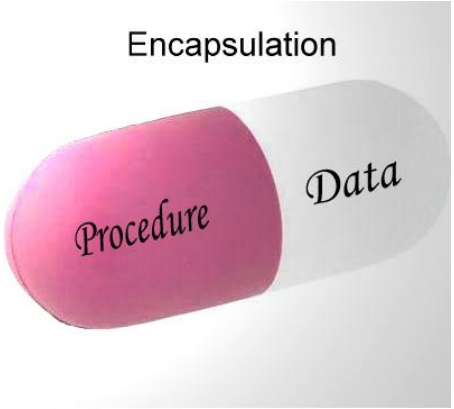
Konsep Dasar PBO: Class

- Adalah tipe data dari objek
  - Dapat dikatakan bahwa objek adalah variabel dari tipe class
- Ketika sebuah class dibuat, kita dapat membuat objek dari class tersebut dalam jumlah berapapun
- Class disebut juga sebagai sekumpulan objek yang memiliki sifat serupa



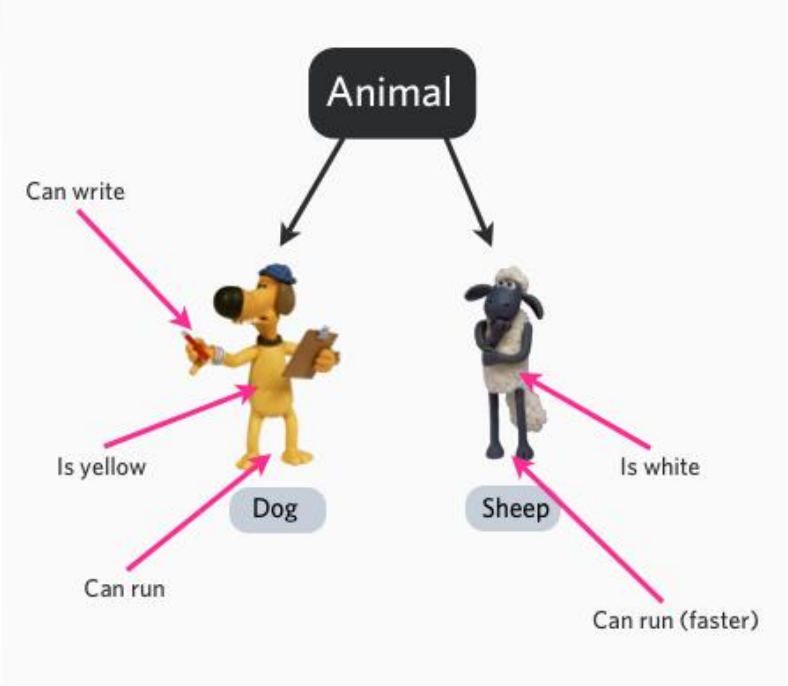
Konsep Dasar PBO: Abstraksi Data dan Enkapsulasi

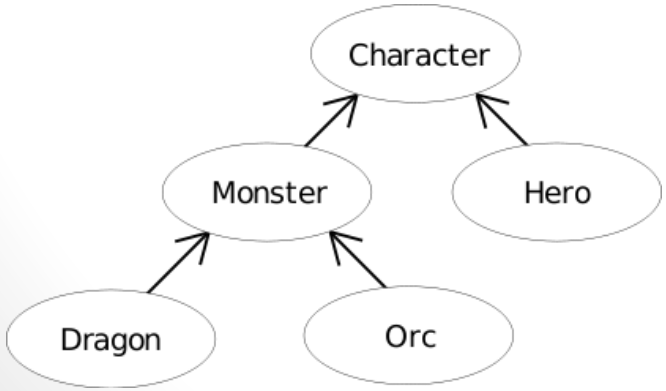
- Enkapsulasi:
  - proses pembungkusan data dan fungsi menjadi satu membentuk sebuah unit (class)
  - Disebut juga mekanisme penyembunyian data (*data hiding*)
- Abstraksi data:
  - Data menjadi abstrak, tidak terekspos detilnya
  - Untuk mengakses data, harus melalui fungsi yang disediakan
  - Tipe data yang menggunakan abstraksi data disebut **Abstract Data Types (ADT)**



ADT = Class

Konsep Dasar PBO: Pewarisan (*Inheritance*)





- Proses dimana sebuah objek dari sebuah class memiliki sifat/properti dari sebuah objek di kelas lain
- Menerapkan ide *reusability* sehingga fitur baru dapat ditambahkan tanpa harus memodifikasi class dengan cara membuat *turunan* dari class

Konsep Dasar PBO:

