1. Access mofifiers berarti hak askses yang diberikan untuk memberikan untuk menginisialisasikan data type agar tidak dapat mudah diakses di class/package lain.

4 macam access modifiers :

* Public   
  Public berarti menginisasikan suatu tipe data variable agar bisa diakses di class,subclass,package dan world yang berbeda.
* Protected   
  Protected berarti menginisiasikan suatu tipe data variable agar bisa diakses di class, subclass dan package.
* No modifier  
  No modifier berarti tidak ada penjagaan tipe data secara pasti, dan bias diakses di class dan subclass yang berbeda.
* Private   
  Private merupakan yang tipe access modifier yang paling aman karena hanya bias diakses di class itu sendiri

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Access Modifier | Class | Subclass | Package | World |
| Private | yes |  |  |  |
| Protected | yes | yes |  |  |
| No modifier | yes | yes | yes |  |
| Public | yes | yes | yes | yes |

1. Overloading berati menggunakan nama fungsi atau method yang sama, namun menggunakan parameter yang berbeda. Biasanya overloading masuk kedalam staticbinding yang berarti akan dijalankan pada saat compile time(early binding). Overriding berarti menggunakan method yang sama namun di class yang berbeda. Overriding masuk kedalam dynamic binding berarti dijalankan pada saat runtime.

Contoh overloading :

**public** **class** Mobil {

**private** **int** roda;

**private** **int** mesin;

//overloading

**public** Mobil(**int** roda) {

**super**();

**this**.roda = roda;

}

//overloading

**public** Mobil(**int** roda, **int** mesin) {

**super**();

**this**.roda = roda;

**this**.mesin = mesin;

}

**public** Mobil() {

// **TODO** Auto-generated constructor stub

}

}

Contoh overriding :

**public** **class** Mobil {

**private** **int** roda;

**private** **int** mesin;

**public** Mobil(**int** roda) {

**super**();

**this**.roda = roda;

}

**void** berjalan(){

System.***out***.println(",mobil berjalan");

}

**public** Mobil() {

// **TODO** Auto-generated constructor stub

}

}

**public** **class** Motor **extends** Mobil {

@Override

**void** berjalan() {

System.***out***.println("Motor berjalan");

}

**public** Motor() {

// **TODO** Auto-generated constructor stub

}

}

1. 3.
2. 3 method Arraylist:

* Add, menambahkan suatu data kedalam array list yang sudah dibuat. Penambahan data otomatis akan pertama kali masuk ke index 0.
* Remove, menghapus 1 data spesifik yang berada dalam array, sesuai dengan index yang ditulis.
* Set, mengupdate 1 data spesifik(mengubah salah satu/isi)dari index yang sudah ditentukan.

**public** **class** Mobil {

**private** **int** roda;

ArrayList<Mobil> daftarMobil = **new** ArrayList<>();

**public** Mobil(**int** roda) {

**super**();

**this**.roda = roda;

}

**void** tambahMobil(){

Mobil m = **new** Mobil(4);

//add ke index 0

daftarMobil.add(m);

//mengubah index 1 dengan cara set

m.roda = 5;

daftarMobil.set(0, m);

//remove index 0

daftarMobil.remove(0);

}

}

}