A. Object

Pada dunia perangkat lunak, sebuah obyek adalah sebuah komponen perangkat lunak yang stukturnya mirip dengan obyek pada dunia nyata. Disekeliling kita ada banyak contoh dari objek, seperti meja, pena, kursi, televisi dan masih banyak lagi.

Yang dimiliki oleh objek antar lain

- state

  Pada implementasinya berupa field dari kelas

- behaviour

  Implementasinya menjadi method dari kelas

B. Class

Class adalah rancangan dari sebuah objek yang mendefinisikan attribut dan method umum pada semua objek dari jenis tertentu. Contohnya sepedaA, sepedaB, sepedaC adalah instance dari dari kelas sepeda. Pada java, ada sebuah kelas yang memiliki method Main, method main inilah yang nantinya akan dijalankan untuk pertama kalinya ketika sebuah program java dijalankan. Sebuah kelas hanya dapat memiliki 1 kelas induk dan dapat mengimplementasikan 1 atau lebih interface.

Dalam penulisan sebuah kelas di java, ada beberapa aturan yang biasanya dilakukan

- Nama kelas tidak menggunakan tanda spasi, dash (-) dan tanda lain.
- Nama kelas untuk 1 suku kata diawali huruf besar untuk huruf pertama, untuk lebih dari 1 suku kata berikan huruf besar pada awal tiap suku kata

  contoh : Sepeda, SepedaRodaDua, AsistenPraktikum

  Latihan

  1. Buatlah sebuah file kelas java dengan nama file Sepeda.java

     ```
     public class Sepeda{
     }
     ```

  1. Buatlah sebuah kelas java dengan nama file Utama.java

```
public class Utama{
        public static void main(String[] args){
                Sepeda sepedaA=new Sepeda();
                Sepeda sepedaB=new Sepeda();
                Sepeda sepedaC=new Sepeda();
        }
}
```

1. Letakkan file Sepeda.java dan Utama.java pada 1 direktori
2. Silahkan anda buka command prompt atau console, kemudian anda ketikkan perintah berikut:

   javac Sepeda.java

   javac Utama.java

   atau

   javac *.java
3. Jalankan kode program anda dengan perintah

   java Utama

C. Constructor

Adalah sebuah method atau fungsi yang diekseskusi ketika sebuah kelas diinisialisasi, secara default sebuah Java Class memiliki 1 buah konstruktor tanpa parameter, konstruktor ini bisa ditulis, bisa tidak ditulis. Setiap kelas dalam java bisa memiliki lebih dari 1 konstruktor dengan parameter yang berbeda – beda.

Nama dari sebuah konstruktor harus sama dengan nama dari kelas dan tidak memiliki nilai kembalian. Apabila sebuah kelas memiliki 1 buah konstruktor dengan 1 buah parameter atau lebih, maka kelas tersebut menjadi tidak memiliki lagi konstruktor default.

```
public class Sepeda{
        public Sepeda(){
        }
}
```

atau

Sebuah konstruktor juga dapat memiliki parameter

```
public class Sepeda{
        public Sepeda(String warna){
        }
}
```

Latihan

1. Silahkan anda tambahkan 2 buah konstruktor pada kelas Sepeda.java

   ```
   public class Sepeda{
           public Sepeda(){
           }
           public Sepeda(String warna){
           }
   }
   ```

2. Kompile dengan perintah javac Sepeda.java
3. Jalankan kode program anda dengan perintah

   java Utama, anda tidak akan menemui error, coba anda ubah kelas Sepeda.java menjadi berikut lalu anda kompile

   ```
   public class Sepeda{
           public Sepeda(String warna){
           }
   }
   ```

4. Jalankan kode program anda dengan perintah

   java Utama, maka akan munucl pesan error

5. Ubah kelas Utama.java menjadi seperti berikut ini

   ```
   public class Utama{
           public static void main(String[ ] args){
                   Sepeda sepedaA=new Sepeda("Merah");
                   Sepeda sepedaB=new Sepeda("Kuning");
   ```

Sepeda sepedaC=new Sepeda(“Hijau”);

}

}

1. Silahkan anda kompile Sepeda.java dan Utama.java, kemudian anda jalankan dengan perintah java Utama

D. Attribute

Digunakan untuk menyimpan informasi yang dimiliki oleh sebuah kelas.