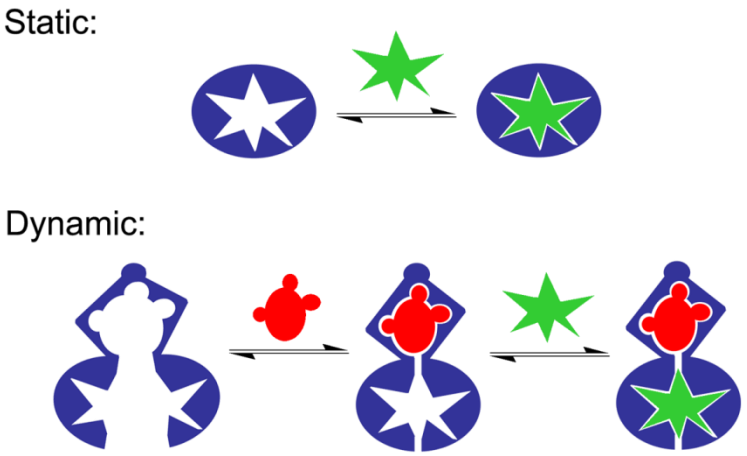
Polymorphism

- Polymorphism (Yunani):
  - Poly = banyak
  - Morph = bentuk
  - Memiliki banyak bentuk
- Sebuah perintah/operasi dapat menghasilkan perilaku yang berbeda pada objek yang berbeda, bergantung pada tipe data yang digunakan di dalam perintah
- Contoh: *operasi penambahan 2 parameter*
  - Jika parameter bertipe angka, maka hasilnya jumlah dari angka
  - Jika parameter bertipe string, maka hasilnya gabungan dari kedua string
- Proses yang memungkinkan operator bertindak berbeda jika diberikan parameter yang berbeda disebut **operator overloading**

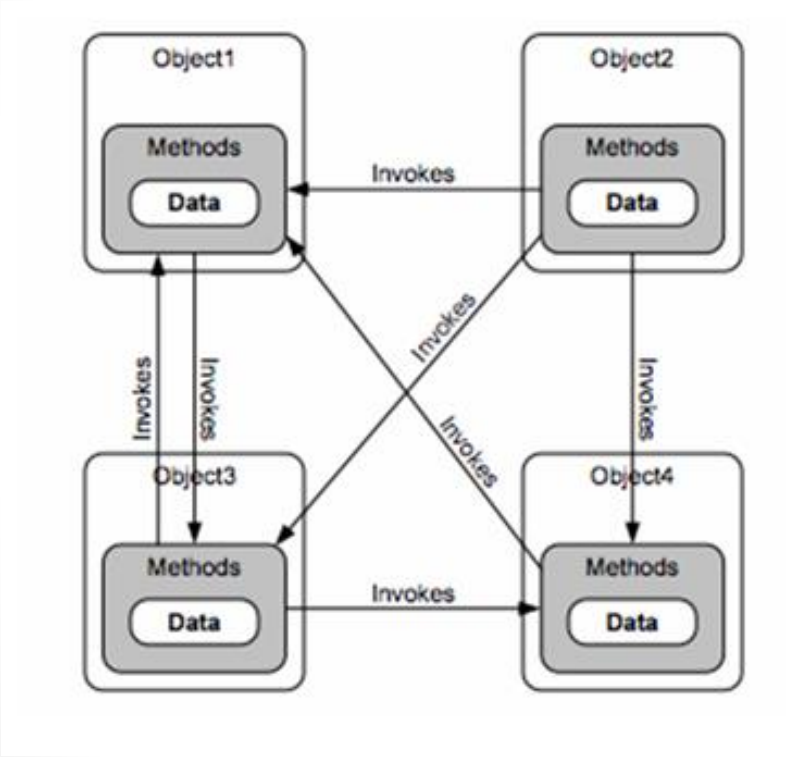
Polymorphism banyak digunakan saat mengimplementasikan pewarisan

Konsep Dasar PBO:
Ikatan Dinamis *(Dynamic Binding)*

- ***Binding*** berarti menghubungkan *(linking)* sebuah pemanggilan prosedur dengan kode program yang akan dieksekusi sebagai respon dari pemanggilan tersebut
- ***Dynamic binding*** berarti kode program yang berkaitan dengan pemanggilan prosedur belum diketahui sampai waktu pemanggilan prosedur tersebut saat *run-time*
- Konsep ini berkaitan dengan *polymorphism* dan pewarisan



