

# PEMROGRAMAN BERORIENTASI OBYEK

## Konsep Pemrograman Berorientasi Obyek (PBO)

Faisal Fajri Rahani, S.Si., M.Cs.

Teknik Informatika

# Konsep Pemrograman Berorientasi Obyek (PBO)

- Konstruktor
- Enkapsulasi & Hidding Information

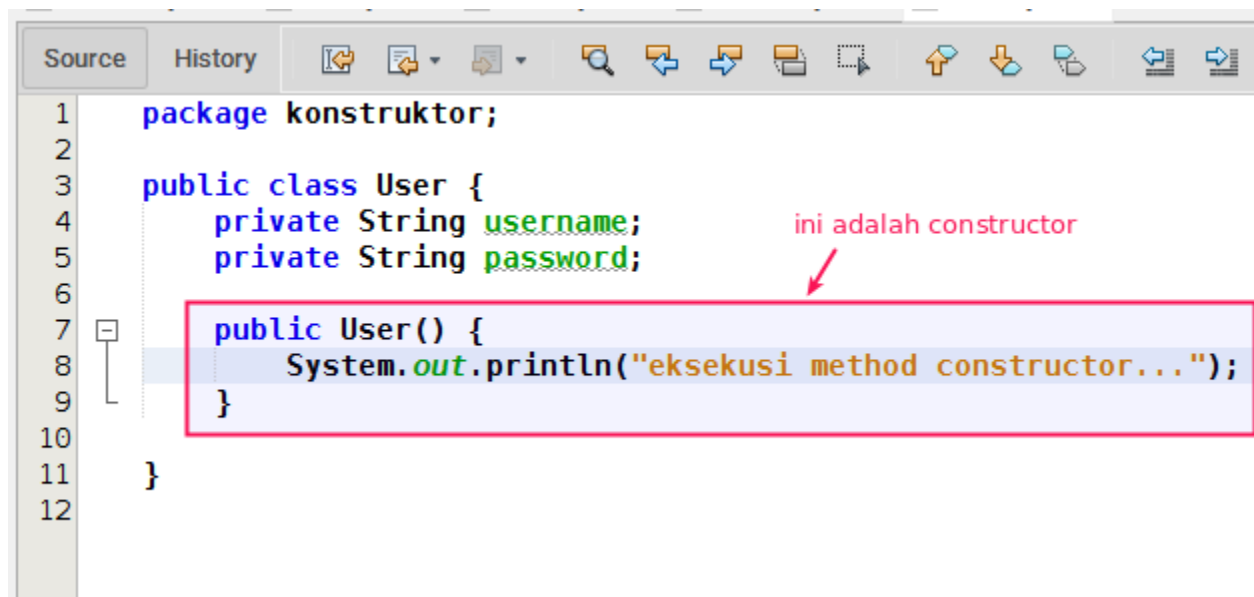
# Konstruktor

# Apa itu Constructor?

Constructor adalah method khusus yang akan dieksekusi pada saat pembuatan objek (instance).

Biasanya method ini digunakan untuk inisialisasi atau mempersiapkan data untuk objek.

# Contoh Constructor dan Cara Membuatnya



```
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
```

ini adalah constructor

- Cara membuat constructor adalah dengan menuliskan nama method constructor sama seperti nama class.
- Pada contoh di slide sebelumnya constructor ditulis seperti ini:

