



POLITEKNIK MASAMY INTERNASIONAL

SK Menristekdikti RI Nomor: 731/KPT/II/2018

Jalan Ikan Paus No.10-15 Kertosari Banyuwangi - 68411
Telp (0333) 3384593 – <http://polmain.info>

PROGRAM STUDI D3 TEKNIK KOMPUTER

Form:
B.Ak/eva/04/20

Nama Dosen : *Arif Fahmi, S.T.,M.T.*

Mata Kuliah : Pemrograman Berorientasi Object

Semester : 4 (Empat)

Kode Mata Kuliah : TKV4044

Th. Akdm : 2019/2020.

BAB

METHOD BAGIAN II

SUB BAB :

1. Method Overriding
2. Constructor
3. Destructor

TUJUAN MATERI

1. Mahasiswa memahami konsep method lanjutan pada java diantaranya, method overriding, constructor, dan destructor serta garbage collection
2. Mahasiswa mampu membuat program dengan menerapkan konsep method overriding, constructor, serta garbage collection

REFERENSI

1. J.Eck, D. (2006). *Introduction to Programming Using Java*. Geneva.
2. An Object-Oriented Approach to Programming Logic and Design, Joyce Farrel, USA, 2013
3. An Introduction to Object Oriented Programming with Java, C. Thomas Wu, McGraw-Hill, New York, 2010.

METHOD BAGIAN II

Pada materi sebelumnya kita telah membahas materi tentang inheritance (pewarisan) dan sebagian materi method. Pada bab ini kita akan melanjutkan materi tentang method diantaranya yaitu method overriding, constructor, destructor serta Method setter & getter. Berikut penjelasan dari masing-masing subbab method bagian II.

1. Method Overriding

Method Overloading adalah sebuah kemampuan yang membolehkan **sebuah class mempunyai 2 atau lebih method dengan nama yang sama, yang membedakan adalah parameternya.**

Pada method overloading perbedaan parameter mencakup :

1. Jumlah parameter
2. Tipe data dari parameter
3. Urutan dari tipe data parameter

Method overriding merupakan method yang parent class yang ditulis kembali oleh subclass.

Aturan dari method overriding pada Java :

1. Parameter yang terdapat pada method overriding di subclass harus sama dengan parameter yang terdapat pada parent class.
2. Aturan hak akses, hak akses method overriding di subclass tidak boleh lebih ketat di bandingkan dengan hak akses method pada parent class.

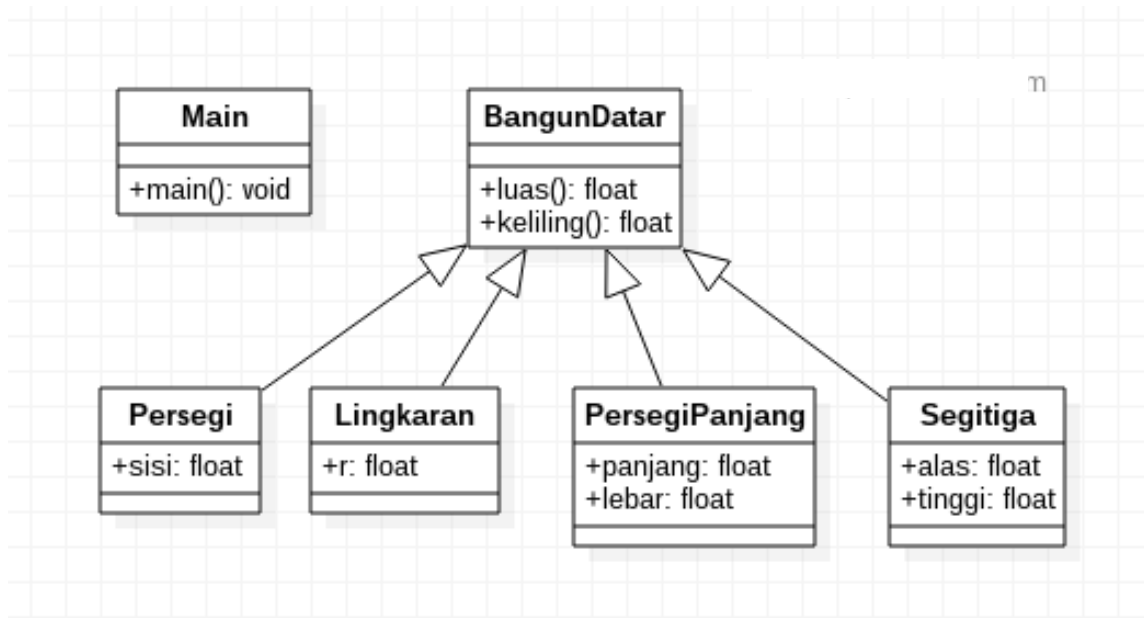
Method Overriding dapat dibuat dengan menambahkan anotasi **@Override** di atas nama method atau sebelum pembuatan method.

Contoh penerapan method overriding

Pada materi sebelumnya kita telah membuat contoh program penerapan konsep inheritance sebagaimana berikut.

Program yang telah kita buat untuk berfungsi menghitung luas dan keliling bangun datar.