Konsep Dasar PBO:
Pertukaran Pesan *(Message Passing)*



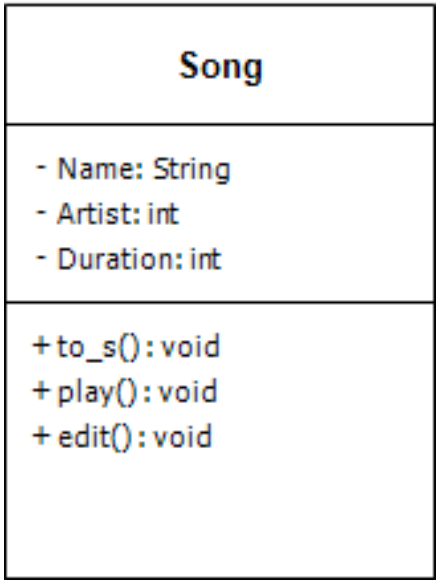
- Sebuah program yang berorientasi objek terdiri atas sejumlah objek yang berkomunikasi satu sama lain.
- Proses PBO meliputi:
  - Pembuatan class yang mendefinisikan objek dan perilakunya
  - Pembuatan objek dari definisi class
  - Membuka komunikasi antar-objek
- Objek berkomunikasi satu sama lain dengan mengirim dan menerima informasi, seperti layaknya manusia berkomunikasi
- Pertukaran pesan meliputi: **nama objek, nama fungsi (pesan),** dan **informasi yang akan dikirim**
- Tiap objek memiliki siklus hidup masing-masing (dapat dibuat dan dapat pula dihapus). Komunikasi dengan objek dapat dilakukan selama objek tersebut masih hidup

CLASS DAN OBJEK

Class

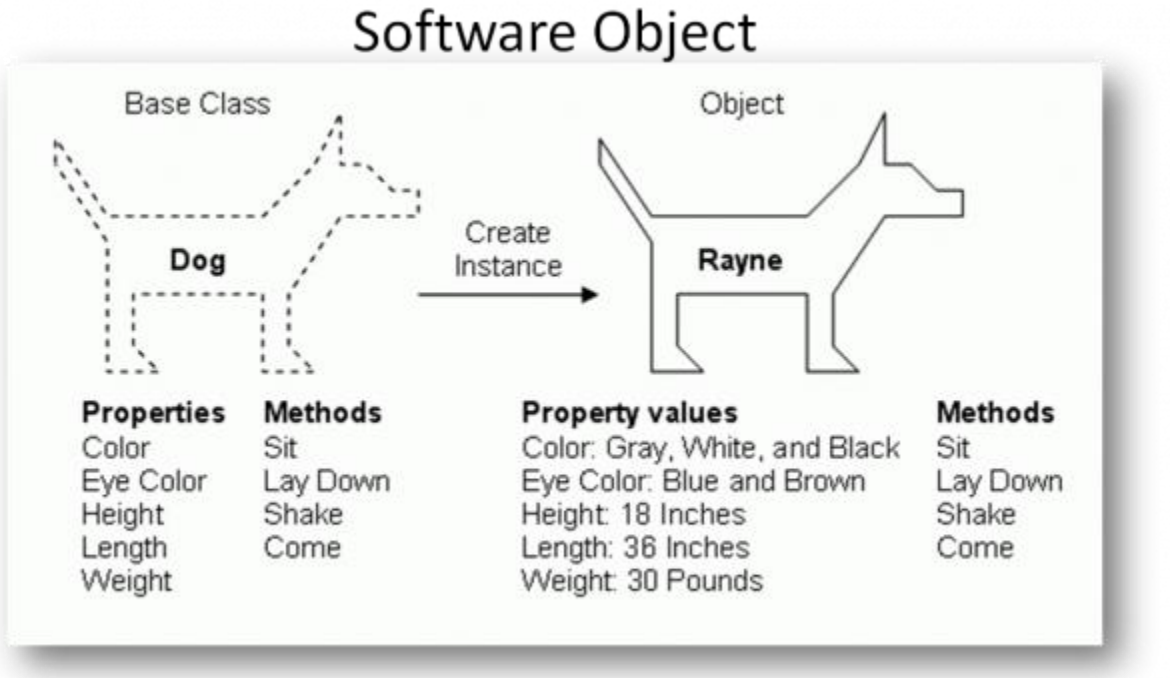
- Merupakan cara untuk mengikat data beserta fungsi-fungsi yang berkaitan dengan data
- Bila diperlukan, data dan fungsi dapat tersembunyi dari pihak eksternal class
- Ketika sebuah class dibuat, kita membuat sebuah Tipe Data Abstrak yang biasanya terdiri atas:
  - Deklarasi class
  - Deklarasi atribut pada class
  - Definisi fungsi/method pada class
- Bentuk umum dari class:

```
class class_name
   deklarasi attribute
   deklarasi fungsi
end
```



