A. Kompetensi

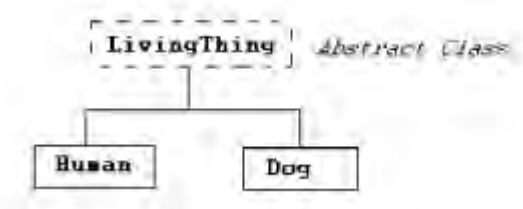
Setelah mengikuti mata kuliah ini, mahasiswa diharapkan mampu memahami prinsip polymorphism.

B. Dasar Teori

1. Abstract Class

Misalnya kita ingin membuat superclass yang mempunyai method tertentu yang berisi implementasi, dan juga beberapa method yang akan di-overridden oleh subclasses nya.

Sebagai contoh, kita akan membuat superclass bernama LivingThing. class ini mempunyai method tertentu seperti breath, eat, sleep, dan walk. Akan tetapi, ada beberapa method di dalam superclass yang sifatnya tidak dapat digeneralisasi. Kita ambil contoh, method walk. *Tidak semua kehidupan berjalan(walk) dalam cara yang sama.* Ambil manusia sebagai misal, kita manusia berjalan dengan dua kaki, dimana kehidupan lainnya seperti anjing berjalan dengan empat kaki. Akan tetapi, beberapa ciri umum dalam kehidupan sudah biasa, itulah kenapa kita inginkan membuat superclass umum dalam hal ini.



Kita dapat membuat superclass yang mempunyai beberapa method dengan implementasi sedangkan yang lain tidak. Class jenis ini yang disebut dengan class abstract.

Sebuah **class abstract** adalah class yang tidak dapat di-instantiate. Seringkali muncul di atas hirarki class pemrograman berbasis object, dan mendefinisikan keseluruhan aksi yang mungkin pada object dari seluruh subclasses dalam class.

Method ini dalam class abstract yang tidak mempunyai implementasi dinamakan method abstract. Untuk membuat method abstract, tinggal menulis deklarasi method tanpa tubuh class dan digunakan menggunakan kata kunci abstract. Contohnya,

```
public abstract void someMethod();
```

Sekarang mari membuat contoh class abstract.

```
public abstract class LivingThing
{
      public void breath(){
            System.out.println("Living Thing breathing...");
      }

      public void eat(){
            System.out.println("Living Thing eating...");
```

```
        }

        /**
        * abstract method walk
        * Kita ingin method ini di-overridden oleh subclasses
        */
        public abstract void walk();
}
```

Ketika class meng-*extend* class abstract LivingThing, dibutuhkan untuk override method abstract walk(), atau lainnya, juga subclass akan menjadi class abstract, oleh karena itu tidak dapat di-instantiate. Contohnya,

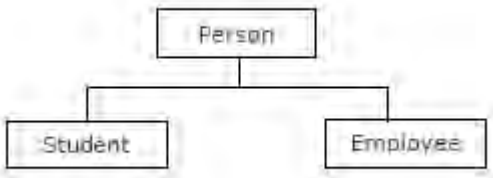
```
public class Human extends LivingThing
{
        public void walk(){
                System.out.println("Human walks...");
        }
}
```

Jika class Human tidak dapat overridde method walk, kita akan menemui pesan error berikut ini,

```
Human.java:1: Human is not abstract and does not override
abstract method walk() in LivingThing
public class Human extends LivingThing
       ^
1 error
```

2. Polimorfisme

Class induk Person memiliki dua buah subclass yaitu Student dan Employee. Di bawah ini adalah hierarkinya,



Dalam Java, kita dapat membuat referensi yang merupakan tipe dari superclass ke sebuah object dari subclass tersebut. Sebagai contohnya,

```
public static main( String[] args )
{
        Person ref;

        Student studentObject = new Student();
        Employee employeeObject = new Employee();
```

```
        ref = studentObject; //Person menunjuk kepada
                             // object Student

        //beberapa kode di sini
}
```

Sekarang dimisalkan kita punya method getName dalam superclass Person kita, dan kita override method ini dalam kedua subclasses Student dan Employee,

```
public class Person
{
        public String getName(){
                System.out.println("Person Name:" + name);
                return name;
        }
}

public class Student extends Person
{
        public String getName(){
                System.out.println("Student Name:" + name);
                return name;
        }
}

public class Employee extends Person
{
        public String getName(){
                System.out.println("Employee Name:" + name);
                return name;
        }
}
```

Kembali ke method utama kita, ketika kita mencoba memanggil method getName dari reference Person ref, method getName dari object Student akan dipanggil. Sekarang, jika kita berikan ref ke object Employee, method getName dari Employee akan dipanggil.

```
public static main( String[] args )
{
        Person ref;

        Student studentObject = new Student();
        Employee employeeObject = new Employee();

        ref = studentObject; //Person menunjuk kepada
                             // object Student

        String temp = ref.getName(); //getName dari Student
                                     //class dipanggil
```

```
        System.out.println( temp );

        ref = employeeObject; //Person menunjuk kepada
                              // object Employee

        String temp = ref.getName(); //getName dari Employee
        /                            /class dipanggil
        System.out.println( temp );
}
```

Kemampuan dari reference untuk mengubah sifat menurut object apa yang dijadikan acuan dinamakan polimorfisme. Polimorfisme menyediakan multiobject dari subclasses yang berbeda untuk diperlakukan sebagai object dari superclass tunggal, secara otomatis menunjuk method yang tepat untuk menggunakannya ke *particular* object berdasar subclass yang termasuk di dalamnya.
Contoh lain yang menunjukkan properti polimorfisme adalah ketika kita mencoba melalui reference ke method. Misalkan kita punya method static **printInformation** yang mengakibatkan object Person sebagai reference, kita dapat me-reference dari tipe Employee dan tipe Student ke method ini selama itu masih subclass dari class Person.

```
public static main( String[] args )
{
      Student studentObject = new Student();
      Employee employeeObject = new Employee();

      printInformation( studentObject );
      printInformation( employeeObject );
}

public static printInformation( Person p ){
. . . .
}
```

C. Alat/ Bahan

1. Lab. Sheet Pemrograman Java 5
2. PC / Laptop with OS installed
3. JDK 1.5 or latest
4. J-Creator or text editor

D. Langkah Kerja

1. Baca dan pahami dasar teori di atas.
2. Lakukan kompilasi dan eksekusi terhadap contoh-contoh source code atau program yang ada di dasar teori dan LAMPIRAN.
3. Kerjakan tugas individu di bawah.

E. Tugas Individu

Buatlah class **abstract** `Geometri` yang memiliki dua buah abstract method yaitu `hitungKeliling()` dan `hitungLuas()`. Buatlah subclass-subclass dari class Geometri sebagai berikut:

a. Class `SegiEmpat`
b. Class `SegiTiga`
c. Class `Lingkaran`

Masing-masing subclass memliki method tambahan untuk mengakses, mengubah dan menampilkan atribut yang dimilikinya.

Kemudian buatlah sebuah class untuk menguji class-class yang dibuat di atas. Gunakanlah **polimorfisme** dalam class penguji ini.

F. Lampiran

- **CH 51 & CH 52** (Introduction to Computer Science using Java, Java 5.0 version, January 2006, Bradley Kjell, Central Connecticut State University http://chortle.ccsu.edu/CS151/cs151java.html)

- **Java™ How to Program, Sixth Edition**, H. M. Deitel - Deitel & Associates, Inc., P. J. Deitel - Deitel & Associates, Inc