PERTEMUAN 12
KONSEP PEMROGRAMAN BERORIENTASI OBJEK
INHERITANTANCE/PEWARISAN

A. TUJUAN PEMBELAJARAN

1. Mahasiswa memahami Konsep OOP *constructor Inheritance*
2. Mahasiswa memahami cara kerja dan jenis jenis *Inheritance*
3. Mahasiswa dapat mempraktekan penggunaan *Inheritance* dalam bahasa java
4. Mahasiswa dapat mempraktekan dan membedakan jenis jenis *Inheritance* dalam bahasa java

B. URAIAN MATERI

Inheritance atau pewarisan pada Jawa adalah mekanisme di mana satu objek *child class* memperoleh semua properti dan perilaku atau method dari *parent class*. Ini adalah bagian penting dari OOP (sistem pemrograman Berorientasi Objek).

Inheritance memungkinkan kita untuk menggunakan kembali kode, serta meningkatkan reusability dalam aplikasi java. Keuntungan terbesar dari *Inheritance* adalah bahwa kode yang sudah ada di *base* atau *parent class* tidak perlu ditulis ulang di *child class*.

***Child class* :**

Adalah class yang memperluas fitur dari *class* lain dikenal sebagai kelas *Child class*, subclass atau class turunan.

Parent Class :

Class dimana properti dan fungsinya digunakan (diwarisi) oleh class lain dikenal sebagai kelas Parent Class super, atau Base class.

1. Bentuk Umum penulisan *Inheritance*

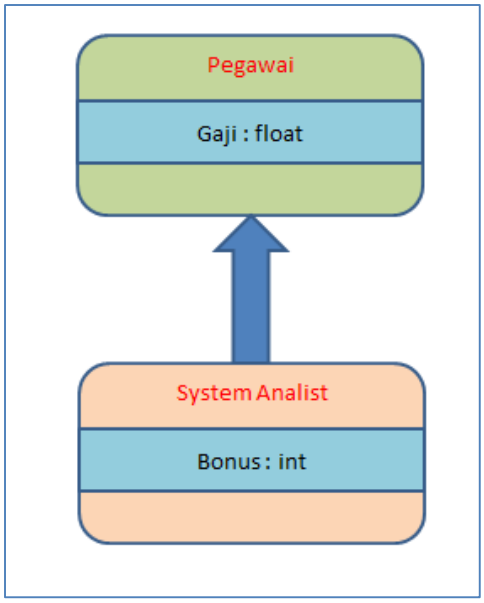
```
class XYZ extends ABC
{

}
```

Untuk mewarisi class, kita menggunakan kata kunci ***extends***. Pada penulisan diatas class **XYZ** adalah ***child class*** dan class **ABC** adalah ***parent class***. *class* XYZ mewarisi properti dan metode class ABC.

Kata kunci ***extends*** menunjukkan bahwa kita membuat *class* baru yang diturunkan dari class yang ada. Arti dari "extends" adalah untuk meningkatkan fungsionalitas.

```
Dalam terminologi Bahasa Java, kelas yang diwarisi disebut
parent atau superclass, dan class baru disebut child class
atau subclass.
```



Gambar 12. 1 ilustrasi inheritance java OOP

Seperti yang ditampilkan pada gambar di atas, **System Analyst** adalah subclass atau *child class* dan **Pegawai** adalah superclass atau *parent class* . Hubungan antara kedua *class* adalah **System Analyst** IS-A **Pegawai** .