Bentuk class diagramnya seperti ini:



Berikut Listing program dari masing masing class

a. BangunDatar

```
Start Page  Player.java  Game.java  BangunDatar.java  Persegi.java  Lingkaran.java
Source  History  [Icons]
1
2  package inheritance;
3
4  /**
5   *
6   * @author ARIF FAHMI
7   */
8  public class BangunDatar {
9
10     float luas() {
11         System.out.println("Menghitung laus bangun datar");
12         return 0;
13     }
14
15     float keliling() {
16         System.out.println("Menghitung keliling bangun datar");
17         return 0;
18     }
19 }
```

b. Persegi

```
Start Page  Player.java  Game.java  BangunDatar.java
Source  History  [Icons]
1
2  package inheritance;
3
4  /**
5   *
6   * @author ARIF FAHMI
7   */
8  public class Persegi extends BangunDatar{
9      float sisi;
10
11 }
```

c. Lingkaran

```
Start Page  Player.java  Game.java  BangunDatar.java
Source  History  [Icons]
1
2  package inheritance;
3
4  /**
5   *
6   * @author ARIF FAHMI
7   */
8  public class Lingkaran extends BangunDatar{
9      // jari-jari lingkaran
10     float r;
11
12 }
```

d. Persegi panjang

```
Start Page  Player.java  Game.java  BangunDatar.java  Pers
Source  History
1
2  package inheritance;
3
4  /**
5   *
6   * @author ARIF FAHMI
7   */
8  public class PersegiPanjang extends BangunDatar{
9      float panjang;
10     float lebar;
11
12 }
13
```

e. Segitiga

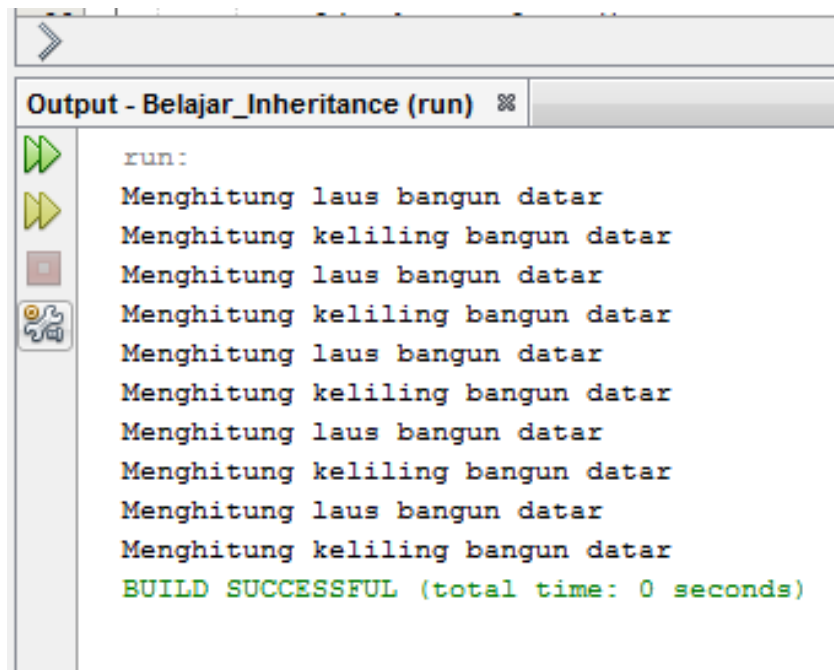
```
Start Page  Player.java  Game.java  BangunDatar.java
Source  History
1
2  package inheritance;
3
4  /**
5   *
6   * @author ARIF FAHMI
7   */
8  public class Segitiga extends BangunDatar{
9      float alas;
10     float tinggi;
11 }
12
```

f. Main

```
Start Page  Player.java  Game.java  BangunDatar.java  Persegi.java  Lingkaran.
Source  History
2  package inheritance;
3  /**
4   *
5   * @author ARIF FAHMI
6   */
7  public class Main {
8      public static void main(String[] args) {
9
10         // membuat objek bangun datar
11         BangunDatar bangunDatar = new BangunDatar();
12
13         // membuat objek persegi dan mengisi nilai properti
14         Persegi persegi = new Persegi();
15         persegi.sisi = 2;
16
17         // membuat objek Lingkaran dan mengisi nilai properti
18         Lingkaran lingkaran = new Lingkaran();
19         lingkaran.r = 22;
20
21         // membuat objek Persegi Panjang dan mengisi nilai properti
22         PersegiPanjang persegiPanjang = new PersegiPanjang();
23         persegiPanjang.panjang = 8;
24         persegiPanjang.lebar = 4;
25     }
26 }
```

```
26      // membuat objek Segitiga dan mengisi nilai properti
27      Segitiga mSegitiga = new Segitiga();
28      mSegitiga.alas = 12;
29      mSegitiga.tinggi = 8;
30
31
32      // memanggil method luas dan keliling
33      bangunDatar.luas();
34      bangunDatar.keliling();
35
36      persegi.luas();
37      persegi.keliling();
38
39      lingkaran.luas();
40      lingkaran.keliling();
41
42      persegiPanjang.luas();
43      persegiPanjang.keliling();
44
45      mSegitiga.luas();
46      mSegitiga.keliling();
47  }
48
49 }
```

Setelah itu, coba jalankan class Main, maka hasilnya:



```
run:
Menghitung laus bangun datar
Menghitung keliling bangun datar
Menghitung laus bangun datar
Menghitung keliling bangun datar
Menghitung laus bangun datar
Menghitung keliling bangun datar
Menghitung laus bangun datar
Menghitung keliling bangun datar
Menghitung laus bangun datar
Menghitung keliling bangun datar
BUILD SUCCESSFUL (total time: 0 seconds)
```

Kenapa hasilnya bisa begitu? Karena yang kita panggil sebenarnya adalah method `luas()` dan `keliling()` milik si induk (`BangunDatar`). Objek anak dari `BangunDatar` belum memiliki method `luas()` dan `keliling()`, akhirnya mengambil milik induknya.

