Laporan Praktikum
Mata Kuliah Pemrograman Berorientasi Objek

Pertemuan 5. Praktikum 1
"Polymorphism"

Dosen Pengampu : Willdan Aprizal Arifin, S.Pd., M.Kom.

Disusun Oleh :
Vera Anggraini
(2307909)

PROGRAM STUDI SISTEM INFORMASI KELAUTAN
UNIVERSITAS PENDIDIKAN INDONESIA
2024

I. Pendahuluan

Polimorfisme adalah salah satu konsep dasar dalam JavaScript yang berarti objek atau fungsi bisa memiliki banyak bentuk. Jadi, ada satu fungsi atau metode yang bisa dipakai untuk beragam objek atau kelas objek yang berbeda dalam cara penggunaannya. Ada dua jenis polimorfisme yang biasa ditemukan di bahasa pemrograman, termasuk JavaScript:

1. Polimorfisme lewat Fungsi: Ini terjadi ketika satu fungsi bisa digunakan untuk berbagai tipe data. Misalnya, kita bisa punya metode yang menerima objek atau string, tetapi perilakunya akan berbeda tergantung jenis informasi yang diberikan.

2. Polimorfisme lewat Pewarisan: Dalam Pemrograman Berorientasi Objek, ini terjadi ketika kelas anak mengubah metode yang ada di kelas induk. Dengan cara ini, kita bisa memanggil metode yang sama pada objek dari kelas yang berbeda, dan setiap objek akan memberikan hasil sesuai dengan cara mereka masing-masing.

II. Penjelasan

```
class Kapal {
  constructor(nama, jenis, panjang, lebar) {
    this.nama = nama;
    this.jenis = jenis;
    this.panjang = panjang;
    this.lebar = lebar;
  }

  infoKapal() {
    return `Kapal ${this.nama} merupakan jenis ${this.jenis} yang berukuran ${this.panjang}m x ${this.lebar}m.`;
  }
}
```

Program tersebut membuat kelas `kapal` yang bisa digunakan untuk membuat objek kapal dengan properti nama, jenis, panjang, dan lebar. Program ini juga punya fungsi `infokapal()` yang menampilkan informasi kapal dalam bentuk kalimat.

```
class KapalAngkut extends Kapal {
  constructor(nama, jenis, panjang, lebar, kapasitasAngkut) {
    super(nama, jenis, panjang, lebar);
    this.kapasitasAngkut = kapasitasAngkut;
  }

  infoKapal() {
    return `${super.infoKapal()} Kapal ini memiliki kapasitas angkut sebesar ${
      this.kapasitasAngkut
    } ton.`;
  }
}
```

Kode di atas membuat kelas `KapalAngkut` yang merupakan turunan dari kelas `Kapal`. Kelas ini menambahkan properti baru yaitu `kapasitasAngkut`, yang menunjukkan berapa banyak muatan kapal bisa bawa. Pada bagian konstruktor, `super()` digunakan untuk memanggil data dari kelas `Kapal`, lalu menambahkan kapasitas angkutnya. Fungsi `infoKapal()`

menggabungkan info dari kelas `Kapal` dan menambahkan keterangan tentang kapasitas muatannya dalam ton.

```
class KapalKargo extends KapalAngkut {
  constructor(nama, panjang, lebar, kapasitasAngkut) {
    super(nama, "Kargo", panjang, lebar, kapasitasAngkut);
  }
}

class KapalTanker extends KapalAngkut {
  constructor(nama, panjang, lebar, kapasitasAngkut) {
    super(nama, "Tanker", panjang, lebar, kapasitasAngkut);
  }
}
```