Contoh 1 penulisan program *inheritance :*

```
class pegawai{
 float gaji=4000000;
}
```

```
class SystemAnalyst extends pegawai{
 int bonus=1000000;
 public static void main(String args[]){
   SystemAnalyst SA=new SystemAnalyst ();
   System.out.println("Gaji       System       Analyst
:"+SA.salary);
   System.out.println("Bonus                    System
Analyst:"+SA.bonus);
}
}
```

Output :

```
Gaji System Analyst : 4000000
Bonus System Analyst : 1000000
```

Contoh 2 penulisan program *inheritance :*

```
public class dosen {
  String prodi = "Sistem Informasi";
   String universitas = "Universitas Pamulang";
   void semester(){
      System.out.println("Semester 1");
   }
   }
```

```
public class ChildInherit extends dosen {
   String matakuliah1 = "PBO";
   String matakuliah2 = "Bahasa Inggris";
   String matakuliah3 = "Matematika";
```

```
   String matakuliah4 = "Agama";

    /**
     * @param args the command line arguments
     */

    public static void main(String[] args) {
        // TODO code application logic here
        ChildInherit matkul=new ChildInherit();

        System.out.println(matkul.universitas);
      System.out.println(matkul.prodi);
        matkul.semester();
      System.out.println(matkul.matakuliah1);
        System.out.println(matkul.matakuliah2);
        System.out.println(matkul.matakuliah3);
        System.out.println(matkul.matakuliah4);

    }
  }
```

Output :

```
Universitas Pamulang
Sistem Informasi
Semester 1
PBO
Bahasa Inggris
Matematika
Agama
```

Pada contoh diatas, kita memiliki class `dosen()` sebagai parent class, dan `ChildInherit` sebagai *child class*. Kemudian class `ChildInherit` di extend sebagai parent class, sehingga properti dan method di warisi oleh *parent class*, kita tidak perlu mendeklarasikan properti dan *method* ini di `ChildInherit()` lagi.

Di sini kita memiliki metode dan property untuk semua yang ada di *parent class* dengan cara ini child class seperti diatas tidak perlu menulis kode lagi (reusebale) dan dapat digunakan langsung dari *child class*.

Berdasarkan contoh di atas kita dapat mengatakan bahwa `ChildInherit()` IS-A. `dosen()` atau child class memiliki *relationship* IS-A dengan *parent class*. Inilah yang disebut *inheritance* atau pewarisan.

Catatan:

Class turunan mewarisi semua anggota dan metode yang dideklarasikan sebagai **public** atau ***protect***. Jika anggota atau metode class super dideklarasikan sebagai ***private*** maka *class* turunan tidak dapat menggunakannya secara langsung. Anggota private hanya dapat diakses di *class* nya sendiri. Anggota ***private*** tersebut hanya dapat diakses menggunakan metode ***setter*** dan ***getter*** yang dilindungi seperti yang ditunjukkan pada contoh di bawah ini.

Contoh 3 Inheritance :

Parent Class : Dosen2

```
public class dosen2 {
    private  String prodi = "Sistem Informasi";
    private     String   universitas   =   "Universitas
Pamulang";

    private String getprodi() {
      return prodi;
   }
    protected void setprodi(String prodi) {
      this.prodi = prodi;
```

```
   }


    private String getuniversitas() {
      return universitas;
   }
    protected void setuniversitas(String universitas)
{
      this.universitas = universitas;
   }



   void semester(){
      System.out.println("Semester 1");
   }


}
```

Child Class : Inheritchild2

```
public class Inheritchild2 extends dosen {
  String matakuliah1 = "PBO";
   String matakuliah2 = "Bahasa Inggris";
   String matakuliah3 = "Matematika";
   String matakuliah4 = "Agama";


    /**
     * @param args the command line arguments
     */
```

```
    public static void main(String[] args) {
        // TODO code application logic here
        InheritMain matkul=new InheritMain();
        System.out.println(matkul.universitas);
      System.out.println(matkul.prodi);
        matkul.semester();
      System.out.println(matkul.matakuliah1);
        System.out.println(matkul.matakuliah2);
        System.out.println(matkul.matakuliah3);
        System.out.println(matkul.matakuliah4);

    }
}
```

Output :

```
Universitas Pamulang
Sistem Informasi
Semester 1
PBO
Bahasa Inggris
Matematika
Agama
```

Poin penting yang perlu diperhatikan dalam contoh di atas adalah bahwa *child class* dapat mengakses anggota private ***parent class*** melalui metode ***parent class*** yang diprivate . Ketika kita membuat variabel instan atau metode yang ***protect*** , ini berarti bahwa ***parent class*** hanya dapat diakses di kelas itu sendiri dan di ***child class*** .*public , private, protect* dll semua penentu akses akan dibahas pada pertemuan mendatang.