Lalu bagaimana kalau kita ingin membuat agar semua class anak memiliki method `luas()` dan `keliling()` yang berbeda dari induk? Jawabannya: **menggunakan *method overriding***.

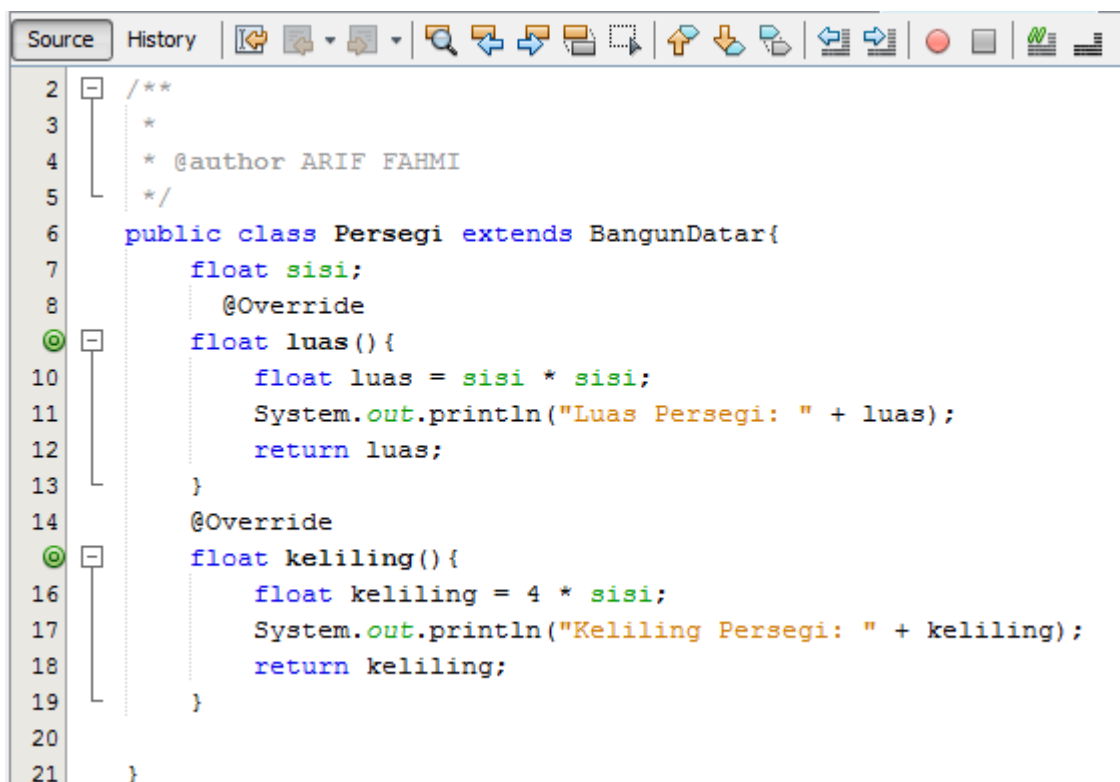
Sesuai dari materi diatas method overriding dapat dibuat dengan menambahkan anotasi **@Override** di atas nama method atau sebelum pembuatan method.

a. Persegi.java

```
Public class Persegi extends BangunDatar {
    float sisi;

    @Override
    float luas(){
        float luas = sisi * sisi;
        System.out.println("Luas Persegi: " + luas);
        return luas;
    }

    @Override
    float keliling(){
        float keliling = 4 * sisi;
        System.out.println("Keliling Persegi: " + keliling);
        return keliling;
    }
}
```



```
Source History
2  /**
3   *
4   * @author ARIF FAHMI
5   */
6  public class Persegi extends BangunDatar{
7      float sisi;
8      @Override
9      float luas(){
10         float luas = sisi * sisi;
11         System.out.println("Luas Persegi: " + luas);
12         return luas;
13     }
14     @Override
15     float keliling(){
16         float keliling = 4 * sisi;
17         System.out.println("Keliling Persegi: " + keliling);
18         return keliling;
19     }
20 }
21 }
```

Artinya kita menulis ulang method luas() dan keliling() di class anak. Sekarang mari kita buat *method overriding* untuk semua class anak.