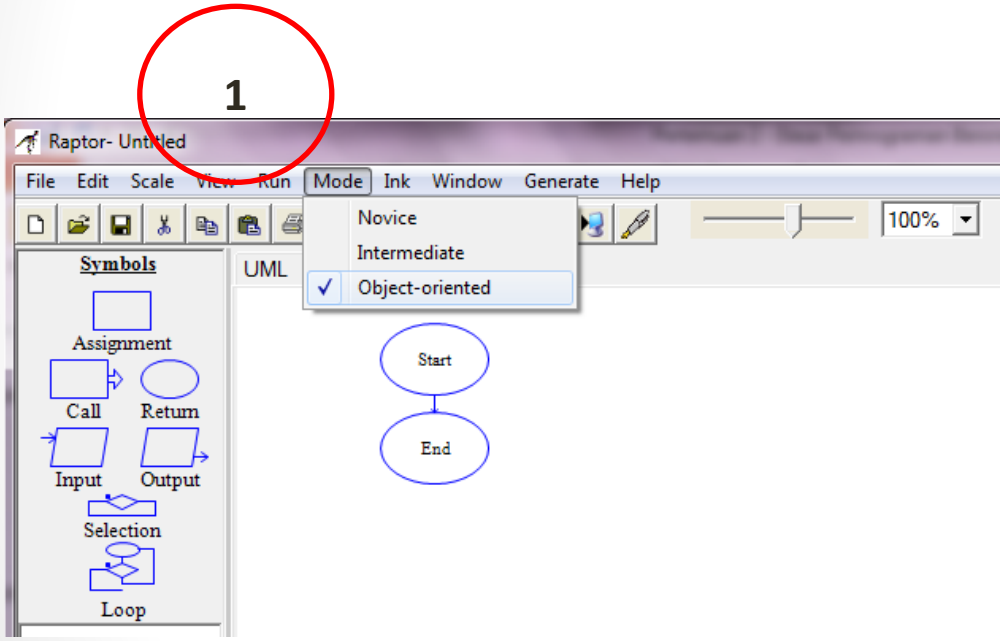
Objek

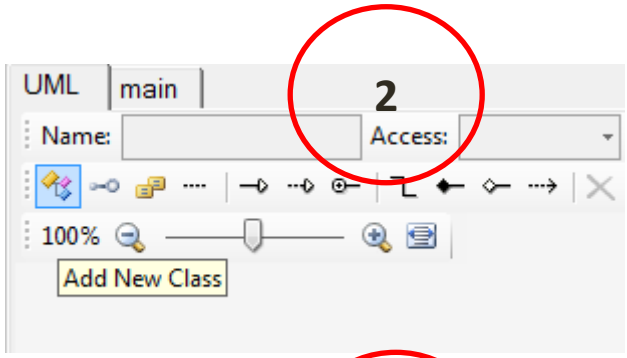
- Objek adalah *instance* dari sebuah class. Dapat pula dikatakan objek adalah variabel dari class
  - Misal terdapat sebuah class KENDARAAN, maka objeknya dapat berupa RODA_DUA, RODA_TIGA, RODA_EMPAT

