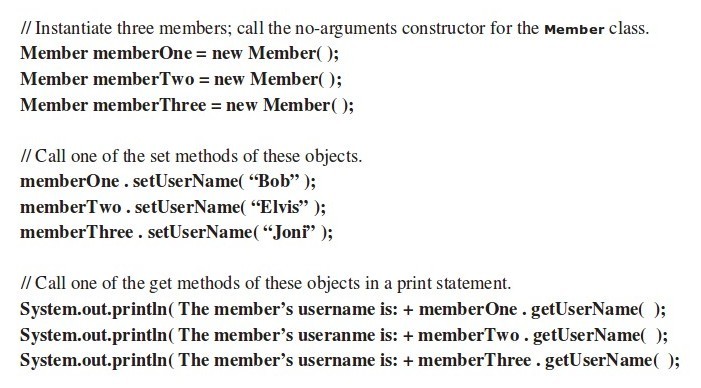
Chapter 4 : Method : Memicu Perilaku Object

4.1 Bagaimana kita mendapat nilai data dalam suatu *method ?*

Suatu *method* dapat dipanggil dengan memilih referensi objek yang di butuhkan. Sintaks Umum dari pemanggilan *method* dapat diringkas sebagai berikut :

Object\_reference.MethodName(param\_1, param\_2, …. , param\_X);

Merujuk kembali, untuk member dari suatu *themed aplication*, kira harus menginstansiasi beberapa *member* objek (di dalam ***main*** *method*) dan memanggil *method* mereka, seperti pada potongan kode berikut :



Singkatnya, kita harus memastikan bahwa kita tahu *method* mana yang di panggil pada objek mana dan dimana urutannya.

Mendeklaraikan definisi *Method* harus bersamaan dengan badan *method* (*body* *method*), seperti pada contoh berikut :

modifier return\_type method\_name(parameter\_list) exception\_list{

// body of method

// return statement (if any) is last

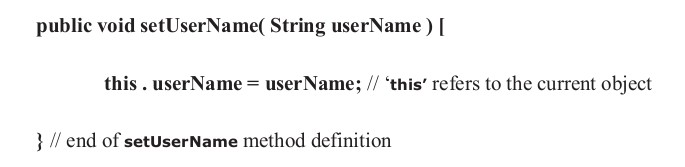
}

*method signature adalah nama dan daftar parameternya.*

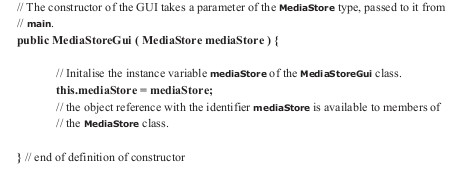
Isi dari *body’s method* antara lain :

* Invocations : memanggil *method* lain
* Assignments : mengganti variable dalam *fields* atau menjadi variable lokal
* Selection : dikarenakan percabangan
* Repetition : dikarenakan iterasi
* detect exceptions seperti kondisi error

Intifier dari parameter sama dengan contoh variable. Kompiler dapat membedakan perbedaan diantara keduanya dengan menggunakan keyword *this,* seperti contohnya :

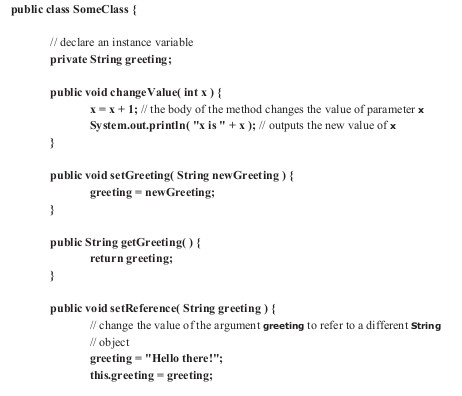


Point terakhir untuk membuat argument yang berkaitan adalah bahwa *method*  tidak dapat disampaikan sebagai argumen ke *method* lain atau konstruktor. Sebagai gantinya referensi objek dapat di sampaikan ke *method* atau konstruktor jadi, referensi objek dapat digunakan untuk *method* konstruktor atau anggota *class* yang lain yang memanggil *method* yang bersangkutan. Sebagai contoh :