b. Lingkaran.Java

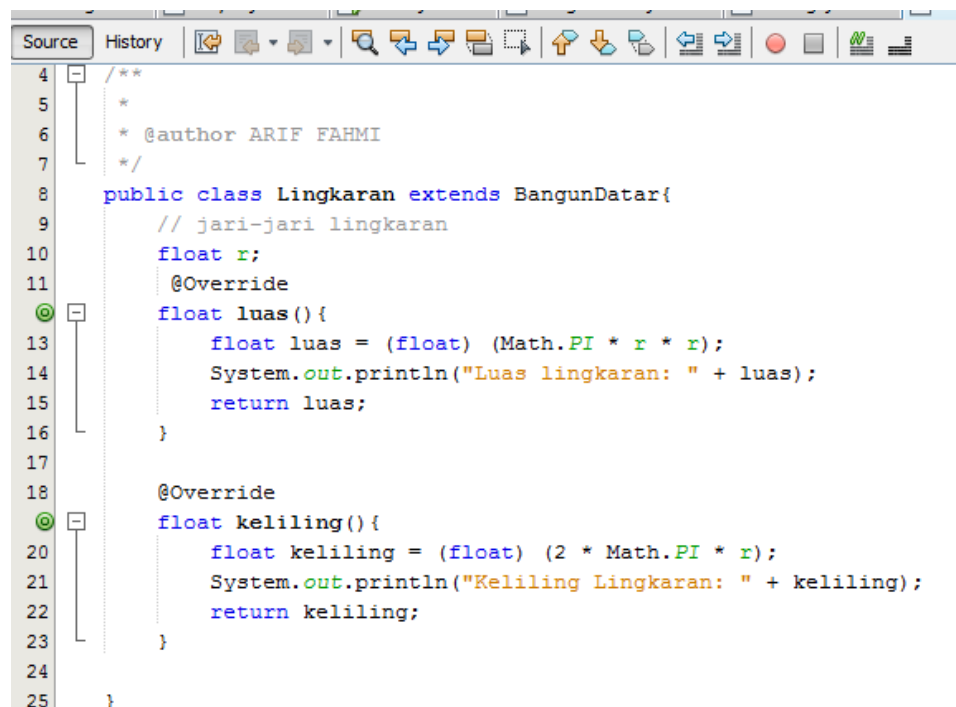
```
public class Lingkaran extends BangunDatar{

    // jari-jari lingkaran
    float r;

    @Override
    float luas(){
        float luas = (float) (Math.PI * r * r);
        System.out.println("Luas lingkaran: " + luas);
        return luas;
    }

    @Override
    float keliling(){
        float keliling = (float) (2 * Math.PI * r);
        System.out.println("Keliling Lingkaran: " + keliling);
        return keliling;
    }

}
```



Dalam rumus luas dan keliling lingkaran, kita bisa memanfaatkan konstanta **Math.PI** sebagai nilai PI. Konstanta ini sudah ada di Java.

c. PersegiPanjang.Java

```
public class PersegiPanjang extends BangunDatar {
    float panjang;
    float lebar;

    @Override
    float luas(){
        float luas = panjang * lebar;
        System.out.println("Luas Persegi Panjang:" + luas);
        return luas;
    }

    @Override
    float keliling(){
        float kll = 2*panjang + 2*lebar;
        System.out.println("Keliling Persegi Panjang: " + kll);
        return kll;
    }

}
```

```
Source History [Icons]
4  /**
5   *
6   * @author ARIF FAHMI
7   */
8  public class PersegiPanjang extends BangunDatar{
9      float panjang;
10     float lebar;
11     @Override
12     float luas() {
13         float luas = panjang * lebar;
14         System.out.println("Luas Persegi Panjang:" + luas);
15         return luas;
16     }
17
18     @Override
19     float keliling() {
20         float kll = 2*panjang + 2*lebar;
21         System.out.println("Keliling Persegi Panjang: " + kll);
22         return kll;
23     }
24 }
```

d. Segitiga.java

```
public class Segitiga extends BangunDatar {

    float alas;
    float tinggi;

    @Override
    float luas() {
        float luas = 1/2 * (alas * tinggi);
        System.out.println("Luas Segitiga: " + luas);
        return luas;
    }
}
```

```
Source History [Icons]
4  /**
5   *
6   * @author ARIF FAHMI
7   */
8  public class Segitiga extends BangunDatar{
9      float alas;
10     float tinggi;
11     @Override
12     float luas() {
13         float luas = 1/2 * (alas * tinggi);
14         System.out.println("Luas Segitiga: " + luas);
15         return luas;
16     }
17 }
```

Untuk class Segitiga, kita hanya melakukan *override* terhadap method luas() saja. Karena untuk method keliling(), segitiga memiliki rumus yang berbeda-beda.

e. **Main.java**

```
Source History
3  /**
4   *
5   * @author ARIF FAHMI
6   */
7  public class Main {
8      public static void main(String[] args) {
9
10         // membuat objek bangun datar
11         BangunDatar bangunDatar = new BangunDatar();
12
13         // membuat objek persegi dan mengisi nilai properti
14         Persegi persegi = new Persegi();
15         persegi.sisi = 2;
16
17         // membuat objek Lingkaran dan mengisi nilai properti
18         Lingkaran lingkaran = new Lingkaran();
19         lingkaran.r = 22;
20
21         // membuat objek Persegi Panjang dan mengisi nilai properti
22         PersegiPanjang persegiPanjang = new PersegiPanjang();
23         persegiPanjang.panjang = 8;
24         persegiPanjang.lebar = 4;
25
26         // membuat objek Segitiga dan mengisi nilai properti
27         Segitiga mSegitiga = new Segitiga();
28         mSegitiga.alas = 12;
29         mSegitiga.tinggi = 8;
30
31
32         // memanggil method luas dan keliling
33         bangunDatar.luas();
34         bangunDatar.keliling();
35
36         persegi.luas();
37         persegi.keliling();
38
39         lingkaran.luas();
40         lingkaran.keliling();
41
42         persegiPanjang.luas();
43         persegiPanjang.keliling();
44
45         mSegitiga.luas();
46         mSegitiga.keliling();
47     }
48 }
49
50
```

```
compile-single:
run-single:
Menghitung laus bangun datar
Menghitung keliling bangun datar
Luas Persegi: 4.0
Keliling Persegi: 8.0
Luas lingkaran: 1520.5309
Keliling Lingkaran: 138.23007
Luas Persegi Panjang:32.0
Keliling Persegi Panjang: 24.0
Luas Segitiga: 96.0
Menghitung keliling bangun datar
BUILD SUCCESSFUL (total time: 0 seconds)
```

