Laporan Praktikum

Mata Kuliah PEMROGRAMAN BERORIENTASI OBJEK

Pertemuan 5

"Tugas Polymorphism"

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

Disusun Oleh :
Dini Dwi Andini
2308802

PROGRAM STUDI SISTEM INFORMASI KELAUTAN
UNIVERSITAS PENDIDIKAN INDONESIA
2024

I. PENDAHULUAN

Pada mata kuliah Pemrograman Berorientasi Objek (PBO) pertemuan ke-5 ini, kita akan membahas konsep Polimorfisme melalui studi kasus objek kapal. Polimorfisme atau Polymorphism adalah kemampuan objek untuk mengambil berbagai bentuk dan menyediakan implementasi berbeda untuk antarmuka yang sama. Dalam praktikum ini, kita akan mengimplementasikan polimorfisme pada berbagai jenis kapal seperti kapal penumpang, kapal kargo, dan kapal pesiar. Tujuan dari praktikum ini adalah untuk memahami penerapan polimorfisme dalam PBO, sehingga dapat membuat kode yang lebih efektif dan mudah dipahami.

II. ALAT DAN BAHAN

- Laptop
- Mouse(Optional)
- Bahasa pemrograman JS, HTML, CSS

III. LANGKAH KERJA

1. Membuat folder Tugas Polymorphism
2. Membuat file index.html, ini kodenya :

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h1>Konsep Polymorphism</h1>
    <script src="Polymorphism.js"></script>
    <link rel="stylesheet" href="style.css">

</body>
</html>
```

3. HTML yang digunakan hanya berfungsi sebagai kerangka untuk menampilkan judul, yaitu "Konsep Polymorphism", dan untuk menyisipkan file JavaScript eksternal bernama Polymorphism.js yang mengandung implementasi konsep polymorphism dalam kode JavaScript.
4. Membuat file polymorphism.js berikut ini kodenya:

```
// Class Utama
class Kapal {
    constructor(nama, panjang, lebar) {
        this.nama = nama;
        this.panjang = panjang;
```

```
        this.lebar = lebar;
    }

    berlayar() {
        return `${this.nama} sedang berlayar.`;
    }

    berhenti() {
        return `${this.nama} sedang berhenti di pelabuhan.`;
    }
}

// Subclass 1: Kapal Penumpang
class KapalPenumpang extends Kapal {
    constructor(nama, panjang, lebar, kapasitasPenumpang) {
        super(nama, panjang, lebar);
        this.kapasitasPenumpang = kapasitasPenumpang;
    }

    berlayar() {
        return `Kapal penumpang ${this.nama} sedang berlayar dengan
kapasitas ${this.kapasitasPenumpang} penumpang.`;
    }

    berhenti() {
        return `Kapal penumpang ${this.nama} sedang berhenti untuk
menurunkan penumpang.`;
    }
}

// Subclass 2: Kapal Kargo
class KapalKargo extends Kapal {
    constructor(nama, panjang, lebar, muatan) {
        super(nama, panjang, lebar);
        this.muatan = muatan;
    }

    berlayar() {
        return `Kapal kargo ${this.nama} sedang berlayar membawa muatan
sebesar ${this.muatan} ton.`;
    }

    berhenti() {
        return `Kapal kargo ${this.nama} sedang berhenti untuk membongkar
muatan.`;
    }
```

```
}

// Subclass 3: Kapal Ikan
class KapalIkan extends Kapal {
    constructor(nama, panjang, lebar, jumlahJaring) {
        super(nama, panjang, lebar);
        this.jumlahJaring = jumlahJaring;
    }

    berlayar() {
        return `Kapal ikan ${this.nama} sedang berlayar dengan ${this.jumlahJaring} jaring untuk menangkap ikan.`;
    }

    berhenti() {
        return `Kapal ikan ${this.nama} sedang berhenti di pelabuhan untuk memuat hasil tangkapan.`;
    }
}

// Subclass 4: Kapal Perang
class KapalPerang extends Kapal {
    constructor(nama, panjang, lebar, jenisSenjata) {
        super(nama, panjang, lebar);
        this.jenisSenjata = jenisSenjata;
    }

    berlayar() {
        return `Kapal perang ${this.nama} sedang berlayar dengan persenjataan ${this.jenisSenjata}.`;
    }

    berhenti() {
        return `Kapal perang ${this.nama} sedang berhenti untuk melakukan perawatan dan persiapan senjata.`;
    }
}

