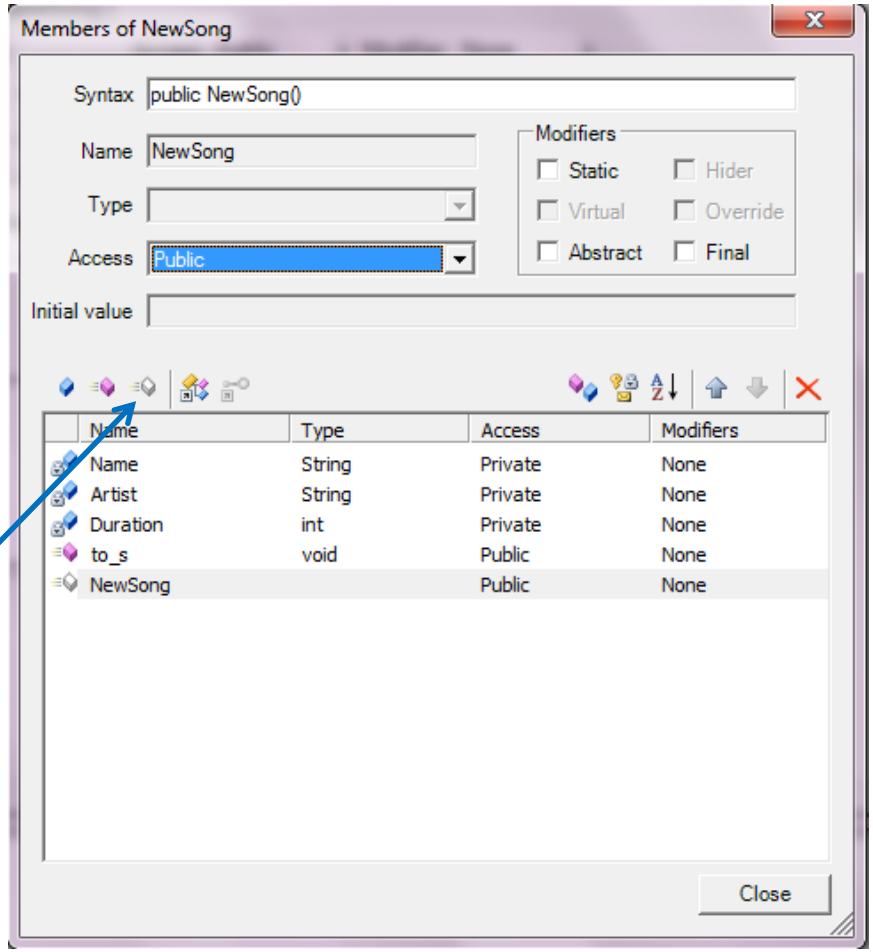
Pemrograman Berorientasi Obyek

Pertemuan 4: Konstruktor dan Destruktor

Constructor

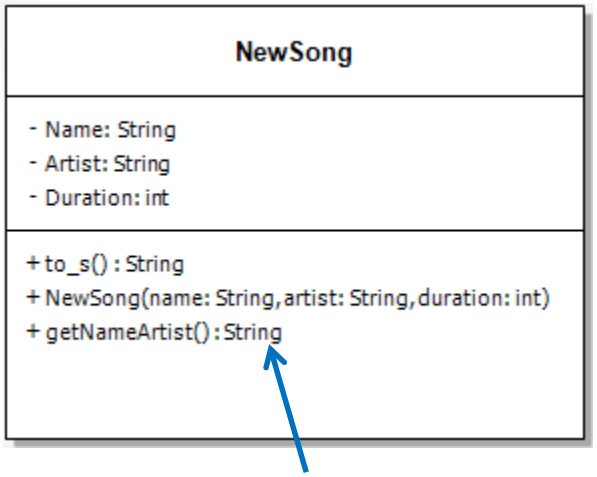
- Adalah sebuah method khusus pada sebuah class yang berfungsi untuk ***menginisialisasi*** objek pada saat pertama kali dibuat
- Method *constructor* akan dipanggil secara implisit (tanpa perlu dipanggil langsung oleh user) saat pertama kali sebuah objek dari class dibuat
- Disebut *constructor* karena method ini *mengkonstruksi/membentuk* nilai dari atribut pada class
- Constructor dapat dibuat dengan atau tanpa parameter

Pembuatan *Constructor* pada Raptor



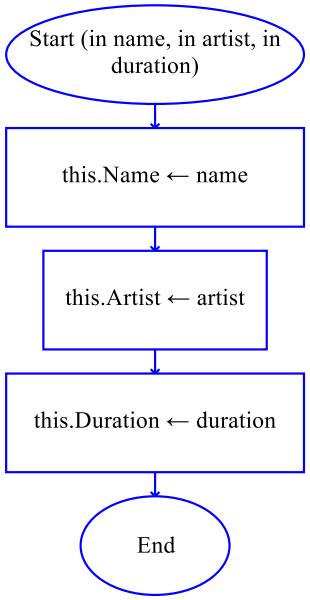
Klik tombol warna putih untuk membuat Constructor pada class. Nama method constructor otomatis sama dengan nama class.

