try this at home

```
public class Buku {
    private String pengarang;
    private String judul;

    private Buku() {
        this("The Naked Traveller", "Trinity");
    }

    private Buku(String judul,String pengarang) {
        this.judul = judul;
        this.pengarang = pengarang;
    }

    private void cetakKeLayar() {
        System.out.println("Judul : " + judul + " Pengarang : " + pengarang);
    }

    public static void main(String[] args){
        Buku a,b ;
        a = new Buku("Edensor", "Andrea Hirata");
        b = new Buku();
        a.cetakKeLayar();
        b.cetakKeLayar();
    }
}
```

Keyword this untuk memanggil constructor yang menerima dua parameter

Hasil output adalah seperti ini:

```
Judul : Edensor , Pengarang : Andrea Hirata

Judul : The Naked Traveller , Pengarang : Trinity
```

Pada contoh program di atas, kita juga dapat mengambil pelajaran bahwa penggunaan modifier private pada constructor mengakibatkan constructor tersebut hanya dapat dikenali dalam satu class saja, sehingga pembuatan objek hanya dapat dilakukan dalam lokal class tersebut.

Catatan penting lainnya : bahwa method **void** main(String[] args) haruslah bersifat **public static.**

Inner Class

Apa itu Inner Class?

Inner class disebut juga sebagai nested class, artinya adalah suatu kelas yang dideklarasikan di dalam bodan class atau interface lain.

Kelas dimana inner class didefinisikan disebut top level class yang merupakan (kelas) anggota langsung dari sebuah paket. Inner class berbeda dengan subclass, karena subclass berada pada kelas yang terpisah dengan superclassnya. Berbeda dengan kelas 'normal' yang dapat diinstansiasi secara independen, inner-class harus selalu melibatkan top level class-nya untuk melakukan instansiasi.

Adakah keuntungan menggunakan inner class?

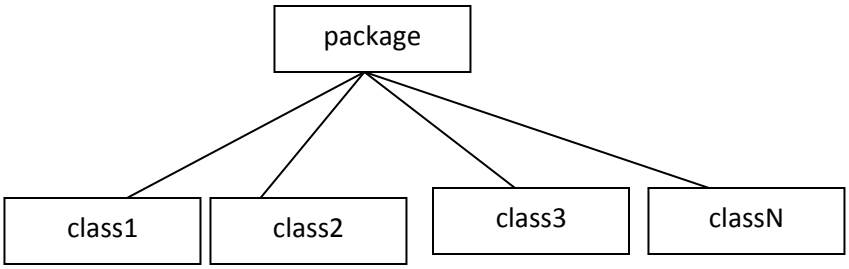
Keuntungan object-oriented

Yang paling penting dari inner class adalah bahwa dengan inner class kita bisa meng'objekkan' sesuatu yang awalnya tidak bisa dianggap sebagai objek. Hal ini memungkinkan code menjadi lebih object-oriented daripada ketika tidak menggunakan inner class. Karena setiap instansiasi inner class merupakan instansiasi dari top-level classnya juga, maka objek inner class juga memiliki akses ke setiap member dan method kelas induknya.

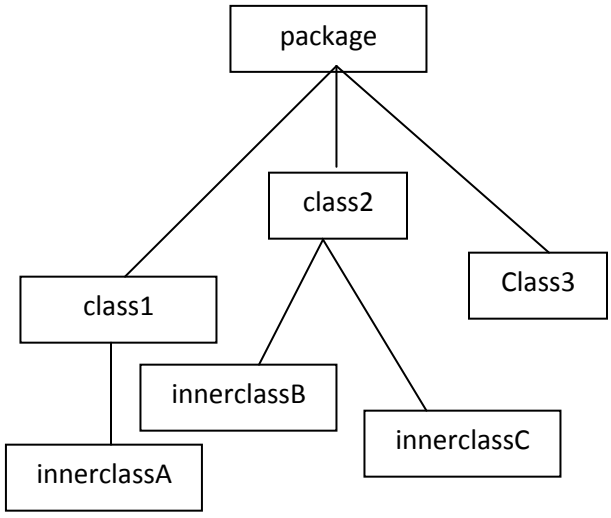
Keuntungan pengorganisasian

Dari sudut pandang pengorganisasian, inner class memungkinkan kita mengatur struktur paket. Daripada menjadikan segala sesuatu menjadi flat-package, kelas dapat disarangkan dalam kelas lain.