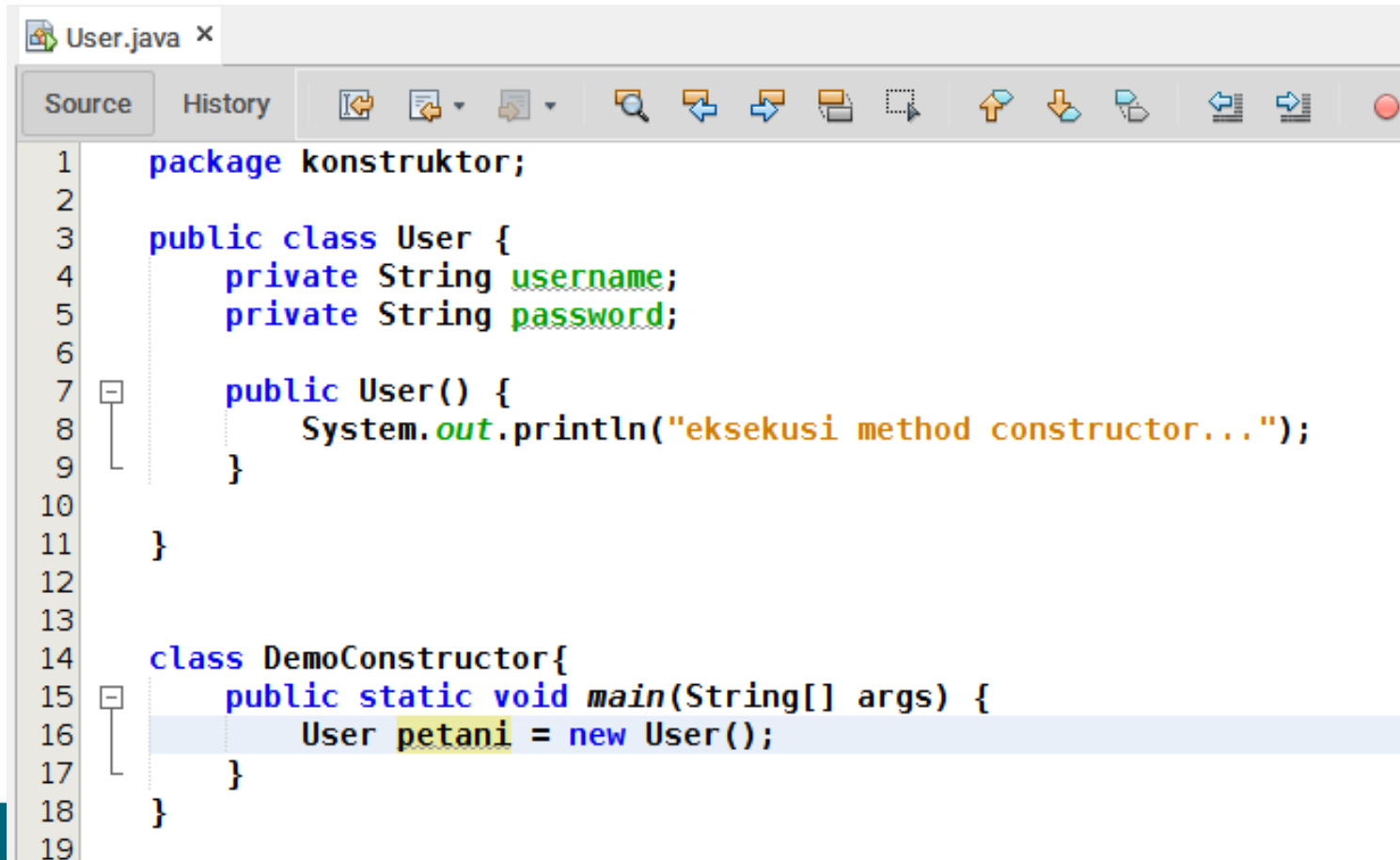
```
public User() {  
    System.out.println("eksekusi method constructor...");  
}
```

- 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();
```

- Sehingga sekarang kita punya kode lengkap seperti ini:



```
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.

```
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);
    }
}
```

# Hasil outputnya

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

# Destructor dalam Java

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

- Sementara untuk bahasa pemrograman lain, seperti C++ kita bisa membuat destructor sendiri seperti ini:

```
class User {  
public:  
    User( String *username ); // <-- ini constructor  
    ~User(); // <-- ini destructor.  
private:  
    String username;  
    String password;  
};
```

# Enkapsulasi & Hidding Information

- Enkapsulasi merupakan suatu cara pembungkusan data dan method yang menyusun suatu kelas sehingga kelas dapat dipandang sebagai suatu modul dan cara bagaimana menyembunyikan informasi detail dari suatu class (information hiding).
- Dalam OOP, enkapsulasi sangat penting untuk keamanan serta menghindari kesalahan pemrograman, enkapsulasi dimaksudkan untuk menjaga suatu proses program agar tidak dapat diakses secara sembarangan atau diintervensi oleh program lain.