Aturan dalam Melakukan Override Method

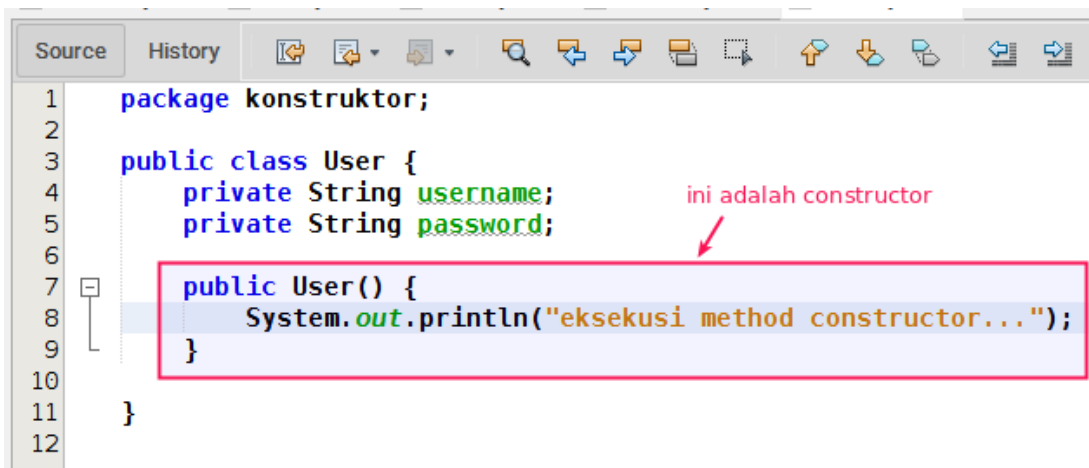
- Daftar argumen harus persis sama seperti pada method yang di override
- Tipe balikan harus sama atau merupakan subtype dari tipe balikan yang dideklarasikan dalam method asli pada superkelas
- Level akses tidak boleh lebih terbatas dari method yang di override
- Method instance dapat di override hanya jika method tersebut diwarisi oleh subkelas
- Method yang dideklarasikan final tidak bisa di override
- Konstruktor tidak bisa di override

2. Constructor

Constructor adalah method khusus yang didefinisikan didalam class dan akan dipanggil secara otomatis setiap kali terjadi instansiasi object. Biasanya method ini digunakan untuk inisialisasi atau mempersiapkan data untuk objek. Apabila kita tidak mendefinisikan constructor dalam class yang kita buat, maka secara otomatis Java akan membuatnya untuk kita. Constructor semacam ini dinamakan dengan *default constructor*. Default constructor akan menginisialisasi semua data yang ada dengan nilai nol. Namun, sekali kita mendefinisikan sebuah constructor baru untuk kelas yang kita buat, maka default constructor sudah tidak berfungsi atau tidak digunakan lagi. Sama halnya seperti method, constructor juga dapat memiliki parameter dan juga dapat di-overload.

Sehingga dapat dinyatakan bahwa Constructor adalah method khusus yang akan dieksekusi pada saat pembuatan objek (*instance*). Biasanya method ini digunakan untuk inisialisasi atau mempersiapkan data untuk objek.

Berikut ini contoh Constructor:



```
1 package konstruktor;
2
3 public class User {
4     private String username;
5     private String password;
6
7     public User() {
8         System.out.println("eksekusi method constructor...");
9     }
10
11 }
12
```

Cara membuat constructor adalah dengan menuliskan nama method constructor sama seperti nama class.

Pada contoh di atas constructor ditulis seperti ini:

```
public User() {
    System.out.println("eksekusi method constructor...");
}
```

Pastikan memberikan modifier **public** kepada Constructor, karena ia akan dieksekusi saat pembuatan objek (*instance*).

Mari kita coba membuat objek baru dari class User:

```
User petani = new User();
```

Catatan penting untuk memahami operator new!

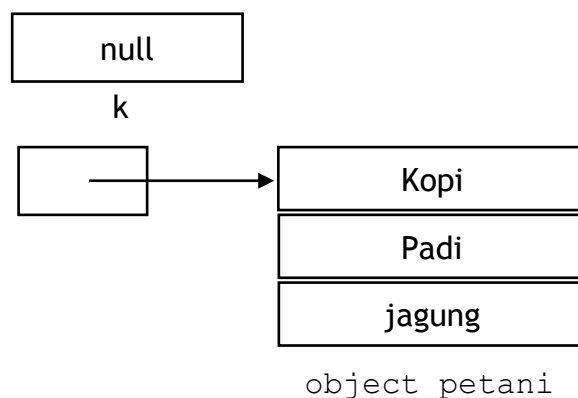
Seperti pada materi pertemuan sebelumnya, pada saat kita mendefinisikan suatu class, kita hanya membuat tipe data baru, bukan membuat object baru. Untuk membuat object baru dari tipe class yang telah didefinisikan, secara eksplisit kita melakukan dua tahap. Pertama, kita perlu mendeklarasikan variabel yang digunakan sebagai referensi ke object dari class bersangkutan. Kedua, kita perlu mengistanisasi class dengan menggunakan operator new dan memasukkan instance-nya ke dalam variabel referensi yang baru saja dideklarasikan. Operator new secara dinamis akan mengalokasikan ruang memori untuk menyimpan suatu object tertentu dan mengembalikan nilai berupa referensi ke object bersangkutan.