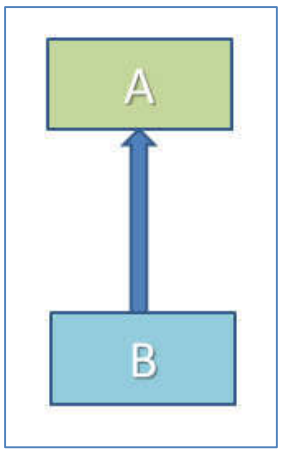
2. Jenis Jenis Inheritance

Pada *Inheritance* terdapat 4 jenis *inheritance* yaitu :

a. *Single Inheritance*
b. *Multiple Inheritance*
c. *Multilevel Inheritance*
d. *Hierarchical Inheritance*

a. Single Inheritance

Mengacu pada hubungan *child class* dan *parent class* di mana sebuah class meng extends *class* lain.



Gambar 12. 2 Single inheritance

Contoh program Single Inheritance

```
class hewan{
void makan(){System.out.println("memakan...");}
}
class kucing extends hewan{
void meong(){
   System.out.println("mengeong...");}
}
```

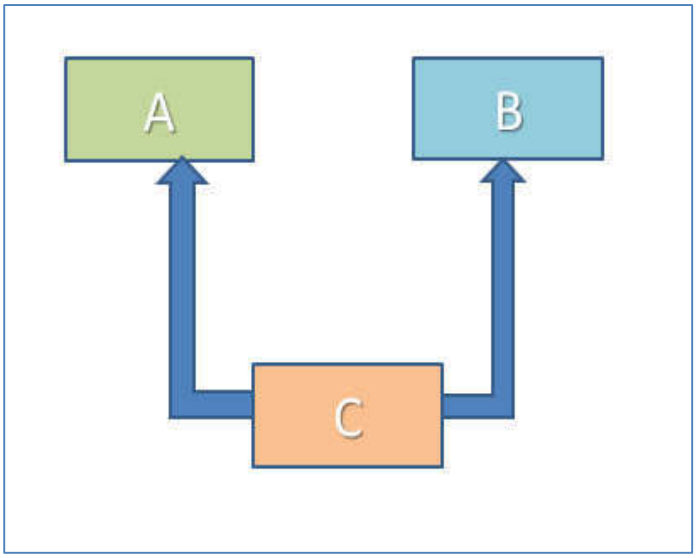
```
class testinheritance1{
public static void main(String args[]){
kucing k=new kucing();
d.meong();
d.makan();
}
}
```

Output

```
mengeong...
memakan...
```

b. Multiple *Inheritance*

Mengacu pada konsep satu *class* yang meng *extend* lebih dari satu *class*, yang berarti *child class* memiliki dua *Parent class*. Misalnya class C meng extend *class* A dan B. Pada Java tidak mendukung *Multiple Inheritance*

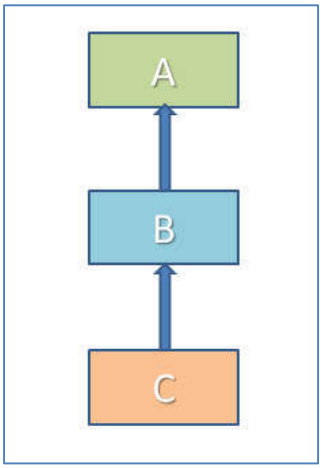


Gambar 12. 3 Multiple inheritance

Kenapa *Multiple inheritance* tidak didukung java*?* Pertimbangannya skenario di mana A, B, dan C adalah tiga class. Class C mewarisi kelas A dan B. Jika kelas A dan B memiliki metode yang sama dan Anda memanggilnya dari objek *child class*, akan ada ambiguitas untuk memanggil metode kelas A atau B.

