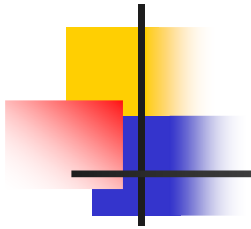
# Inheritance



# Topik

- Subclasses
- Inheritance & Variabel
- Method Overriding
- Inheritance & Methods
- Inheritance & Constructors
- Class Modifier
- Variables Modifier
- Constructor Modifier
- Method Modifier

# Subclass

## Inheritance dari Class

*class namaClass2 extends namaClass1 {*
*// Body of class*
*}*

## Deklarasi variabel Class

*namaClass namaVar;*

```
class X {
}

class X1 extends X {
}

class X2 extends X {
}

class X11 extends X1 {
}
```

Instantiating

Inherit

```
  class X12 extends X1 {
}

class X21 extends X2 {
}

class X22 extends X2 {
}

class InheritanceHierarchy {
  public static void main(String args[]) {
    X x;
    System.out.println("Instantiating X");
    x = new X();
    System.out.println("Instantiating X1");
    x = new X1();
    System.out.println("Instantiating X11");
    x = new X11();
    System.out.println("Instantiating X12");
    x = new X12();
    System.out.println("Instantiating X2");
    x = new X2();
    System.out.println("Instantiating X21");
    x = new X21();
    System.out.println("Instantiating X22");
    x = new X22();
  }
}
```

# Subclasses

## Referensi ke Variabel dari Subclass

*super.namaVar*

Inherited Variables

```
class W {
  float f;
}

class X extends W {
  StringBuffer sb;
}

class Y extends X {
  String s;
}

class Z extends Y {
  Integer i;
}
```

Inherit

```
class Wxyz {
  public static void main(String args[]) {
    Z z = new Z();
    z.f = 4.567f;
    z.sb = new StringBuffer("abcde");
    z.s = "Belajar Java";
    z.i = new Integer(41);
    System.out.println("z.f = " + z.f);
    System.out.println("z.sb = " + z.sb);
    System.out.println("z.s = " + z.s);
    System.out.println("z.i = " + z.i);
  }
}
```

Hasil :

z.f = 4.567

z.sb = abcde

z.s = Belajar Java

z.i = 41

# Subclass

```
class E {
  int x;
}

class F extends E {
  String x;
}

class Ef {
  public static void main(String args[]) {
    F f = new F();
    f.x = “Ini adalah string";
    System.out.println("f.x = " + f.x);
    E e = new E();
    e.x = 45;
    System.out.println("e.x = " + e.x);
  }
}
```

Hasil :

f.x = Ini adalah string

e.x = 45

```
class P {
  static int x;
}

class Q extends P {
  static String x;
}

class Pq {
  public static void main(String args[]) {
    P p = new P();
    p.x = 55;
    System.out.println("p.x = " + p.x);
    Q q = new Q();
    q.x = “Ini adalah string";
    System.out.println("q.x = " + q.x);
  }
}
```

Hasil :

p.x = 55

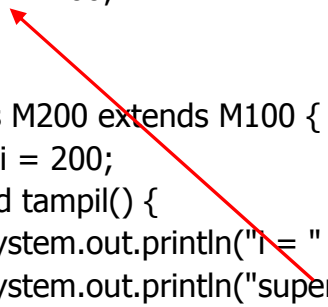
q.x = Ini adalah string

# Subclass

```
class M100 {
  int i = 100;
}

class M200 extends M100 {
  int i = 200;
  void tampil() {
    System.out.println("i = " + i);
    System.out.println("super.i = " + super.i);
  }
}

class SuperKeyword {
  public static void main(String args[]) {
    M200 m200 = new M200();
    m200.tampil();
  }
}
```



Hasil :

i = 200

super.i = 100

# Method Overriding

## Overloading & Overriding

*Overriding : Pada saat method dengan signature yang sama didefinisikan dalam Subclass*