Statemen

```
User petani;
```

```
petani = new User ();
```

Pengaruh yang ditimbulkan



pada prakteknya, dua tahap diatas biasanya di tulis dalam satu baris, seperti berikut

```
User petani = new User();
```

melanjutkan konsep constructor, sehingga sekarang kita punya kode lengkap seperti ini:

```
User.java x
Source History
1 package konstruktor;
2
3 public class User {
4     private String username;
5     private String password;
6
7     public User() {
8         System.out.println("eksekusi method constructor...");
9     }
10
11 }
12
13
14 class DemoConstructor{
15     public static void main(String[] args) {
16         User petani = new User();
17     }
18 }
19
```



Hasilnya saat dieksekusi:

```
Output - BelajarOOP (run-single)
deps-jar:
Updating property file: /home/petanikode/NetBeansProje
Compiling 1 source file to /home/petanikode/NetBeansPro
Running javac...
compile-single:
run-single:
eksekusi method constructor...
BUILD SUCCESSFUL (total time: 0 seconds)
```

Constructor dengan Parameter

Constructor biasanya digunakan untuk *initialize* (menyiapkan) data untuk class. Untuk melakukan ini, kita harus membuat parameter sebagai inputan untuk constructor.

Contoh program,

```
public class User {
    public String username;
    public String password;

    public User(String username, String password) {
        this.username = username;
        this.password = password;
    }
}
```

Pada kode class User di atas, kita menambahkan parameter username dan password ke dalam constructor.

Berarti nanti saat kita membuat objek, kita harus menambahkan nilai parameter seperti ini:

```
User petani = new User("petanikode", "kopi");
```

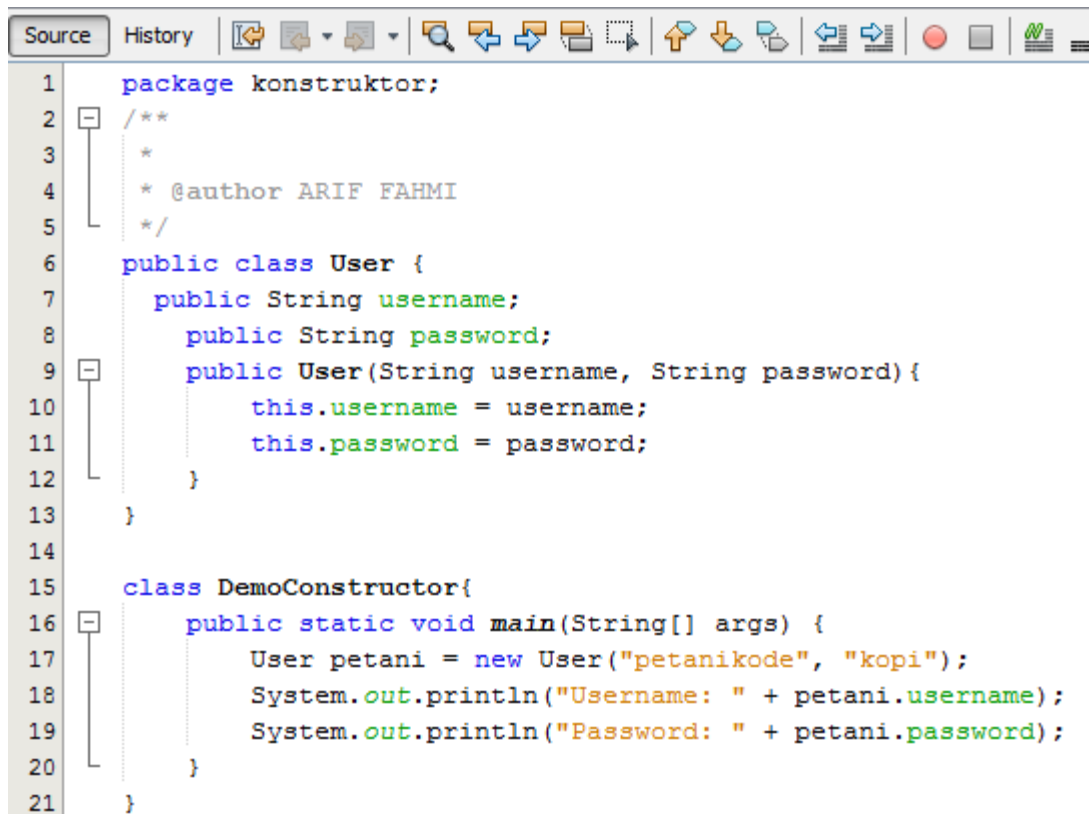
Contoh Kode lengkapnya:

```
package konstruktor;

public class User {
    public String username;
    public String password;

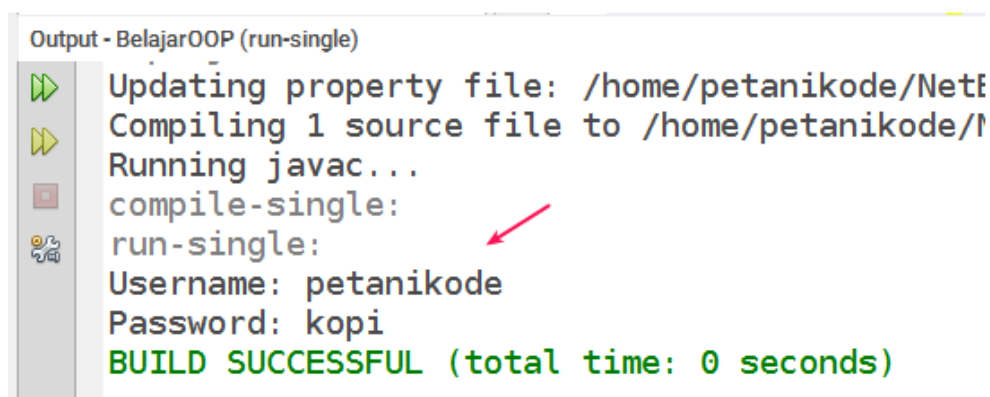
    public User(String username, String password) {
        this.username = username;
        this.password = password;
    }
}

class DemoConstructor{
    public static void main(String[] args) {
        User petani = new User("petanikode", "kopi");
        System.out.println("Username: " + petani.username);
        System.out.println("Password: " + petani.password);
    }
}
```



```
1 package konstruktor;
2 /**
3  *
4  * @author ARIF FAHMI
5  */
6 public class User {
7     public String username;
8     public String password;
9     public User(String username, String password){
10         this.username = username;
11         this.password = password;
12     }
13 }
14
15 class DemoConstructor{
16     public static void main(String[] args) {
17         User petani = new User("petanikode", "kopi");
18         System.out.println("Username: " + petani.username);
19         System.out.println("Password: " + petani.password);
20     }
21 }
```