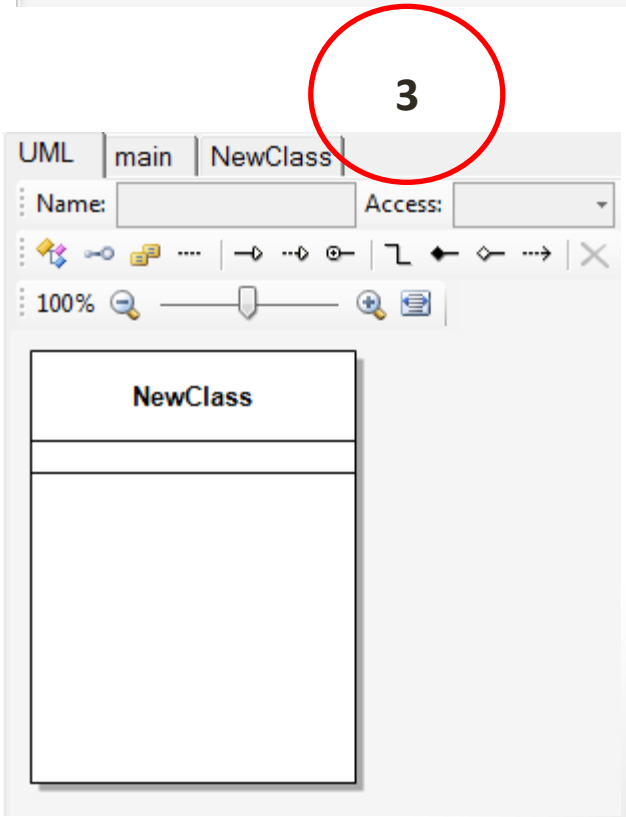


Desain Class dalam Raptor

- Pilih Mode **Object-Oriented**
- Masuk ke Tab **UML**

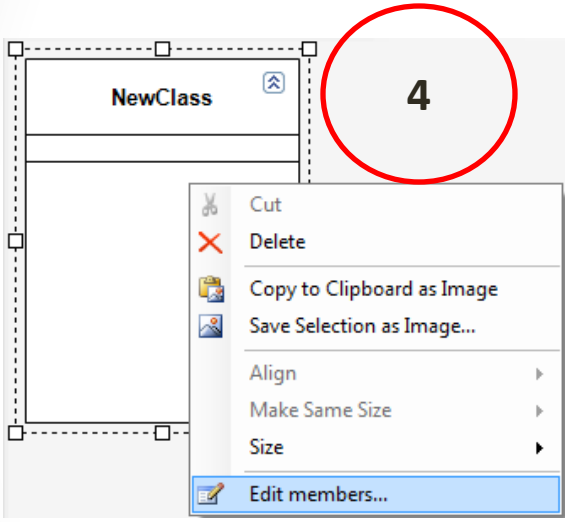


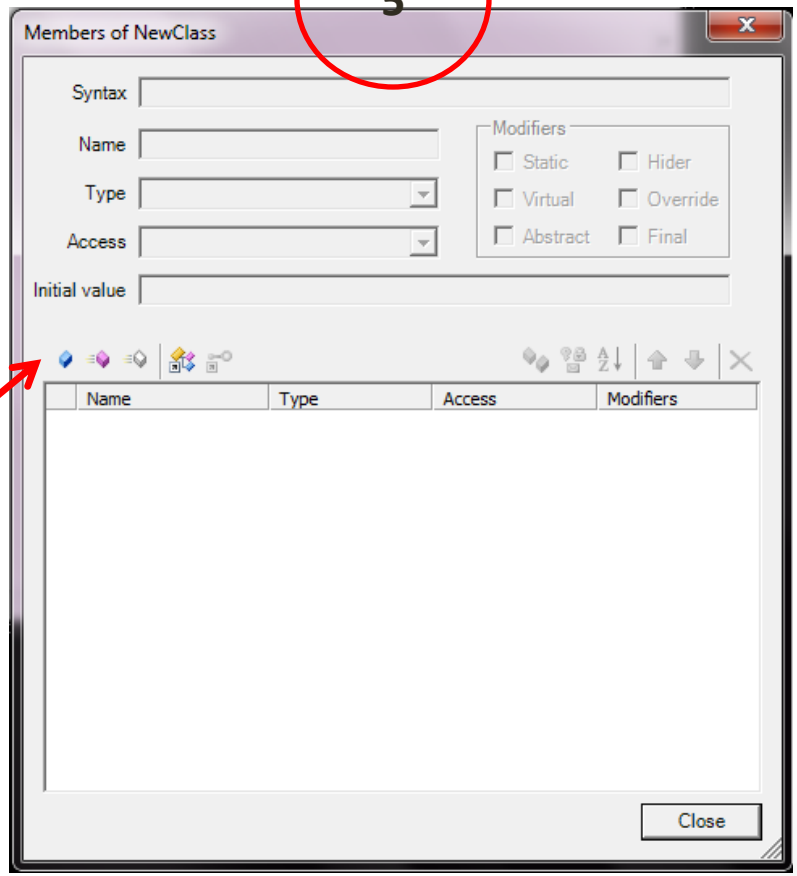




Desain Class dalam Raptor

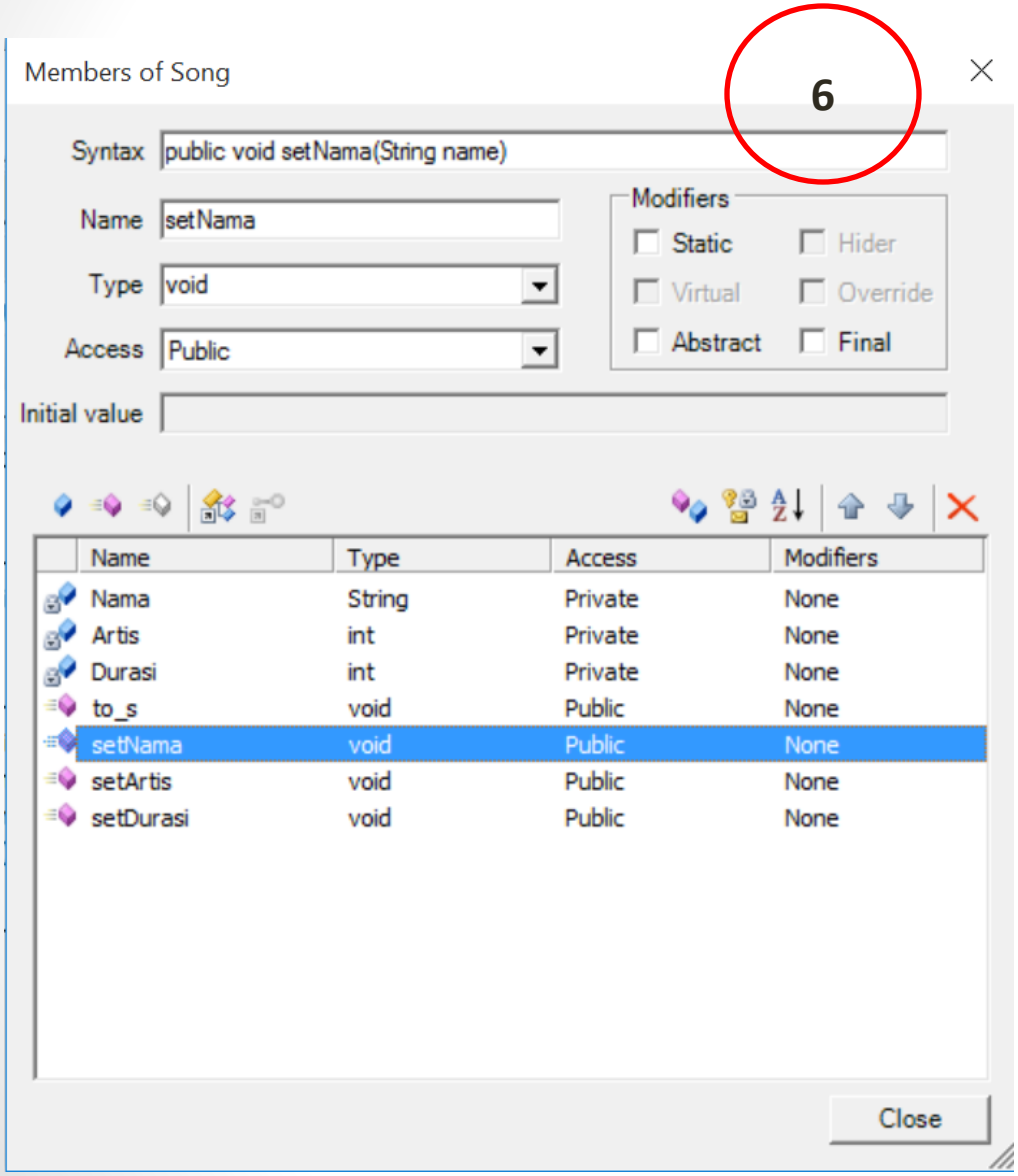
- Klik kanan pada class, pilih **Edit members…**





- Klik kotak biru untuk menambahkan atribut