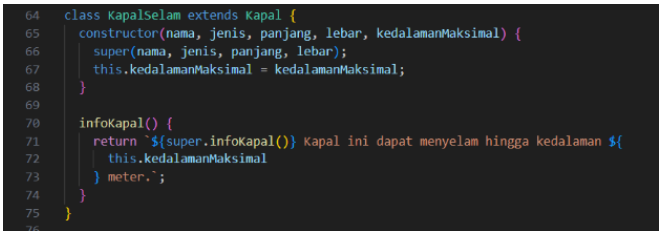
program di atas membuat subclass dengan nama kapalKargo yang merupakan turunan dari KapalAngkut. Juga membuat subclass dengan nama kapalTanker yang juga merupakan turunan dari kapalAngkut.

```
class KapalCepat extends Kapal {
  constructor(nama, jenis, panjang, lebar, kecepatanMaksimal) {
    super(nama, jenis, panjang, lebar);
    this.kecepatanMaksimal = kecepatanMaksimal;
  }

  infoKapal() {
    return `${super.infoKapal()} Kapal ini memiliki kecepatan maksimal ${
      this.kecepatanMaksimal
    } knot.`;
  }
}
```

Kode di atas membuat kelas `KapalCepat` yang merupakan turunan dari kelas `Kapal`. Kelas ini menambahkan properti baru yaitu `kecepatanMaksimal`, yang menunjukkan berapa maksimal kecepatan suatu kapal. Pada bagian konstruktor, `super()` digunakan untuk memanggil data dari kelas `Kapal`, lalu menambahkan kecepatan maksimalnya. Fungsi `infoKapal()` menggabungkan info dari kelas `Kapal` dan menambahkan keterangan tentang kecepatan kapal dalam satuan knot.

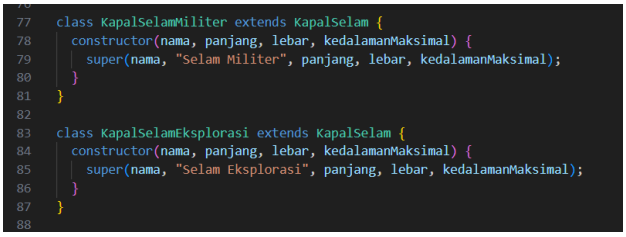
```
class KapalPatroli extends KapalCepat {
  constructor(nama, panjang, lebar, kecepatanMaksimal) {
    super(nama, "Patroli", panjang, lebar, kecepatanMaksimal);
  }
}

class KapalPenumpangCepat extends KapalCepat {
  constructor(nama, panjang, lebar, kecepatanMaksimal) {
    super(nama, "Penumpang Cepat", panjang, lebar, kecepatanMaksimal);
  }
}
```

program di atas membuat subclass dengan nama kapalPatroli yang merupakan turunan dari KapalCepat. Juga membuat subclass dengan nama kapalPenumpangCepat yang juga merupakan turunan dari kapalCepat.



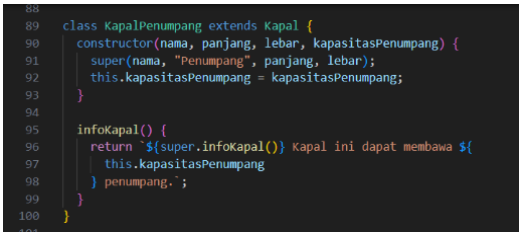
```
class KapalSelam extends Kapal {
  constructor(nama, jenis, panjang, lebar, kedalamanMaksimal) {
    super(nama, jenis, panjang, lebar);
    this.kedalamanMaksimal = kedalamanMaksimal;
  }

  infoKapal() {
    return `${super.infoKapal()} Kapal ini dapat menyelam hingga kedalaman ${
      this.kedalamanMaksimal
    } meter.`;
  }
}
```

Kode di atas membuat kelas `KapalSelam` yang merupakan turunan dari kelas `Kapal`. Kelas ini menambahkan properti baru yaitu `kedalamanMaksimal`, yang menunjukkan berapa maksimal kedalaman yang bisa dicapai oleh suatu kapal. Pada bagian konstruktor, `super()` digunakan untuk memanggil data dari kelas `Kapal`, lalu menambahkan kedalaman maksimalnya. Fungsi `infoKapal()` menggabungkan info dari kelas `Kapal` dan menambahkan keterangan tentang kedalaman maksimal kapal dalam satuan meter.