Hasil outputnya:



```
Output - BelajarOOP (run-single)
Updating property file: /home/petanikode/.Netf
Compiling 1 source file to /home/petanikode/
Running javac...
compile-single:
run-single:
Username: petanikode
Password: kopi
BUILD SUCCESSFUL (total time: 0 seconds)
```

3. Destructor

Destructor adalah method khusus yang akan dieksekusi saat objek dihapus dari memori. Java sendiri tidak memiliki method destructor, karena Java menggunakan **gerbage collector** untuk manajemen memorinya. Jadi Si **gerbage collector** akan otomatis menghapus objek yang tidak terpakai.

A. Gerbage Collector

Garbage Collection pertama kali ditemukan oleh John McCarthy pada sekitar tahun 1959 untuk melakukan abstraksi terhadap manajemen memori manual pada bahasa pemrograman Lisp.

Garbage Collector adalah salah satu mekanisme dan fitur dari JVM (Java Virtual Machine), untuk meningkatkan Memory Management, yang akan menghapus objek secara otomatis pada memory, jika tidak dibutuhkan lagi. Objek yang akan kita buat dalam coding Java, jika kita tidak menggunakan dan mereferensikan lagi, kita tidak lagi menulis coding secara eksplisit, seperti pada bahasa pemrograman C/C++. Ini adalah salah satu keunggulan bahasa pemrograman Java, yang memudahkan para programmer dan developer, untuk efisiensi pada memori.

Keunggulan Garbage Collector pada Java

1. Programmer atau Developer tidak perlu khawatir ,tentang object yang tidak di referensikan
2. Ini akan secara otomatis terkontrol oleh JVM (Java Virtual Machine).
3. Meningkatkan efisiensi pada memori

Akan tetapi kita bisa melihat Cara Kerja Garbage Collector. Dengan menulis beberapa codingan ,dengan method *finalize()* , dan juga *System.gc()* Sehingga kalian dapat melihat contoh dari output Garbage Collector. Codinganya seperti berikut :

```
1 public class Pisang {
2
3     /**
4      * @param args the command line arguments
5      */
6
7     public void finalize() {
8         System.out.println("Garbage Collector Berhasil Berkumpul");
9     }
10    //Akan ditampilkan jika Garbage Collector bekerja
11
12    public static void main(String[] args) {
13
14
15        /*1*/Pisang satu = new Pisang();
16        /*2*/Pisang dua = new Pisang();
17        /*3*/Pisang tiga = satu;
18        /*4*/satu = null;
19        /*5*/dua = null;
20        /*6*/System.gc();
21        System.out.println("Gargabe Collector Siap !");
22    }
23 }
24
25 }
```

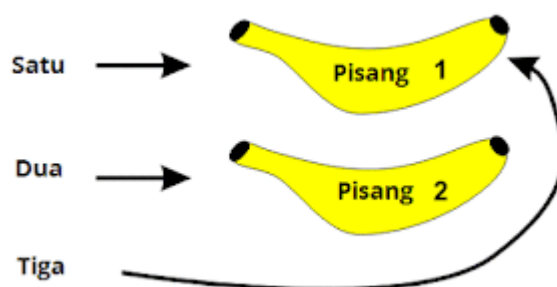
Output :

Gargabe Collector Siap !

Garbage Collector Berhasil Berkumpul

Untuk lebih jelasnya bisa lihat contoh gambar berikut :

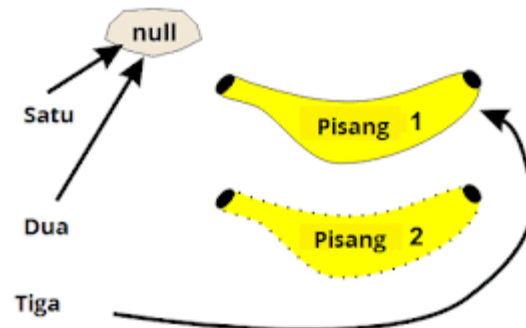
Jika kita lihat baris 15 - 17, codingan diatas akan terlihat , seperti pada gambar dibawah:



Pada class Pisang ,kita akan membuat beberapa variable , dari satu ,dua ,dan tiga. Pada variable satu dan dua , kita akan menginisialisasi object ke dalam Pisang 1 dan Pisang 2.

Pada variable tiga ,kita akan menentukan nilai kembali ke dalam object Pisang 1.

Nah pada baris selanjutnya , kita menentukan variable satu dan dua ,ke dalam *null* artinya kosong tidak ada nilai. Nah pada variable tiga , kita masih menentukan nilainya ke dalam object Pisang 1.