*Overloading : Hanya namanya yang sama, tapi berbeda jumlah maupun tipe parameternya*

```
class A1 {
  void hello() {
    System.out.println("Hello dari A1");
  }
}

class B1 extends A1 {
  void hello() {
    System.out.println("Hello dari B1");
  }
}
```

Di-Overrided oleh hello-nya B1

```
class C1 extends B1 {
  void hello() {
    System.out.println("Hello dari C1");
    super.hello();
  }
}

class MethodOverriding1 {
  public static void main(String args[]) {
    C1 obj = new C1();
    obj.hello();
  }
}
```

Hasil :

Hello dari C1

Hello dari B1

# Method Overriding

```
class A2 {
  void hello() {
    System.out.println("Hello dari A2");
  }
}

class B2 extends A2 {
  void hello() {
    System.out.println("Hello dari B2");
  }
}

class C2 extends B2 {
  void hello() {
    System.out.println("Hello dari C2");
  }
}

class MethodOverriding2 {
  public static void main(String args[]) {
    A2 obj = new C2();
    obj.hello();
  }
}
```

Hasil :

Hello dari C2

Overrided oleh hellonya B2

Object dari C2

# Inheritance dan Methods

## Referensi Methods dalam Superclass

***super**.namaMethod(args)*

```
class I1 {
  void hello(String s) {
    System.out.println("I1: " + s);
  }
}

class J1 extends I1 {
  void hello(String s) {
    super.hello(s);
    System.out.println("J1: " + s);
  }
}

class K1 extends J1 {
  void hello(String s) {
    super.hello(s);
    System.out.println("K1: " + s);
  }
}
```

```
class SuperForMethods1 {

  public static void main(String args[]) {

    System.out.println("Instantiating I1");
    I1 obj = new I1();
    obj.hello("Selamat pagi");

    System.out.println("Instantiating J1");
    obj = new J1();
    obj.hello("Selamat siang");

    System.out.println("Instantiating K1");
    obj = new K1();
    obj.hello("Selamat malam");
  }
}
```

```
Hasil :
Instantiating I1
I1: Selamat pagi
Instantiating J1
I1: Selamat siang
J1: Selamat siang
Instantiating K1
I1: Selamat malam
J1: Selamat malam
K1: Selamat malam
```

# Inheritance and Constructors

## Pemanggilan Super-Constructor

*super* (*args*)

## Pemanggilan Constructor dari Class yang sama

*this* (*args*)

```
class S1 {
  int s1;
  S1() {
    System.out.println("S1 Constructor");
    s1 = 1;
  }
}

class T1 extends S1 {
  int t1;
  T1() {
    System.out.println("T1 Constructor");
    t1 = 2;
  }
}
```

```
class U1 extends T1 {
  int u1;
  U1() {
    System.out.println("U1 Constructor");
    u1 = 3;
  }
}

class InheritanceDanConstructors {
  public static void main(String args[]) {
    U1 u1 = new U1();
    System.out.println("u1.s1 = " + u1.s1);
    System.out.println("u1.t1 = " + u1.t1);
    System.out.println("u1.u1 = " + u1.u1);
  }
}
```

Hasil :
S1 Constructor
T1 Constructor
U1 Constructor
u1.s1 = 1
u1.t1 = 2
u1.u1 = 3

# Inheritance dan Constructors

## Urut-urutan Constructor

1) ***Constructor dari Super Class***
2) ***Inisialisasi Field***
3) ***Constructor Body***

```
class S2 {
  int s2;
  S2(int s2) {
    this.s2 = s2;
  }
}

class T2 extends S2 {
  int t2;
  T2(int s2, int t2) {
    super(s2);
    this.t2 = t2;
  }
}
```

```
class U2 extends T2 {
  int u2;
  U2(int s2, int t2, int u2) {
    super(s2, t2);
    this.u2 = u2;
  }
}

class InheritanceAndConstructors2 {
  public static void main(String args[]) {
    U2 u2 = new U2(1, 2, 3);
    System.out.println("u2.s2 = " + u2.s2);
    System.out.println("u2.t2 = " + u2.t2);
    System.out.println("u2.u2 = " + u2.u2);
  }
}
```

```
Hasil :
u2.s2 = 1
u2.t2 = 2
u2.u2 = 3
```