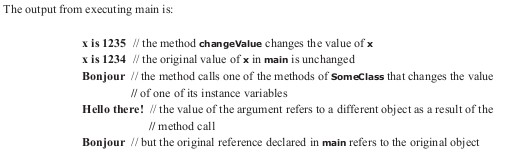


4.2 melewati Nilai Melawan Melewati Referensi

Semua argument ke *method* (dan konstruktor) di *java,* melewati nilai. Yang artinya salinan dari argumen masuk ke suatu method (atau konstruktor) pemanggil. Sebagai contoh untuk yang melewati nilai :





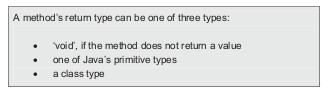


*method* changeValue mengganti nilai dari argumen lalu memindahkgannya (a salinan x ) tetapi tidak mengganti nilai asli dari x, seperti yang di tunjukan hasilnya.

Ketika sebuah parameter referensi objek, ada salunan dari referensi objek yang di berikan kepada *method.* Kita dapat mengganti objek mana yang mengacu kedalam *method*, tanpa memengaruhi referensi objek asli yang sudah di berikan. Akan tetapi jika di badan *method* ada yang memanggil *method* dari objek asli lewati salinan referensi, itu akan mengganti keadaan objek, keadaan onjek terganti selama ada di *scope* program.

**

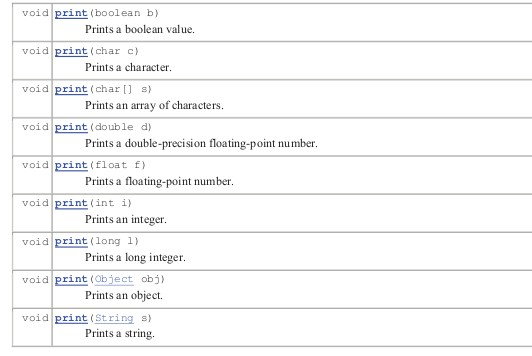
4.2 Bagaimana kita mendapat nilai data keluar dari *method ?*

**

Statemen ke dua dan tiga diatas, harus sama dengan tipe pengembalian yang ditentukan pada *mehod* deklarasi dan harus mengembalikan nilai dari tipe yang sama dengan tipe pengembalian. Tidak ada tipe pengembalian ketika menggunakan *method* “voyaid”. Kesalah yang terjadi pada kompiler biasanaya adalah menghilankan tipe pengembalian, atau memasukannya kedalam *method void,* atau secara keliru menempatkan nilai yang mau dikembalikan di akhir deklarasi dan tipe yang harus di deklarasikan di deklarasi *mehod.*

4.3 *Method Overloading*

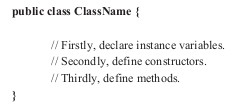
*Method* dengan nama yang sama tetaoi berbeda daftar parameternya dinamakan *Overloaded.* Diantara *methods Prinstream class* terdapat banyak bentuk *print method,* contohnya :

**

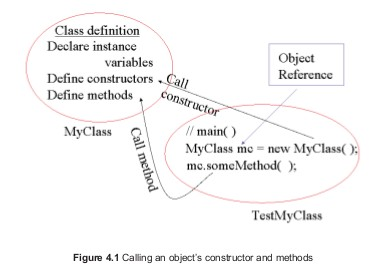
Daftar *print method* diatas berarti, *method Overloading* secara khusus digunakan ketika macam macam *method* menerima data dari berbagai macam bentuk. Dalam hal ini konstruktor juga dapat *overloaded*.

4.4 Struktur dari Definisi Umum *Class*

Gamabaran Definisi umum class adalah sebagai berikut :

**

Dari gambar diatas dapat diartikan bahwa ada tiga katagori yaitu contoh variable, konstruktor, dan *method.* Kita dapat menggunakan struktur sederhana ini untuk meringkas poin-poin utama dengan memanggil objek dari *method,* seperti pada ilustrasi berikut :

**

Gambar 4.1 menunjukkan potongan kode kelas tes sederhana untuk kelas yang definisinya disebut **MyClass**. Jika kita berasumsi bahwa salah satu konstruktor objek dari jenis **MyClass** adalah konstruktor tanpa argumen dan bahwa diagram kelas **MyClass** menyertakan metode tanpa argumen dengan identifier **someMethod**, dua pernyataan dari potongan kode untuk *main* sebagai mana yaitu :

**