# CLASS DAN OBJECT

Praktikum Pemrograman Berorientasi Objek

Cahaya Dewi

Claudia Alves

Daniel Gallego

# CLASS

# Class

Class adalah blueprint atau template yang digunakan untuk membuat objek. Class mendefinisikan struktur dan perilaku yang akan dimiliki oleh objek-objek yang dibuat dari class tersebut.

# Class

- Atribut/Field: Variabel yang menyimpan data
- Method: Fungsi yang mendefinisikan perilaku atau aksi
- Constructor: Method khusus untuk menginisialisasi objek

# Contoh Sederhana Class

```java
public class Mobil {
    // Atribut
    private String merek;
    private String warna;
    private int tahun;

    // Method
    public void nyalakanMesin() {
        System.out.println("Mesin dinyalakan");
    }

    public void matikanMesin() {
        System.out.println("Mesin dimatikan");
    }
}
```

Cahaya Dewi Claudia Alves Daniel Gallego

# Constructor

# Constructor

Constructor adalah method khusus yang dipanggil secara otomatis ketika objek dibuat. Constructor digunakan untuk menginisialisasi nilai awal pada atribut objek.

# Constructor

- Nama constructor sama dengan nama class
- Tidak memiliki return type (bahkan void)
- Dipanggil otomatis saat objek dibuat dengan kata kunci **new**
- Jika tidak didefinisikan, Java akan membuat default constructor

# Contoh Constructor

```
public class Mobil {
    private String merek;
    private String warna;

    // Default constructor
    public Mobil() {
        this.merek = "Unknown";
        this.warna = "Unknown";
    }
}
```

# Contoh Constructor

```
public class Mobil {
    private String merek;
    private String warna;
    private int tahun;

    // Constructor dengan parameter
    public Mobil(String merek, String warna, int tahun) {
        this.merek = merek;
        this.warna = warna;
        this.tahun = tahun;
    }
}
```

Cahaya Dewi Daniel Gallego

# Object Instantiation

# Object Instantiation

Object adalah instance dari class.
Object memiliki state (nilai atribut) dan behavior (method yang dapat dipanggil).

Instantiation adalah proses membuat objek dari class menggunakan kata kunci **new**.

# Object Instantiation

```
NamaClass namaObjek = new NamaClass();
```

# Contoh Object Instantiation

```
Mobil mobil1 = new Mobil();
Mobil mobil2 = new Mobil();
```

Cahaya Dewi Claudia Alves Daniel Gallego

# Overloading

# Overloading

Overloading adalah kemampuan untuk memiliki **lebih dari satu** method atau constructor dengan nama yang sama dalam class yang sama, tetapi dengan **parameter yang berbeda** (jumlah parameter, tipe parameter, atau urutan parameter).

# Contoh Overloading

```java
public class Mobil {
    private String merek;
    private String warna;
    private int tahun;
    private double harga;

    // Constructor 1: Default
    public Mobil() {
        this.merek = "Unknown";
        this.warna = "Putih";
        this.tahun = 2023;
        this.harga = 0.0;
    }

    // Constructor 2: Dengan merek saja
    public Mobil(String merek) {
        this.merek = merek;
        this.warna = "Putih";
        this.tahun = 2023;
        this.harga = 0.0;
    }
}
```

# Contoh Overloading

```java
public class Kalkulator {
    // Method dengan 2 parameter
    public int tambah(int a, int b) {
        return a + b;
    }

    // Method dengan 3 parameter
    public int tambah(int a, int b, int c) {
        return a + b + c;
    }

    // Method dengan 4 parameter
    public int tambah(int a, int b, int c, int d) {
        return a + b + c + d;
    }
}
```

# Contoh Overloading

```java
public class Kalkulator {
    // Method untuk integer
    public int tambah(int a, int b) {
        System.out.println("Menambah integer");
        return a + b;
    }

    // Method untuk double
    public double tambah(double a, double b) {
        System.out.println("Menambah double");
        return a + b;
    }

    // Method untuk string (concatenation)
    public String tambah(String a, String b) {
        System.out.println("Menggabung string");
        return a + b;
    }
}
```

# Aturan Overloading

- Signature method harus berbeda (parameter berbeda)
- Tidak boleh ada ambiguitas dalam pemanggilan
- Return type tidak mempengaruhi overloading
- Access modifier dapat berbeda

# Contoh Overloading

```java
// Constructor overloading
Mobil mobil1 = new Mobil();
Mobil mobil2 = new Mobil("Toyota");

// Method overloading
Kalkulator calc = new Kalkulator();
System.out.println(calc.tambah(5, 3));
System.out.println(calc.tambah(5.5, 3.2));
System.out.println(calc.tambah("Hello", "World"));
```

# Abstact Class

# Abstact Class

Abstract class adalah class yang **tidak dapat diinstantiasi** secara langsung. Class ini dirancang untuk menjadi parent class (superclass) yang menyediakan struktur dasar untuk subclass-nya.

# Abstact Class

- Dideklarasikan dengan keyword abstract
- Tidak dapat dibuat objeknya secara langsung
- Dapat memiliki method abstract dan concrete
- Dapat memiliki constructor
- Dapat memiliki instance variables

# Abstact Method

Method abstract adalah method yang dideklarasikan tanpa implementasi (tanpa body). Subclass wajib mengimplementasikan method abstract dari parent class.

# Contoh Abstract Class

```
abstract class Hewan {
    String nama;

    Hewan(String nama) {
        this.nama = nama;
    }

    // method abstract (wajib diimplementasikan oleh subclass)
    abstract void suara();

    // method biasa
    void info() {
        System.out.println("Nama hewan: " + nama);
    }
}
```

# TERIMA KASIH

# Tugas

- Buat sebuah class yang berisi construtor dan ada method overloading serta constructor overloading, kemudian buatlah 3 object yang berbeda dari class tersebut.
- Tidak boleh menggunakan contoh yang sama pada praktikum