- Klik kotak merah muda untuk menambahkan method
- Klik kotak putih untuk membuat Constructor

Desain Class dalam Raptor



7

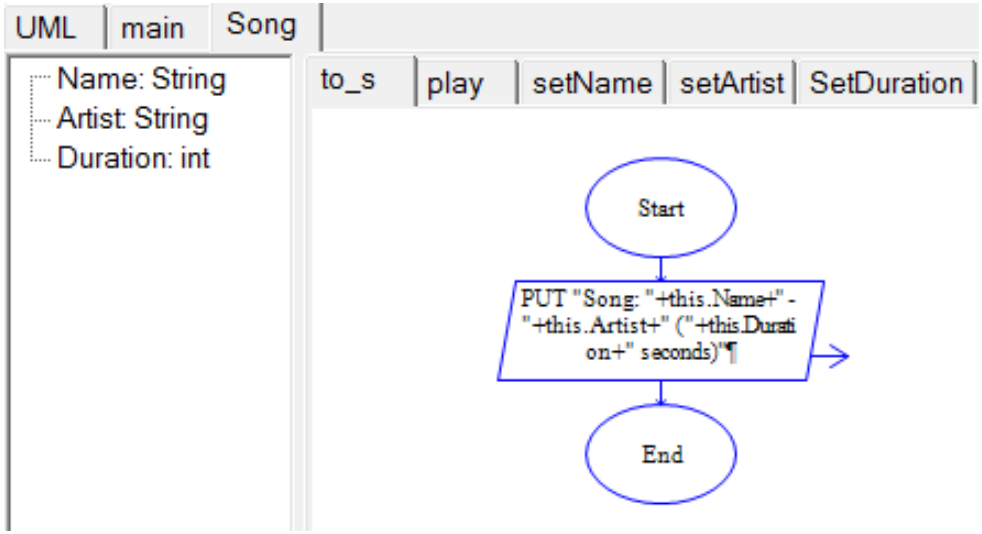
Song

- Name: String
- Artist: String
- Duration: int

+ to_s() : void
+ play() : void
+ setName(Name: String) : void
+ setArtist(Artist: String) : void
+ SetDuration(Duration: int) : void