Contoh :

private String nama;

private String warna;

private int panjang;

Latihan

1. Tambahkan 1 buah field pada kelas Sepeda.java

public class Sepeda{

private String warna;

public Sepeda(String warna){

this.warna=warna;

}

}

2. Silahkan anda kompile Sepeda.java, kemudian anda jalankan java Utama

Pada contoh kode program diatas, ada keyword **this** yang berarti menunjuk pada field warna pada kelas Sepeda, bukan parameter warna pada konstruktor **Sepeda(String warna)**

E. Access Modifier

Mendefinisikan bagaimana kelas lain menggunakan attribut atau memanggil method dari suatu kelas. Tabel berikut ini menjelaskan hak akses yang diberikan

Modifier	Class	Package	SubClass	World
public	Y	Y	Y	Y
protected	Y	Y	Y	N

default (tidak ditulis)	Y	Y	N	N
private	Y	N	N	N

keterangan

- public

  bisa diakses dari kelas bersangkutan, kelas yang berada dalam satu package, kelas turunannya dan bisa diakses dari bukan ketiganya

- protected

  hanya dapat diakses dari kelas bersangkutan, kelas yang berada dalam satu package dan kelas turunannya.

- default

  hanya dapat diakses dari kelas bersangkutan dan kelas yang berada dalam satu package

- private

  hanya dapat diakses dari kelas bersangkutan

F. Method

Dalam pemrograman yang tidak berorientasi objek dikenal dengan nama fungsi dan prosedur. Dalam java, fungsi digantikan dengan method yang memiliki tipe kembalian selain void dan prosedur digantikan dengan method yang memiliki tipe kembalian void.

Latihan

1. Tambahkan 2 buah method untuk mendapatkan warna dan mengubah warna

```
public class Sepeda{
        private String warna;
        public Sepeda(String warna){
                this.warna=warna
        }
        public String getWarna(){
                return warna;
        }
        public void setWarna(String warna){
```

```
            this.warna=warna;
        }
    }
```

2. Silahkan anda compile kelas Sepeda.java
3. Tambahkan kode program untuk mengetahui informasi warna dari sepeda, di kelas Utama.java

```
public class Utama{
    public static void main(String[ ] args){
        Sepeda sepedaA=new Sepeda("Merah");
        System.out.println("Warna sepeda A " + sepedaA.getWarna());
        Sepeda sepedaB=new Sepeda("Kuning");
        System.out.println("Warna sepeda B " + sepedaB.getWarna());
        Sepeda sepedaC=new Sepeda("Hijau");
        System.out.println("Warna sepeda C " + sepedaC.getWarna());
    }
}
```

4. Silahkan anda kompile Utama.java kemudian anda jalankan dengan perintah java Utama

G. extends

Merupakan sebuah keyword untuk menandakan bahwa sebuah kelas merupakan turunan dari kelas yang lain.

public class X extends Y memiliki keterangan bahwa kelas X adalah turunan Y.

Latihan

1. Silahkan anda buat kelas kendaraan

```
public class Kendaraan {
    private int jumlahRoda;
    public Kendaraan(int jumlahRoda){
        this.jumlahRoda=jumlahRoda;
    }
    public void setJumlahRoda(int jumlahRoda){
        this.jumlahRoda=jumlahRoda;
```

```
    }
    public int getJumlahRoda(){
        return jumlahRoda;
    }
}
```

1. Silahkan anda ubah kelas Sepeda yang merupakan turunan dari kelas Kendaraan

```
public class Sepeda extendss Kendaraan{
    private String warna;
    public Sepeda(int jumlahRoda,String warna){
        super(jumlahRoda);
        this.warna=warna;
    }
    public String getWarna(){
        return warna;
    }
    public void setWarna(String warna){
        this.warna=warna;
    }
}
```