// Menggunakan Polymorphism
const kapal1 = new KapalPenumpang("Oasis of the Seas", 300, 50, 3000);
const kapal2 = new KapalKargo("CargoX", 200, 40, 5000);
const kapal3 = new KapalIkan("Nelayan", 100, 25, 10);
const kapal4 = new KapalPerang("Destroyer", 150, 30, "Meriam 120mm");

function aktivitasKapal(kapal) {
    console.log(kapal.berlayar());
```

```
    console.log(kapal.berhenti());
}

// Memanggil metode menggunakan polymorphism
aktivitasKapal(kapal1);
aktivitasKapal(kapal2);
aktivitasKapal(kapal3);
aktivitasKapal(kapal4);

app.listen(3000,() => {
    console.log("Server berjalan di port 3000, buka web melalui http://localhost:3000")
})
```

5. Kelas Kapal didefinisikan sebagai kelas dasar yang memiliki properti nama, panjang, dan lebar, serta metode berlayar() dan berhenti().
6. Metode ini mengembalikan string yang menggambarkan aktivitas dasar kapal.
7. Beberapa subclass (turunan) dibuat dari kelas Kapal:
   - KapalPenumpang: Menambah properti kapasitasPenumpang dan mengubah perilaku metode berlayar() dan berhenti() untuk menggambarkan aktivitas kapal penumpang.
   - KapalKargo: Menambah properti muatan dan menyesuaikan metode untuk menggambarkan kapal kargo.
   - KapalIkan: Menambah properti jumlahJaring dan mengubah perilaku metode untuk aktivitas kapal ikan.
   - KapalPerang: Menambah properti jenisSenjata dan menggambarkan aktivitas kapal perang melalui metode yang diubah.
8. Fungsi aktifvitasKapal ini menerima objek kapal sebagai parameter dan memanggil metode berlayar() serta berhenti(). Karena polymorphism, meskipun metode yang dipanggil sama, perilakunya akan sesuai dengan jenis objek (subclass) yang diproses.
9. Objek-objek dari masing-masing subclass (KapalPenumpang, KapalKargo, KapalIkan, dan KapalPerang) dibuat, dan fungsi aktivitasKapal dipanggil dengan masing-masing objek. Hasilnya, meskipun fungsi yang dipanggil sama, perilakunya berbeda sesuai dengan jenis kapal.
10. Membuat file style.css kode nya seperti berikut :

```
/* Mengatur gaya umum untuk seluruh halaman */
body {
    font-family: Arial, sans-serif;
    background-color: #f4f4f9;
    margin: 0;
    padding: 0;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
```

```
    color: #333;
}

/* Mengatur gaya untuk heading utama */
h1 {
    text-align: center;
    color: #0056b3;
    margin-bottom: 20px;
}

/* Container untuk konten utama */
.container {
    background-color: #ffffff;
    padding: 30px;
    border-radius: 8px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
    width: 60%;
    max-width: 800px;
    text-align: center;
}

/* Gaya untuk tombol (jika ada di JavaScript) */
button {
    background-color: #007bff;
    color: #fff;
    border: none;
    padding: 10px 20px;
    border-radius: 5px;
    cursor: pointer;
    font-size: 16px;
    transition: background-color 0.3s ease;
}

button:hover {
    background-color: #0056b3;
}
```

11. Body: Mengatur font, warna latar belakang, margin, dan padding agar tampilannya lebih rapi dan terpusat.
12. h1: Mengatur heading utama dengan warna biru dan menyesuaikan margin bawah.
13. Container: Mengatur tata letak dan gaya dari konten utama, memberikan latar belakang putih, padding, border radius, dan shadow untuk efek visual.
14. Button: Menambahkan gaya untuk tombol interaktif dengan efek hover.

IV. KESIMPULAN

Konsep polymorphism memungkinkan metode yang sama (berlayar dan berhenti) memiliki perilaku berbeda berdasarkan jenis objek yang digunakan, sehingga mempermudah penanganan berbagai objek dengan antarmuka yang seragam dan mengurangi ketergantungan pada implementasi spesifik setiap kelas.