Pembuatan Method pada Class Raptor

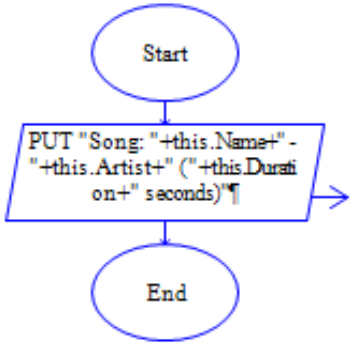
- Sekarang muncul Tab sesuai nama class beserta sub-tab method-method yang ada dalam class tersebut



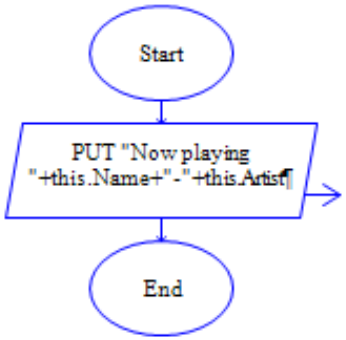
- Edit tiap tab method dengan menambahkan proses sesuai fungsinya

Pembuatan Method pada Class Raptor

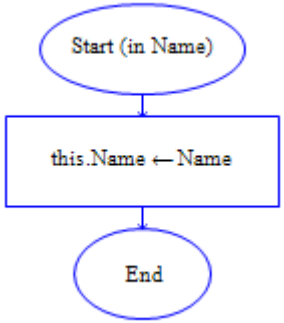
Method `to_s`



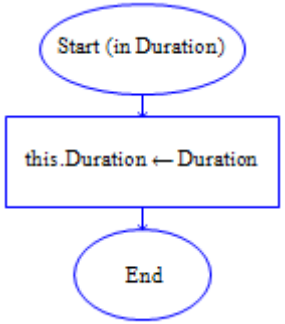
Method `play`



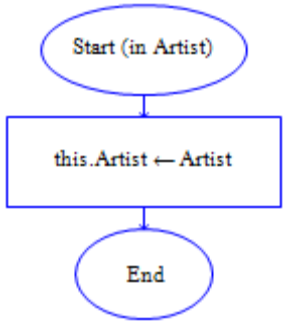
Method `setName`



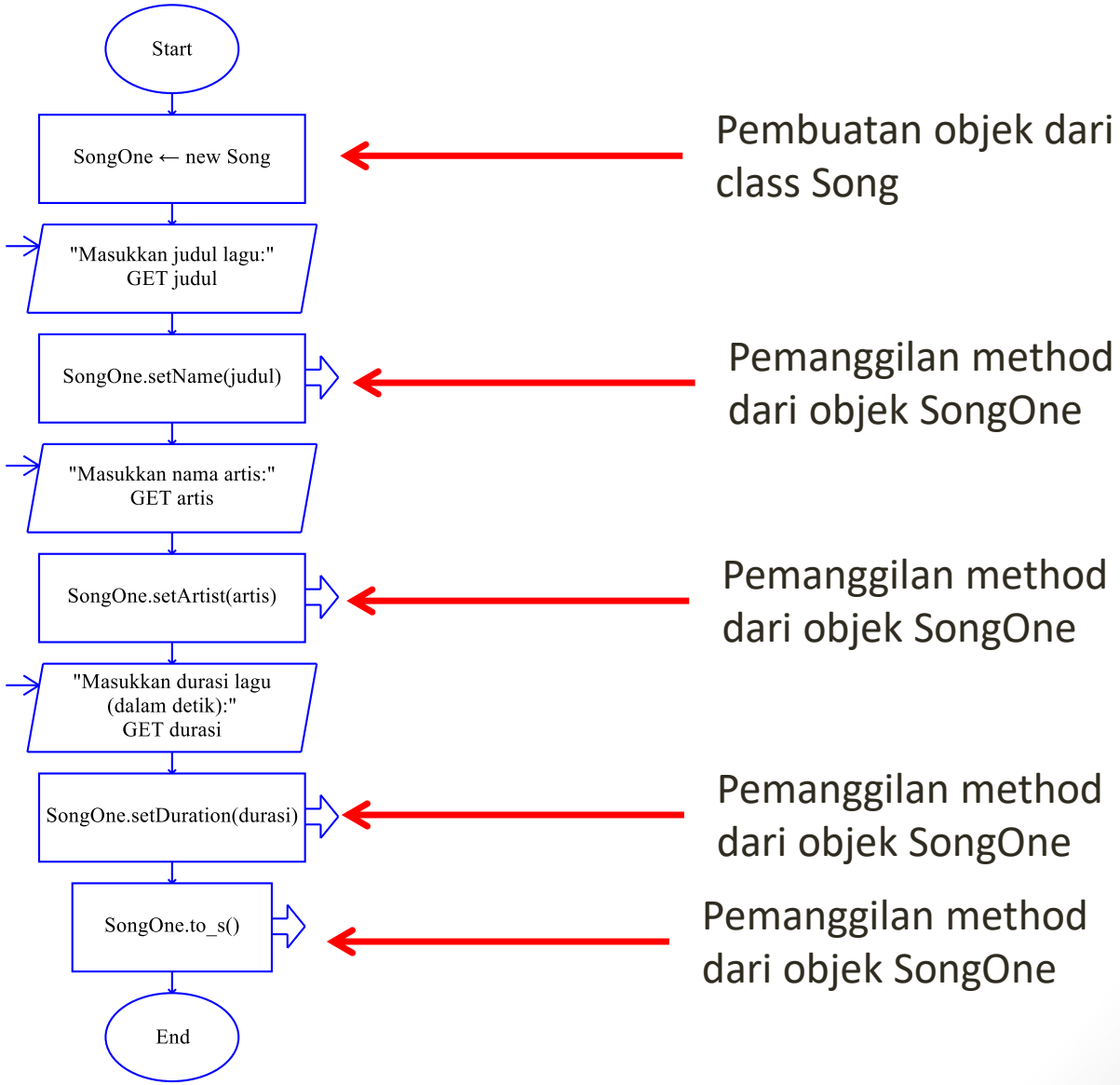
Method `SetDuration`



Method `SetArtist`



Pemanggilan Class pada Main program Raptor



Tugas 1

- Buat sebuah class bernama **`Kubus`**
  - Atribut: `sisi`, `volume`
  - Method:
    - `SetSisi(sisiBaru)`: mengisikan nilai untuk atribut `sisi`
    - `ComputeAndSetVolume()`: menghitung dan mengisikan nilai untuk atribut `volume`
    - `GetVolume()`: menampilkan nilai dari atribut `volume`
- Buat main program yang menggunakan class Kubus yang sudah Anda buat.

Tugas 2

- Buat sebuah class bernama **`Balok`**
  - Atribut: `panjang`, `lebar`, `tinggi`, `volume`
  - Method:
    - `SetPanjang(panjang)`: mengisikan nilai untuk atribut `panjang`
    - `SetLebar(lebar)`: mengisikan nilai untuk atribut `lebar`
    - `SetTinggi(tinggi)`: mengisikan nilai untuk atribut `tinggi`
    - `ComputeAndSetVolume()`: menghitung dan mengisikan nilai untuk atribut `volume`