c. Multilevel *Inheritance*

Mengacu pada hubungan *child class* dan *parent class* berdasarkan level di mana *parent class* meng extends child class. Misalnya kelas C meng-extend kelas B dan kelas B meng-extend kelas A



Gambar 12. 4 Multilevel inheritance

Contoh program *Multilevel inheritance*

```
class hewan{
void makan(){System.out.println("menyusui...");}
}
class kucing extends hewan{
void meong(){System.out.println("mengeong...");}
}
class bayikucing extends kucing{
void tangis(){System.out.println("menangis...");}
}
```

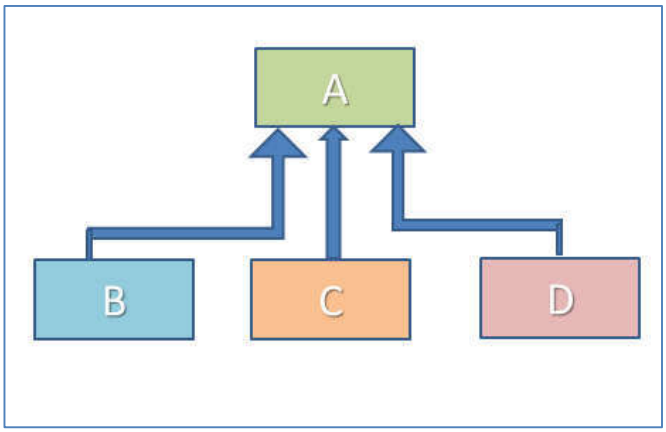
```
class TestInheritance2{
public static void main(String args[]){
bayikucing d=new bayikucing();
d.tangis();
d.meong();
d.makan();
}
}
```

Output

```
Menangis...
Mengeong...
Menyusui...
```

d. Hierarchical *Inheritance*

Mengacu pada hubungan child class dan parent class di mana lebih dari satu *class* mengextends *class* yang sama. Misalnya, *class* B, C & D meng extend *class* A yang sama.



Gambar 12. 5 Hierarchical inheritance

```
class hewan{
void makan(){System.out.println("menyusui...");}
}
class kucing extends hewan{
```

```
void meong(){System.out.println("mengeong...");}
}
class burung extends hewan{
void cicit(){System.out.println("mencicit...");}
}
```

```
class TestInheritance3{
public static void main(String args[]){
burung b=new burung ();
b.cicit();
c.makan();
//c.meong ();//C.T.Error
}
}
```

Output :

```
mencicit ...
menyusui ...
```

C. LATIHAN / TUGAS

1. Kembangkan contoh 1 program diatas dengan menambahkan attribut Tunjangan Makan, Transport pada parent class, lalu pada child class kalkulasi gaji total dengan membuat mehod nya,
   Rumus **gaji total** = Gaji + Bonus+ Tunjangan makan + Transport

   Output yang diharapkan :

```
Gaji System Analyst   : 4000000
Bonus                 : 1000000
Tunjangan Makan       :  400000
```

```
Transport :           :  500000
Gaji Total            : 5900000
```

D. REFERENSI

Horstmann Cay S., (2011). *Big Java 4t^{h} Edition ,san jose university , united state Of America*. *RRD jefferson city publishing.*

Deitel Paul , Deitel Harvey, (2012) Java how to program eighth edition, pearson education, Boston Massachusetts , *USA, publishing as prentice hall.*

Rose Cristhoper, (2017), Java Succinctly Part 2, Morrisville, NC 27560, USA, Syncfusion, Inc.

Downey Allen B. , Mayfield Chris, (2017), Think Java, Needham, Massachusetts, USA, Green Tea Press

Hayes Helen, (2021), BeginnersBook.com, https://beginnersbook.com/java-tutorial-for-beginners-with-examples/, di akses pada tanggal 21 November 2021.

Sonoo Jaiswal , (2021) JavaTpoint offers college campus training , https://www.javatpoint.com/java-tutorial, diakses pada tanggal 5Desember 2021