1. Silahkan anda ubah kelas Utama, dengan kode program sebagai berikut

```
public class Utama{
    public static void main(String[ ] args){
        Sepeda sepedaA=new Sepeda(2,"Merah");
        System.out.println("Warna sepeda A " + sepedaA.getWarna());
        System.out.println("Roda sepeda A " + sepedaA.getJumlahRoda());
        Sepeda sepedaB=new Sepeda(3,"Kuning");
        System.out.println("Warna sepeda B " + sepedaB.getWarna());
        System.out.println("Roda sepeda B " + sepedaB.getJumlahRoda());
        Sepeda sepedaC=new Sepeda(1,"Hijau");
        System.out.println("Warna sepeda C " + sepedaC.getWarna());
        System.out.println("Roda sepeda C " + sepedaC.getJumlahRoda());
    }
```

```
}
```

2. Kompile Kendaraan.java, Sepeda.java dan Utama.java
3. Jalankan java Utama

H. Super

Super merupakan sebuah keyword yang digunakan untuk memanggil method atau konstruktor yang terdapat pada kelas induk dari sebuah kelas turunan. Pada contoh diatas, terdapat kode program ***super(jumlahRoda);*** *Kode program ini maksudnya memanggil konstruktor dari kelas induk yang memiliki 1 parameter bertipe integer.*

I. Override

Override atau diartikan menulis ulang memiliki pengertian bahwa kita melakukan proses pendefinisian ulang method yang ada di kelas induk atau method pada suatu interface dengan method yang ada di kelas turunannya atau kelas yang mengimplementasikannya.

Ada beberapa cara untuk melakukan override

1. Override melalui interface

   Buatlah sebuah interface bernama IBarang.java

```
public interface IBarang{
        void setMerk(String merk);
        String getMerk();
}
```

   Implementasikan IBarang di kelas Kendaraan.

```
public class Kendaraan implements IBarang{
        private int jumlahRoda;
        private String merk;
        public Kendarran(int jumlahRoda){
                this.jumlahRoda=jumlahRoda;
        }
        public int getJumlahRoda(){
                return this.jumlahRoda;
        }
```

```
    public void setJumlahRoda(int jumlahRoda){
        this.jumlahRoda=jumlahRoda;
    }
    @Override
    public String getMerk(){
        return merk;
    }
    @Override
    public void setMerk(String merk){
        this.merk=merk;
    }
}
```

2. Override melalui instance

Silahkan anda ubah kelas Utama.java menjadi seperti kode program berikut ini

```
public class Utama{
    public static void main(String[ ] args){
        Sepeda sepedaA=new Sepeda(2,"Merah");
        System.out.println("Warna sepeda A " + sepedaA.getWarna());
        System.out.println("Roda sepeda A " + sepedaA.getJumlahRoda());
        Sepeda sepedaB=new Sepeda(3,"Kuning");
        System.out.println("Warna sepeda B " + sepedaB.getWarna());
        System.out.println("Roda sepeda B " + sepedaB.getJumlahRoda());
        Sepeda sepedaC=new Sepeda(1,"Hijau"){
            public int getJumlahRoda(){
                return super.getJumlahRoda()*2;
            }
            public String getMerk(){
                return "Belum ada merk";
            }
        };
```

```
        System.out.println("Warna sepeda C " + sepedaC.getWarna());
        System.out.println("Roda sepeda C " + sepedaC.getJumlahRoda());
    }
}
```

3. Override melalui extends

   Berikan juga perubahan pada kelas Sepeda.java

```
public class Sepeda extends Kendaraan{

    private String warna;
    public Sepeda(int jumlahRoda,String warna){
        super(jumlahRoda);
        this.warna=warna;
    }
    public String getWarna(){
        return warna;
    }
    public void setWarna(String warna){
        this.warna=warna;
    }
    @Override
    public int getJumlahRoda(){
        return super.getJumlahRoda()*2;
    }
    @Override
    public String getMerk(){
        return "Belum ada merk";
    }
}
```

J. Overloaded

Overloaded diartikan sebagai suatu method atau konstruktor yang memiliki nama sama dalam satu kelas tetapi memiliki parameter yang berbeda.

```
public class Sepeda extends Kendaraan{
```

```
    private String warna;
    public Sepeda(String warna){
        super(2);
        this.warna=warna;
    }
    public Sepeda(int jumlahRoda,String warna){
        super(jumlahRoda);
        this.warna=warna; //bisa juga ditulis this(warna);
    }
    public String getWarna(){
        return warna;
    }

    public void setWarna(String warna){
        this.warna=warna;
    }
    public void setAttribut(String warna){
        this.warna=warna;
    }
    public void setAttribut(int jumlahRoda,String warna){
        this.warna=warna;//bisa juga ditulis setAttribut(warna);
        setJumlahRoda(jumlahRoda);
    }
    @Override
    public int getJumlahRoda(){
        super.getJumlahRoda()*2;
    }
    @Override
    public String getMerk(){
        return "Belum ada merk";
    }
}
```