Hirarki tanpa inner class



Hirarki dengan inner class



References

Modul Praktikum Pemrograman Berorientasi Objek Common Labz 2008-2009.
http://java.ygrace.com
http://battleprogrammer.wordpress.com
http://www.javaworld.com
http://bernazlionk.wordpress.com
http://www.wikipedia.org
http://one.indoskripsi.com

Modul 3
Object Oriented Programming (OOP) II

Tujuan:

1. Mengerti konsep OOP lebih dalam.
2. Ngerti konsep dan bisa implementasiin inheritance, polymorphisme, interface, dan abstract class pada program.
3. Ngerti perbedaan class dengan interface, interface dengan abstract class.

Pada Modul 2, kita sudah mempelajari dasar-dasar OOP. Di modul ini kita akan mempelajari OOP lebih dalam. Mari kita mulai dari Inheritance.

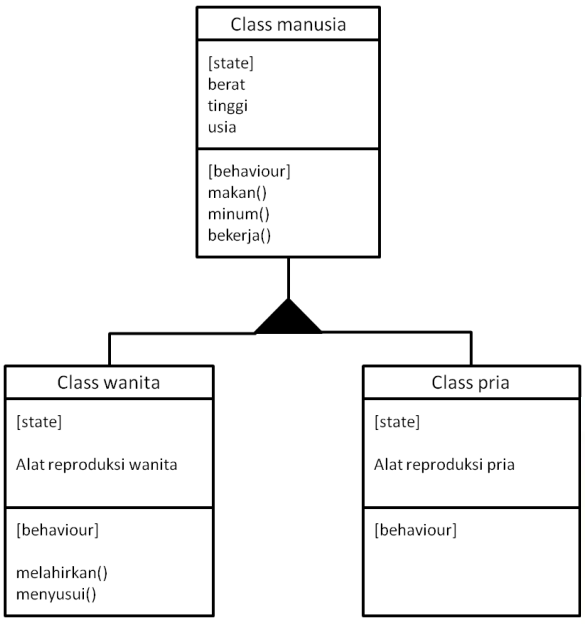
Inheritance

Apa itu inheritance?

Pewarisan merupakan proses peciptaan kelas baru dengan mewarisi karakteristik kelas yang sudah ada, ditambah dengan karakteristik unik kelas baru itu.

Sebelumnya inheritance sudah dijelaskan pada modul 2.

Contoh pewarisan pada Class manusia



Penjelasan diagram di sebelah:
Manusia adalah superclass dari pria dan wanita. Pria dan wanita mewarisi semua state dan juga behaviour yang dimiliki class manusia. Coba kita lihat perbedaannya, pada kelas manusia terdapat atribut berat, tinggi dan usia. Untuk kelas turunannya yaitu pria, terdapat atribut tambahan yang unik yaitu alat reproduksi pria. Jadi disini kelas pria mempunyai atribut total : berat, tinggi, usia dan juga alat reproduksi pria.

Petunjuk Ringkas penggunaan pewarisan:

a. Tempatkan operasi dan field yang sama di superclass
b. Jangan gunakan protected fields
c. Jangan gunakan pewarisan kecuali semua method yang diturunkan adalah berarti
d. Gunakan pewarisan untuk memodelkan hubungan "is-a"

Pendeklarasian Inheritance

```
[modifier] class NamaKelasInduk {
      // deklarasi konstanta
      // deklarasi method
}
[modifier] class NamaKelasAnak extends NamaKelasInduk{
      // deklarasi konstanta
      // deklarasi method
}
```

try this at home

```
public class PersegiPanjang{
      private int panjang;
      private int lebar;

      public void setPanjang(int p){
            panjang=p;
      }
      public void setLebar(int l){
            lebar=l;
      }
      public int getPanjang(){
            return panjang;
      }
      public int getLebar(){
            return lebar;
      }
      public int Luas(){
            int luas=panjang*lebar;
            return luas;
      }
}
```

