Konsep Object Oriented

*Object Oriented Programming (OOP) adalah teknik untuk membuat program berorientasi objek.
*Sebuah objek digambarkan oleh entitas dalam dunia nyata yang dapat diidentifikasi secara jelas. Contoh: siswa, mobil, meja dan hal-hal lain yang dapat dilihat sebagai objek.
*Objek dalam bahasa pemrograman harus memiliki behaviour / method, property, type, dan identity.
*Semua data dan fungsi dibungkus dalam kelas dan objek.
*OOP menyediakan fasilitas untuk menulis sebuah program untuk mengubah objek yang sebenarnya dalam kode pemrograman komponen.

Access Modifier

Sering dikenal dengan visibilitas atau pengubah aksesibilitas
Untuk melindungi anggota kelas (data dan metode) yang ada di kelas.
Private: menyembunyikan anggota kelas sehingga tidak dapat diakses dari luar kelas
Protected: memungkinkan anggota kelas dapat diakses oleh kelas atau turunannya di kelas beberapa paket serupa.
Public: memungkinkan kelas anggota dapat diakses oleh kelas kelas lain
Package: digunakan untuk mengatur kelas. Jika kelas yang dibuat tanpa

Array List

```
import java.util.ArrayList;

public class ArrayListDemo {

  public static void main(String[] args) {

    ArrayList al = new ArrayList();
    System.out.println("Inisialisasi ukuran ArrayList :  " + al.size());

    //Menambah element ke dalam array list
    al.add("C");
    al.add("A");
    al.add("E");
    al.add("B");
    al.add("D");
    al.add("F");
    al.add(1,"A2");//Memasukan objek A2 dalam array pada indek ke 1

    System.out.println("Ukuran setelah penambahan " + al.size());

    //Menampilkan Array List
    System.out.println("Isi Array List: " +  al );

    //Menghapus element dalam array list
    al.remove("F");
    al.remove(2);

    System.out.println("Ukuran setelah penghapusan : " + al.size());
    System.out.println("Isi Array List:" + al);

  }

}
```

```
C:\WINDOWS\system32\cmd.exe
Inisialisasi ukuran ArrayList :  0
Ukuran setelah penambahan 7
Isi Array List: [C, A2, A, E, B, D, F]
Ukuran setelah penghapusan : 5
Isi Array List:[C, A2, E, B, D]
Press any key to continue . . .
```

Vector Code

```
import java.util.Vector;

public class VectorDemo {

  public static void main(String[] args) {

    Vector v = new Vector();
    System.out.println("Inisialisasi ukuran Vector :  " + v.size());

    //Menambah element ke dalam vector
    v.add("C");
    v.add("A");
    v.add("E");
    v.add("B");
    v.add("D");
    v.add("F");
    v.add(1,"A2");//Memasukan objek A2 dalam array pada indek ke 1

    System.out.println("Ukuran setelah penambahan " + v.size());

    //Menampilkan vector
    System.out.println("Isi Vector: " +  v + "\n");

    //lihat isi vector
    System.out.println("Isi Vector pertama:" +  v.firstElement() );
    System.out.println("Isi Vector ke 2:  " +  v.elementAt(4) );
    System.out.println("Isi Vector terakhir: " +  v.lastElement() );

    //Menghapus element dalam Vector
    v.remove("F");
    v.remove(2);

    System.out.println("\nUkuran setelah penghapusan : " + v.size());
    System.out.println("Isi Vector:" + v);

  }

}
```

```
C:\WINDOWS\system32\cmd.exe
Inisialisasi ukuran Vector :  0
Ukuran setelah penambahan 7
Isi Vector: [C, A2, A, E, B, D, F]

Isi Vector pertama:C
Isi Vector ke 2:  B
Isi Vector terakhir: F

Ukuran setelah penghapusan : 5
Isi Vector:[C, A2, E, B, D]
Press any key to continue . . .
```

Perbedaan Abstract Class vs Interface

Abstract Class	Interface
Bisa berisi abstract dan non-abstract method.	Hanya boleh berisi abstract method.
Kita harus menuliskan sendiri modifiernya.	Kita tidak perlu susah2 menulis public abstract di depan nama method. Karena secara implisit, modifier untuk method di interface adalah **public** dan ***abstract***.
Bisa mendeklarasikan **constant** dan **instance variable**.	Hanya bisa mendeklarasikan **constant**. Secara implisit variable yang dideklarasikan di interface bersifat **public**, **static** dan **final**.
Method boleh bersifat **static**.	Method tidak boleh bersifat **static**.
Method boleh bersifat **final**.	Method tidak boleh bersifat **final**.
Suatu abstact class hanya bisa meng-*extend* satu abstract class lainnya.	Suatu interface bisa meng-*extend* satu atau lebih interface lainnya.
Suatu abstract class hanya bisa meng-*extend* satu abstract class dan meng-**implement** beberapa interface.	Suatu interface hanya bisa meng-*extend* interface lainnya. Dan tidak bisa meng-*implement* class atau interface lainnya.

Polymorphism

```
public class PolymorphismDemo{
    public static void main(String [] args) {
        Rectangle c = new Rectangle();
        System.out.println(c.computeArea());
    }
}

abstract class Shape {
    protected int x, y;
    public abstract double computeArea( );
}
class Rectangle extends Shape {
    double width =12, height=20;
    public double computeArea( ) {
        return width * height;
    }
}
class Circle extends Shape {
    double radius =20;
    public double computeArea( ) {
        return Math.PI * radius * radius;
    }
}
```

Wrapper Class Constants

- Example of use :

```
public class Nilai
{
    public static void main(String [] args)
    {
        System.out.println("Nilai Max Integer = " + Integer.MAX_VALUE );
        System.out.println("Nilai Min Positive Float = " + Float.MIN_VALUE );
        System.out.println("Nilai Max Double Floating-point = " + Double.MAX_VALUE );
    }
}
```

- Output :

```
Nilai Max Integer = 2147483647
Nilai Min Positive Float = 1.4E-45
Nilai Max Double Floating-point = 1.7976931348623157E308
```

Inheritance

Parent Class (*superclass*) : Bicycle.java

```
public class Bicycle {
   public int gear;
   public int speed;

   // constructor from class Bicycle with passing parameter
   public Bicycle( int startSpeed, int startGear ) {
      gear = startGear;
      speed = startSpeed;
   }

   public void setGear ( int newValue ) {
      gear = newValue;
   }

   public void applyBrake ( int decrement ) {
      speed -= decrement;
   }

   public void speedUp ( int increment ) {
      speed += increment;
   }
}
```

Pada UTS ini, mahasiswa akan dapat:

LO 1: Jelaskan konsep Object Oriented
LO 2: Memecahkan masalah algoritma menggunakan konsep Object Oriented
LO 3: Membangun aplikasi dengan Konsep Berorientasi Objek