# Dua hal yang mendasar dalam enkapsulasi (1)

- ***Information hiding***

- Sebelumnya untuk pengaksesan atribut atau method menggunakan objek secara langsung.
- Hal ini karena akses kontrol yang diberikan pada atribut dan method di dalam kelas tersebut adalah public.
- Untuk menyembunyikan informasi dari suatu kelas sehingga anggota kelas tersebut tidak dapat diakses kelas lain yaitu dengan memberi hak akses private pada atributnya.
- Proses ini disebut dengan information hiding.

# Dua hal yang mendasar dalam enkapsulasi (2)

- ***Interface to access data***

- *Interface to access data* ini merupakan cara melakukan perubahan terhadap atribut yang disembunyikan, caranya adalah dengan membuat suatu interface berupa method untuk menginisialisasi atau merubah nilai dari suatu atribut tersebut.
- Manfaat utama teknik encapsulation adalah kita mampu memodifikasi kode tanpa merusak kode yang telah digunakan pada class lain.

# Manfaat Enkapsulasi

- Modularitas
  - Source code dari sebuah class dapat dikelola secara independen dari source code class yang lain. Perubahan internal pada sebuah class tidak akan berpengaruh bagi class yang menggunakannya.
- Information Hiding
  - Penyembunyian informasi yang tidak perlu diketahui objek lain.

# Keuntungan menerapkan Encapsulasi (1)

- Bersifat independen
  - Suatu modul yang terencapsulasi dengan baik akan bersifat independen, sehingga tidak akan terikat pada bagian tertentu dari program.
- Bersifat transparan
  - Bila melakukan modifikasi pada suatu modul, maka perubahan tersebut akan dirasakan juga oleh bagian program yang menggunakan modul tersebut.

# Keuntungan menerapkan Encapsulasi (2)

- Menghindari efek diluar perencanaan
  - Modul yang terencapsulasi dengan baik hanya akan berinteraksi dengan bagian program lainnya melalui variable-variabel input/output yang telah didefinisikan sebelumnya.
- Melindungi listing program
  - Saat program didistribusikan pada khalayak, untuk melindungi listing program Anda dapat menerapkan prinsip enkapsulasi. Di sini pengguna hanya dapat menggunakan program melalui variable input atau output yang didefinisikan tanpa disertai bagaimana proses yang terjadi di dalam modul tersebut.

# Contoh

```
public class Siswa {  
    private int nrp;  
    public String nama;  
  
    public void Info() {  
        System.out.println("Saya siswa PENS");  
    }  
}
```

1	public class IsiData {
2	public static void main(String args[]) {
3	Siswa IT2=new Siswa();
4	<b>IT2.nrp=5;</b>
5	IT2.nama="Andi";
6	IT2.Info();
7	}
8	}

### **Hasil Runing ?**

Hallo.java:4: nrp has private access in Siswa  
IT2.nrp=5;



# Contoh Enkapsulasi

- Terdapat Class Circle.

```
class MyMain
{
    public static void main(String args[])
    {
        Circle aCircle; // creating reference
        aCircle = new Circle(); // creating object
        aCircle.setX(10);
        aCircle.setY(20);
        aCircle.setR(5);
        double area = aCircle.area(); // invoking method
        double circumf = aCircle.circumference();
        System.out.println("Radius="+aCircle.getR()+" Area="+area);
        System.out.println("Radius="+aCircle.getR()+" Circumference =" +circumf);
    }
}
```

Untuk memberikan nilai X dengan menggunakan cara **aCircle.setX(10);** begitu juga dengan nilai Y dan Z. Untuk mendapatkan jari-jari menggunakan **aCircle.getR()**

# Terima Kasih