K. Interface

Merupakan sebuah keyword untuk mendefinisikan sekumpulan method dan konstanta. Interface bisa digunakan apabila sudah dilakukan implementasi pada sebuah kelas non abstract. Interface dapat digunakan untuk

1. Mewakili suatu tingkat laku yang bisa dimiliki oleh banyak kelas tanpa memaksakan relasi antar kelas tersebut.

```
public class Lemari implements IBarang{
        private int panjang, lebar, tinggi;
        private String merk;
        public Lemari(int panjang, int lebar, int tinggi){
                this.panjang=panjang;
                this.lebar=lebar;
                this.tinggi=tinggi;
        }
        public String getMerk(){
                return merk;
        }
        public void setMerk(String merk){
                this.merk=merk;
        }
        public int getPanjang(){
                return panjang;
        }
        public int getLebar(){
                return lebar;
        }
        public int getTinggi(){
                return tinggi;
        }
}
```

2. Menangani Event atau Action yang terjadi pada suatu kelas.

   Buatlah sebuah interface untuk menangani aksi ketika ada perubahan warna

```
public interface SepedaEvent{
        void onWarnaChange(String warnaLama,String warnaBaru);
}
tambahkan pada kelas sepeda anda
public class Sepeda extends Kendaraan{
        private String warna;
        private SepedaEvent sepedaEvent;
        public Sepeda(String warna){
                super(2);
                this.warna=warna;
        }
        public Sepeda(int jumlahRoda,String warna){
                super(jumlahRoda);
                this.warna=warna; //bisa juga ditulis this(warna);
        }
        public String getWarna(){
                return warna;
        }
        public void setWarna(String warna){
                if(sepedaEvent!=null){
                        sepedaEvent.onWarnaChange(this.warna,warna);
                }
                this.warna=warna;
        }
        public void setAttribut(String warna){
                if(sepedaEvent!=null){
                        sepedaEvent.onWarnaChange(this.warna,warna);
                }
                this.warna=warna;
                //bisa juga ditulis
                /*
                setWarna(warna);
                */
```

```
    }
    public void setAttribut(int jumlahRoda,String warna){
        if(sepedaEvent!=null){
            sepedaEvent.onWarnaChange(this.warna,warna);
        }
        this.warna=warna;
        setJumlahRoda(jumlahRoda);
        //bisa juga ditulis - I
        /*
        setWarna(warna)
        setJumlahRoda(jumlahRoda);
        */
        //bisa juga ditulis – II
        /*
        setAttribut(warna);
        setJumlahRoda(jumlahRoda);
        */
    }
    @Override
    public int getJumlahRoda(){
        super.getJumlahRoda()*2;
    }
    @Override
    public String getMerk(){
        return "Belum ada merk";
    }
    public void setSepedaEvent(SepedaEvent sepedaEvent){
        this.sepedaEvent=sepedaEvent;
    }
}
```

kemudian buatlah sebuah kelas dengan nama Utama2.java

```
public class Utama2 implements SepedaEvent{
    public Utama2(){
```

```
        Sepeda sepedaA=new Sepeda(2,"Biru");
        sepedaA.setSepedaEvent(this);
        System.out.println("Warna " + sepedaA.getWarna());
        System.out.println("Roda " + sepedaA.getJumlahRoda());
        sepedaA.setWarna("Hijau");
        System.out.println("Warna " + sepedaA.getWarna());
        System.out.println("Roda " + sepedaA.getJumlahRoda());
    }
    public void onWarnaChange(String warnaLama,String warnaBaru){
        System.out.println("Warna berubah dari " + warnaLama +
            " menjadi " + warnaBaru);
    }
    public static void main(String[ ] args){
        Utama2 utm2=new Utama2();
    }
}
```