Nah karena object Pisang 2 ,tidak ada variable manapun ,yang di referensikan nilainya. Jadi object Pisang 2 ,layak diangkut oleh system Garbage Collector ,pada JVM (Java Virtual Machine).

Contoh program selanjutnya,

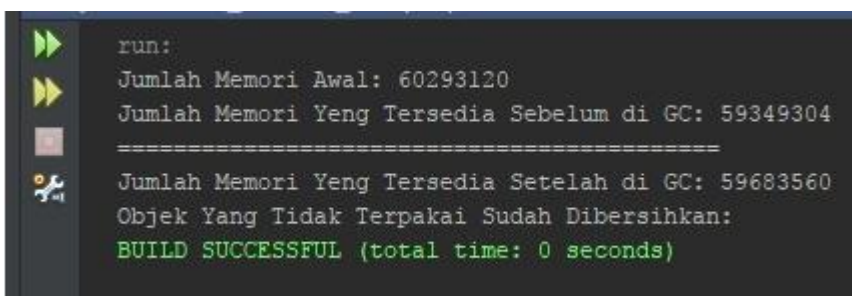
```
Package Garbage Collection;

public class Balok {

    //Method ini akan ditampilkan jika Garbage Collection bekerja
    public void finalize(){
        System.out.println("Objek Yang Tidak Terpakai Sudah Dibersihkan:");
    }

    public static void main(String[] args){
        Runtime RT = Runtime.getRuntime();
        System.out.println("Jumlah Memori Awal: "+RT.totalMemory());

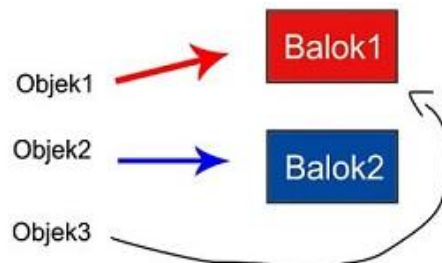
        Block objek1 = new Block(); //Block Satu
        Block objek2 = new Block(); //Block Dua
        Block objek3 = objek1; //Objek3 Mengembalikan Nilai Objek1
        objek1 = null;
        objek2 = null;
        System.out.println("Jumlah Memori Yeng Tersedia Sebelum di GC:
"+RT.freeMemory());
        System.gc();
        System.out.println("=====");
        System.out.println("Jumlah Memori Yeng Tersedia Setelah di GC:
"+RT.freeMemory());
    }
}
```



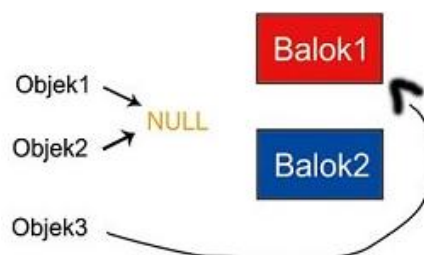
```
run:
Jumlah Memori Awal: 60293120
Jumlah Memori Yeng Tersedia Sebelum di GC: 59349304
=====
Jumlah Memori Yeng Tersedia Setelah di GC: 59683560
Objek Yang Tidak Terpakai Sudah Dibersihkan:
BUILD SUCCESSFUL (total time: 0 seconds)
```


Penjelasan:

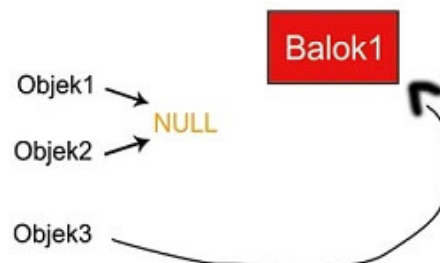
Pada program tersebut, disana kita membuat 3 buah objek atau variable dari Class Balok, objek1 dan objek2 diinisialisasikan kedalam Balok1 dan Balok2, dan objek3 mempunyai nilai kembali dari objek Balok1.



Pada baris selanjutnya, objek1 dan objek2 kita ubah nilainya menjadi *null* yang berarti kosong, sedangkan objek3 masih mengembalikan nilainya objek1.



Karena pada Balok2 tidak ada yang direferensikan nilainya, maka secara otomatis, Balok2 tersebut akan di bawa oleh System Garbage Collection pada JVM.



System.gc(), berfungsi untuk menjalankan Garbage Collection pada java. lalu Method **finalize()**, digunakan untuk memastikan bahwa objek telah bersih dan Garbage Collection telah bekerja, Di dalam metode **finalize()**, kita dapat menentukan tindakan yang harus dilakukan sebelum suatu objek hancur. Dan terakhir, kita menambahkan sebuah variable *Runtime*, yang digunakan untuk mengecek Total Memori serta Jumlah Memori yang tersedia sebelum dan sesudah di Garbage Collection.



Tugas

Catatan.

- a. Tugas dikumpulkan maks pada hari Kamis tgl 24-04-2020 pukul 16.00
- b. Tugas boleh dikumpulkan dalam bentuk (PDF,PPT,WORD)
- c. File dikirim ke email : fahmi03031995@gmail.com
- d. Tugas program dapat dikerjakan di aplikasi (Dcoder, java ide for android, maupun software Netbeans,atau Notepad)
- e. Format file dengan tata cara: NamaMhs_NIM_PBO

Soal

1. Buatlah sebuah pemrograman Java dengan menerapkan konsep “ method Overriding” sesuai kreatifitas kalian.

Catatan : setelah program dibuat jelaskan secara terperinci maksud dan fungsi dari program yang kamu buat