    - `GetVolume()`: menampilkan nilai dari atribut `volume`
- Buat main program yang menggunakan class Balok yang sudah Anda buat.

Tugas 3

- Buat sebuah class bernama **`Tabung`**
  - Atribut: `radius`, `tinggi`, `volume`
  - Method:
    - `SetRadius(radius)`: mengisikan nilai untuk atribut `radius`
    - `SetTinggi(tinggi)`: mengisikan nilai untuk atribut `tinggi`
    - `ComputeAndSetVolume()`: menghitung dan mengisikan nilai untuk atribut `volume`
    - `GetVolume()`: menampilkan nilai dari atribut `volume`
- Buat main program yang menggunakan class Tabung yang sudah Anda buat.

Tugas 4

- Buat sebuah class bernama **`LimasSegiEmpat`**
  - Atribut: `sisi`, `tinggi`, `volume`
  - Method:
    - `SetSisi(sisi)`: mengisikan nilai untuk atribut `sisi`
    - `SetTinggi(tinggi)`: mengisikan nilai untuk atribut `tinggi`
    - `ComputeAndSetVolume()`: menghitung dan mengisikan nilai untuk atribut `volume`
    - `GetVolume()`: menampilkan nilai dari atribut `volume`
- Buat main program yang menggunakan class LimasSegiEmpat yang sudah Anda buat.

Tugas 5

- Buat sebuah class bernama **`Bola`**
  - Atribut: `radius`, `volume`
  - Method:
    - `SetRadius(radius)`: mengisikan nilai untuk atribut `radius`
    - `ComputeAndSetVolume()`: menghitung dan mengisikan nilai untuk atribut `volume`
    - `GetVolume()`: menampilkan nilai dari atribut `volume`
- Buat main program yang menggunakan class Bola yang sudah Anda buat.