3. Memudahkan penanganan objek dalam parameter.

   Silahkan anda buat kelas PembacaMerk.java

```
public class PembacaMerk{
    public static void sebutkanMerk(IBarang iBarang){
        System.out.println(iBarang.getMerk());
    }
}
```

   Silahkan anda buat Utama3.java kemudian anda compile

```
public class Utama3{
    public static void main(String[ ] args){
        Sepeda sepedaA=new Sepeda(2,"Hijau");
        sepedaA.setMerk("Belum ada merk");
        Lemari lemariA=new Lemari(4,3,2);
        lemariA.setMerk("Lemari kecil");
        System.out.print("Merk dari SepedaA adalah ");
        PembacaMerk.sebutkanMerk(sepedaA);
```

```
            System.out.print("Merk dari LemariA adalah ");
            PembacaMerk.sebutkanMerk(lemariA);
        }
    }
```

pada contoh diatas, Method sebutkanMerk yang terletak di kelas PembacaMerk menggunakan kata kunci **static** yang berarti method tersebut dapat dipanggil tanpa perlu membuat instance terlebih dahulu dari kelas PembacaMerk.

L. Abstract Class

Kelas ini memiliki 1 buah method abstract, untuk menggunakan kelas ini adalah dengan mewariskan kelas ini baru kemudian membuat instance dari turunannya. Apabila anda hendak memaksakan untuk membuat instance dari kelas ini, maka anda diwajibkan melakukan override melalui instance.

Perbedaan antara abstract class dengan interface adalah pada abstract class kita dapat membuat method abstract maupun non abstract, attribut, konstanta.

Buatlah sebuah kelas Meja yang merupakan abstract class dan mengimplementasikan IBarang.

```
public abstract class Meja implements IBarang{
    private int panjang, lebar, tinggi;
    public Meja(int panjang, int lebar, int tinggi){
        this.panjang=panjang;
        this.lebar=lebar;
        this.tinggi=tinggi;
    }
    public int getTinggi(){
        return tinggi;
    }
    public int getLebar(){
        return lebar;
    }
    public int getPanjang(){
        return panjang;
    }
}
```

Sekarang buat turunan dari kelas Meja menjadi kelas yang lebih spesifik yaitu MejaTulis.

```
public class MejaTulis extends Meja{
        public MejaTulis(int panjang,int lebar, int tinggi){
                super(panjang,lebar,tinggi);
        }
        public String getMerk(){
                return "Olympic";
        }
        public void setMerk(String merk){
        }
}
```

Selanjutnya kita buat sebuah kelas untuk menguji objek meja tersebut, buat kelas dengan nama Utama4.java

```
public class Utama4{
        public static void main(String[ ] args){
                MejaTulis mejaTulis=new MejaTulis(4,5,6);
                Meja meja=new Meja(4,5,6){
                        public String getMerk(){
                                return "Tidak punya merk";
                        }
                        public void setMerk(String merk){
                        }
                };
        }
}
```

M.Final

Diterapkan pada class dan attribut, class yang bersifat final tidak dapat diwariskan ke kelas lain, sedangkan attribut yang bersifat final tidak dapat diubah nilainya.

Contoh final:

1. Pada Attribut

   public final int BERAT=10;

2. Pada Class

public final class Batu{

}

N. Package

Digunakan untuk melakukan pengaturan letak dari kelas atau mengelompokkkan sekumpulan class dan interface yang berhubungan dalam satu kelompok.

package latihan

public class Barang ….....

Untuk setiap package, misalnya package latihan, maka dibuatkan 1 buah folder dengan nama latihan, apabila kelas Barang terletak pada package latihan maka file Barang.java juga terletak pada folder latihan.

O. Instance Of

Digunakan untuk melakukan pengecekan apakah sebuah objek adalah instance dari suatu kelas;

```
public class Utama4{
    public static void main(String[ ] args){
        MejaTulis mejaTulis=new MejaTulis(4,5,6);
        Meja meja=new Meja(4,5,6){
            public String getMerk(){
                return "Tidak punya merk";
            }
            public void setMerk(String merk){
            }
        };
        boolean cek=meja instanceOf Meja;
        System.out.println("meja instanceof Meja ? " + cek);

    }
}
```