Dibawah ini kita akan membuat kelas Balok sebagai kelas turunan dari kelas PersegiPanjang

```
public class Balok extends PersegiPanjang{
      private int tinggi;

      public void setTinggi(int t){
            tinggi=t;
      }
      public int getTinggi(){
            return tinggi;
      }
      public int Volume(){
            int v=getPanjang()*getLebar()*tinggi;
            return v;
      }
}
```

Output dari program:

```
      DEMO INHERITANCE

SuperClass ==PERSEGIPANJANG==
 Panjang: 5
 Lebar  : 5
 Luas   : 25

SubClass ==BALOK==
 Panjang: 4
 Lebar  : 3
 Tinggi : 5
 Volume : 60
```

Bagaimanakah bentuk program utama agar output yang dihasilkan seperti di samping?

Answer:

```
public class Demo{
      public static void main(String args[]){
            PersegiPanjang a= new PersegiPanjang();
            a.setPanjang(5);
            a.setLebar(5);
            System.out.println("");
            System.out.println("       DEMO INHERITANCE     ");
            System.out.println("");
            System.out.println("    SuperClass ==PERSEGIPANJANG==");
            System.out.println("      Panjang    : "+a.getPanjang());
            System.out.println("      Lebar      : "+a.getLebar());
            System.out.println("      Luas       : "+a.Luas());
            System.out.println("");

            Balok b= new Balok();
            //kelas balok tinggal memanggil method yang ada didalam kelas persegi
            b.setPanjang(4);
            b.setLebar(3);
            b.setTinggi(5);

            System.out.println("    SubClass ==BALOK==");
            System.out.println("     Panjang     : "+b.getPanjang());
            System.out.println("     Lebar             : "+b.getLebar());
            System.out.println("     Tinggi            : "+b.getTinggi());
            System.out.println("     Volume     : "+b.Volume());
      }
}
```

Overloading

Apa itu overloading?

Didalam java, kita dapat membuat dua atau lebih konstruktor/ method yang mempunyai nama sama dalam satu kelas, tetapi jumlah dan tipe argumen dari masing-masing kontrukstor atau method haruslah berbeda satu dengan yang lainnya. Hal ini yang dinamakan overloading.

```
public void setHarga(int harga){}
public void setHarga(double harga){}
public void setHarga(float harga){}
public void setHarga(float harga, String jumlah){}
```

Yang penting tipe data dalam parameter method yang dioverload itu HARUS BERBEDA!!!

try this at home

```
public class Phone{
	private String merk;
	private int harga;

	public Phone(){
	}
	public Phone(String merk){
		this.merk=merk;
	}
	public Phone(String merk, int harga){
		this.merk=merk;
		this.harga=harga;
	}
	public void isiPhone(String merk){
		this.merk=merk;
	}
	public void isiPhone(String merk, int harga){
		this.merk=merk;
		this.harga=harga;
	}
	public void lihatPhone(){
		System.out.println("    Merk   : "+merk);
		System.out.println("    Harga        : "+harga);
		System.out.println("");
	}
}
```

overloading konstruktor, disini terdapat 3 konstruktor dengan nama sama dengan parameter inputan yang berbeda

overloading method, disini terdapat dua method dengan nama sama dengan parameter inputan yang berbeda

```
public class DemoOverLoading{
    public static void main(String args[]){
       System.out.println("");
       Phone p1 = new Phone();
       Phone p2 = new Phone("Nokia");
       Phone p3 = new Phone("Sony Ericsoon",500);
       System.out.println("Perbedaan Output dari masing2 konstruktor");
       p1.lihatPhone();
       p2.lihatPhone();
       p3.lihatPhone();

       Phone p4,p5;
       p4 = new Phone();
       p5 = new Phone();
       p4.isiPhone("Samsung");
       p5.isiPhone("Samsung", 5000);
       System.out.println("Perbedaan Output dari masing2 method");
       p4.lihatPhone();
       p5.lihatPhone();
    }
}
```

Output dari program:

```
Perbedaan Output dari masing2 konstruktor
        Merk    : null
        Harga   : 0

        Merk    : Nokia
        Harga   : 0

        Merk    : Sony Ericsoon
        Harga   : 500

Perbedaan Output dari masing2 method
        Merk    : Samsung
        Harga   : 0

        Merk    : Samsung
        Harga   : 5000
```