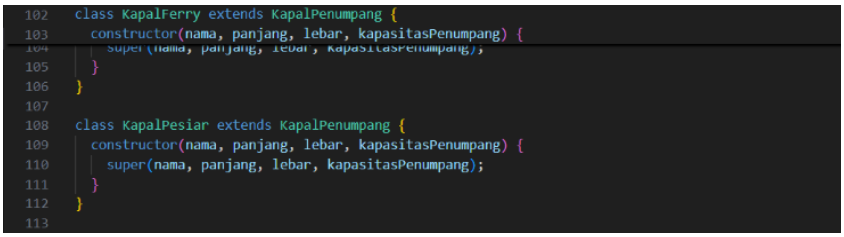
```
class KapalSelamMiliter extends KapalSelam {
  constructor(nama, panjang, lebar, kedalamanMaksimal) {
    super(nama, "Selam Militer", panjang, lebar, kedalamanMaksimal);
  }
}

class KapalSelamEksplorasi extends KapalSelam {
  constructor(nama, panjang, lebar, kedalamanMaksimal) {
    super(nama, "Selam Eksplorasi", panjang, lebar, kedalamanMaksimal);
  }
}
```

program di atas membuat subclass dengan nama kapalSelamMiliter yang merupakan turunan dari KapalSelam. Juga membuat subclass dengan nama kapalSelamExplorasi yang juga merupakan turunan dari kapalSelam.



```
class KapalPenumpang extends Kapal {
  constructor(nama, panjang, lebar, kapasitasPenumpang) {
    super(nama, "Penumpang", panjang, lebar);
    this.kapasitasPenumpang = kapasitasPenumpang;
  }

  infoKapal() {
    return `${super.infoKapal()} Kapal ini dapat membawa ${
      this.kapasitasPenumpang
    } penumpang.`;
  }
}
```

Kode di atas membuat kelas `KapalPenumpang` yang merupakan turunan dari kelas `Kapal`. Kelas ini menambahkan properti baru yaitu `kapasitasPenumpang`, yang menunjukkan berapa kapasitas penumpang dari kapal penumpang. Pada bagian konstruktor, `super()` digunakan untuk memanggil data dari kelas `Kapal`, lalu menambahkan kapasitas penumpangnya. Fungsi `infoKapal()` menggabungkan info dari kelas `Kapal` dan menambahkan keterangan tentang kapasitas maksimal kapal penumpang.



```
class KapalFerry extends KapalPenumpang {
  constructor(nama, panjang, lebar, kapasitasPenumpang) {
    super(nama, panjang, lebar, kapasitasPenumpang);
  }
}

class KapalPesiar extends KapalPenumpang {
  constructor(nama, panjang, lebar, kapasitasPenumpang) {
    super(nama, panjang, lebar, kapasitasPenumpang);
  }
}
```

program di atas membuat subclass dengan nama kapalFerry yang merupakan turunan dari KapalPenumpang. Juga membuat subclass dengan nama kapalPesiar yang juga merupakan turunan dari kapalPenumpang.

```
function cetakInfoKapal(kapal) {
  console.log(kapal.infoKapal());
}
```

Program di atas mencetak informasi kapal menggunakan methode infokapal dan mencetak infokapal sesuai dengan jenis kapal.

```
const kapalKargo = new KapalKargo("Laju Nusantara", 300, 80, 1500);
const kapalTanker = new KapalTanker("Oil Master", 400, 60, 200000);
const kapalPatroli = new KapalPatroli("Sea Hawk", 50, 15, 60);
const kapalPenumpangCepat = new KapalPenumpangCepat("Fast Ferry", 100, 20, 45);
const kapalSelamMiliter = new KapalSelamMiliter("Poseidon", 150, 30, 500);
const kapalSelamEksplorasi = new KapalSelamEksplorasi("Explorer", 100, 20, 300);
const kapalFerry = new KapalFerry("Budiyono Siregar", 200, 100, 600);
const kapalPesiar = new KapalPesiar("Royal Caribbean", 350, 120, 3000);

cetakInfoKapal(kapalKargo);
cetakInfoKapal(kapalTanker);
cetakInfoKapal(kapalPatroli);
cetakInfoKapal(kapalPenumpangCepat);
cetakInfoKapal(kapalSelamMiliter);
cetakInfoKapal(kapalSelamEksplorasi);
cetakInfoKapal(kapalFerry);
cetakInfoKapal(kapalPesiar);
```

Pada program di atas membuat instance dari masing masing jenis kapal dan memanggil method infokapal dari masing masing objeknya.