Class NewSong

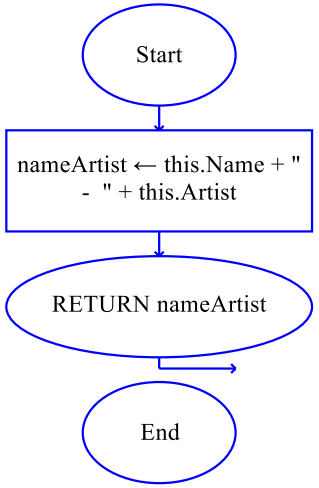


Method dengan *return value*

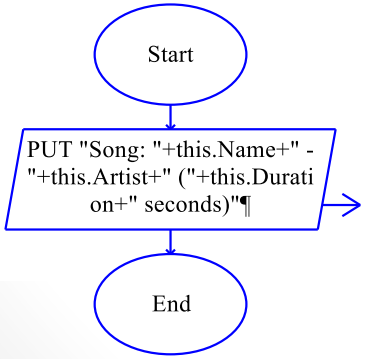
method constructor `NewSong`



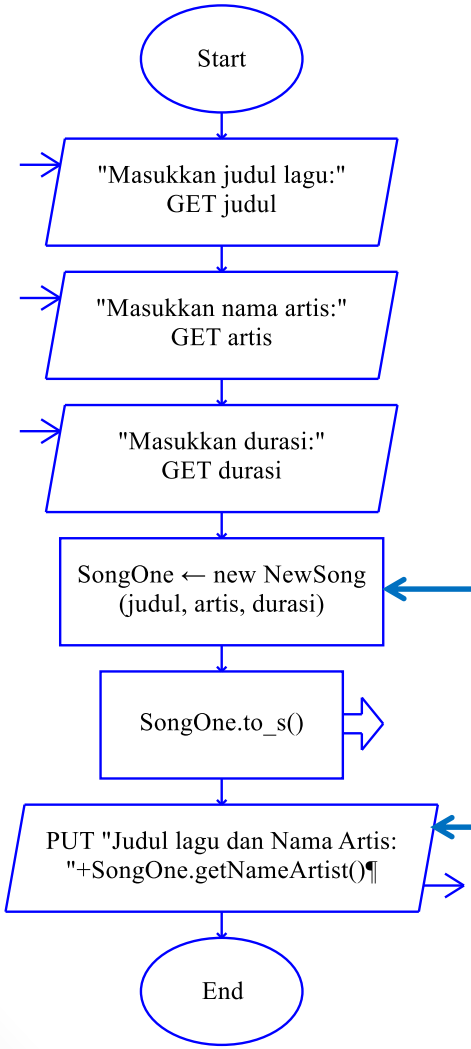
method `getNameArtist`



method `to_s`



Main Program



Pembuatan objek baru dan menggunakan *constructor*

Pemanggilan method yang mengembalikan *return value* dan menampilkannya ke layar

Constructor pada Java

- Method Constructor pada Java memiliki nama yang sama dengan nama classnya, tanpa tipe
- Ketika kita menulis perintah `new NewSong()` untuk membuat objek baru dari kelas `NewSong`, Java mengalokasikan memori untuk menampung objek yang belum diinisialisasi dan memanggil method constructor dari objek tersebut, dengan memberikan parameter-parameter kepada class `new`
- Mekanisme ini memberikan kita kesempatan untuk menyiapkan state awal dari objek baru tersebut

Contoh *Constructor* pada Java

```
public class NewSong {
    String name;
    String artist;
    int duration;
    public NewSong(String n,String a,int d)
        name = n;
        artist = a;
        duration = d;
    }
}
```

Catatan:
Modifier dari atribut di dalam class Java adalah PRIVATE BY DEFAULT, artinya jika tidak memiliki modifier, maka atribut akan diperlakukan sebagai atribut private

Class NewSong

```
1  package samples;
2
3  public class NewSong {
4      String name;
5      String artist;
6      int duration;
7      public NewSong(String n, String a, int d){
8          this.name=n;
9          this.artist=a;
10         this.duration=d;
11     }
12     public void to_s(){
13         System.out.println("Song: "+this.name+"--"+this.artist+"("+this.duration+" seconds)");
14     }
15     public String getNameArtist(){
16         String output=this.name+" - "+this.artist;
17         return output;
18     }
19     public static void main (String[] args){
20         NewSong ns = new NewSong("Hello","Lionel Richie",300);
21         ns.to_s();
22         System.out.println(ns.getNameArtist());
23     }
24 }
```