# Class Modifier

## Qualifier dari Class

**abstract : tidak dapat di-instantiate**

**final : tidak dapat di-extend**

**public : dapat direfer dari semua class-**

## Abstract Class

**Class yang memiliki abstract method(s).**

**Tidak punya detail implementasi, dan dapat diimplementasikan dalam subclass**

```
abstract class Bentuk {
  void tampil() {
  }
}
class Lingkaran extends Bentuk {
  void tampil() {
    System.out.println("Lingkaran");
  }
}
```

```
class Kotak extends Bentuk {
  void tampil() {
    System.out.println("Kotak");
  }
}
class Segitiga extends Bentuk {
  void tampil() {
    System.out.println("Segitiga");
  }
}
class AbstractClassDemo {
  public static void main(String args[]) {
    Bentuk s = new Lingkaran();
    s.tampil();
    s = new Kotak();
    s.tampil();
    s = new Segitiga();
    s.tampil();
  }
}
```

Hasil :

Lingkaran

Kotak

Segitiga

# Variable Modifier

## Variable Modifier

**final : sebuah konstanta**

**private : dapat diakses hanya pada class yang sama**

**protected : dapat diakses dari subclass dan pada package yang sama**

**public : dapat direfer dari semua class**

**static : bukan instance variable**

```
class L {
  static final int x = 5;
}

class FinalVariable {
  public static void main(String args[]) {
    System.out.println(L.x);
  }
}
```

Hasil :

5

# Constructor Modifier

## Qualifier dari Constructor

**private : hanya dapat diakses pada class yang sama**

**protected : dapat diakses dari subclass dan package yang sama**

**public : dapat direfer dari semua class**

```
class Orang {
  String nama;
  int umur;

  public Orang(String nama, int umur) {
    this.nama = nama;
    this.umur = umur;
  }

  private Orang() {
  }
}
```

```
class PrivateConstructor {

  public static void main(String args[]) {

    // pemanggilan public Constructor
    Orang p1 = new Orang("John", 30);
    System.out.println(p1.nama);
    System.out.println(p1.umur);

    // pemanggilan private constructor
    // Orang p2 = new Orang();
  }
}
```

Jika komentar ini dihilangkan dan jadi statement, apa hasilnya ?

Hasil sementara :

John

30

# Method Modifier

## Method Modifier

**abstract : method tanpa implementasi**

**final: tidak dapat di-override**

**native : kode mesin semisal C++**

**private : dapat diakses hanya pada class yang sama**

**protected : dapat diakses dari subclass dan package yang sama**

**public : dapat direfer dari semua class**

**static : bukan instance method**

```
abstract class Pesawat {
  abstract int jumlahMesin();
}
class DC8 extends Pesawat {
  int jumlahMesin() {
    return 4;
  }
}
```

```
class DC10 extends Pesawat  {
  int jumlahMesin() {
    return 3;
  }
}

class Pesawatku {
  public static void main(String args[]) {
    System.out.println(new DC8().jumlahMesin());
    System.out.println(new DC10().jumlahMesin());
  }
}
```

Hasil :

4

3

# Method Modifier

```
class Periksa {
  static Periksa periksa;

  private Periksa() {
  }

  public static Periksa getInstance() {
    if (periksa == null)
      periksa = new Periksa();
    return periksa;
  }
}

class PeriksaDemo {
  public static void main(String args[]) {
    Periksa s1 = Periksa.getInstance();
    Periksa s2 = Periksa.getInstance();
    if (s1 == s2)
      System.out.println(“Sama");
    else
      System.out.println(“Tidak sama");
  }
}
```

Hasil :

Sama