Output:

```
Song: Hello--Lionel Richie(300 seconds)
Hello - Lionel Richie
```

Latihan

- Modifikasi class yang telah Anda buat pada tugas sebelumnya sehingga memiliki fitur :
  - Constructor (Anda dapat menghilangkan method *setter* dan menggantinya langsung dengan parameter dalam *constructor*)
  - Method dengan *return value* (Anda dapat memodifikasi method `GetVolume()` untuk mengembalikan nilai volume)
- Modifikasi *main program* yang menunjukkan penggunaan fitur Constructor dan Method dengan *return value* yang Anda buat tersebut

Multiple Constructor

- Beberapa bahasa pemrograman berorientasi objek memperbolehkan adanya konstruktor lebih dari 1 (*multiple constructor*) pada sebuah class
- *Multiple constructors* berarti membolehkan untuk menginisialisasi class dengan beberapa cara/method

Multiple Constructor

Pada Java

```
public class MyClass {

    private int number = 0;

    public MyClass() {
    }

    public MyClass(int theNumber) {
        this.number = theNumber;
    }
}
```

Pada C++

```
1  // overloading class constructors
2  #include <iostream>
3  using namespace std;
4
5  class Rectangle {
6      int width, height;
7    public:
8      Rectangle ();
9      Rectangle (int,int);
10     int area (void) {return (width*height);}
11 };
12
13 Rectangle::Rectangle () {
14   width = 5;
15   height = 5;
16 }
17
18 Rectangle::Rectangle (int a, int b) {
19   width = a;
20   height = b;
21 }
22
23 int main () {
24   Rectangle rect (3,4);
25   Rectangle rectb;
26   cout << "rect area: " << rect.area() << endl;
27   cout << "rectb area: " << rectb.area() << endl;
28   return 0;
29 }
```

Multiple Constructor pada Ruby

- Ruby tidak memiliki mekanisme untuk melakukan *multiple constructor* secara langsung seperti Java dan C++
- Sebagai salah satu alternatifnya, kita dapat menggunakan variasi jumlah pada parameter method `initialize`

```ruby
class MultiConstructor
    def initialize(*args)
        if (args.size==1)
            @parameter1=args[0]
        elsif (args.size==2)
            @parameter1=args[0]
            @parameter2=args[1]
        elsif (args.size==3)
            @parameter1=args[0]
            @parameter2=args[1]
            @parameter3=args[2]
        end
    end
    def printParameters
        puts "Parameter 1 = #@parameter1"
        puts "Parameter 2 = #@parameter2"
        puts "Parameter 3 = #@parameter3"
    end
end

mc=MultiConstructor.new(1)
mc.printParameters()
puts "------------------------"
mc=MultiConstructor.new(1,2)
mc.printParameters()
puts "------------------------"
mc=MultiConstructor.new(1,2,"Hello World")
mc.printParameters()
```

Output

```
Parameter 1 = 1
Parameter 2 =
Parameter 3 =
------------------------
Parameter 1 = 1
Parameter 2 = 2
Parameter 3 =
------------------------
Parameter 1 = 1
Parameter 2 = 2
Parameter 3 = Hello World
```

Pembuatan objek baru dengan memberikan jumlah parameter yang berbeda

Destructor pada bahasa C++

- Bahasa pemrograman C++ memiliki method *destructor* (pasangan dari *constructor*)
- Sesuai namanya, *destructor* berfungsi untuk menghapus objek yang telah dibuat oleh *constructor*
- Jika *constructor* dipanggil secara implisit ketika objek akan dibuat, maka *destructor* akan dipanggil (secara implisit juga) ketika eksekusi program akan berakhir untuk membersihkan memori yang sudah tidak terpakai lagi
- Sama seperti *constructor*, *destructor* memiliki nama yang sama dengan nama class, hanya ditambah awalan tanda “tilde” (~)

Destructor pada bahasa C++

```
1  class String
2  {
3      public:
4      String() //constructor with no arguments
5          :str(NULL),
6          size(0)
7      {
8      }
9
10     String(int size) //constructor with one argument
11         :str(NULL),
12         size(size)
13     {
14         str = new char[size];
15     }
16
17     ~String() //destructor
18     {
19         delete [] str;
20     };
21
22     private:
23     char *str;
24     int size;
25 }
```

"Destructor" pada Java, C#, dan Ruby

- Java, C#, dan Ruby (serta sebagian besar bahasa pemrograman *script* lainnya) tidak memiliki method *destructor*
- Mereka memiliki fitur ***garbage collector* (GC)** yang bertugas untuk membersihkan memori dari bagian program yang sudah tidak terpakai lagi
- Kelebihan GC: membersihkan memori secara otomatis (tanpa campur tangan *programmer*)
- Kekurangan GC: Menambah total waktu eksekusi dari sebuah program, sehingga dapat berpengaruh terhadap kinerja dari program

Next Week

- Inheritance (pewarisan)