

Laporan Praktikum
Mata Kuliah Pemrograman Berorientasi Objek



Tugas 5
“Polymorphism”

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

Disusun Oleh :
Alia Selvi Solekhah
(2301121)

PROGRAM STUDI SISTEM INFORMASI KELAUTAN UNIVERSITAS
PENDIDIKAN INDONESIA
2024

I. PENDAHULUAN

Pemrograman Berorientasi Objek (Object-Oriented Programming atau OOP) merupakan paradigma pemrograman yang berfokus pada penggunaan objek dan kelas untuk merancang dan mengimplementasikan solusi dari suatu masalah. Salah satu konsep penting dalam OOP adalah *polymorphism*, yang memungkinkan suatu fungsi atau metode dapat berperilaku berbeda tergantung pada objek yang memanggilmnya. Dalam konteks praktek pemrograman web, konsep ini sangat bermanfaat untuk meningkatkan fleksibilitas dan skalabilitas aplikasi, karena dapat membantu mengelola variasi perilaku antar objek yang berbeda.

Pada praktek ini, kita mengimplementasikan konsep *polymorphism* melalui sebuah studi kasus mengenai kelas kapal. Dalam studi ini, kami membuat kelas dasar Kapal yang berisi informasi umum tentang kapal seperti nama, jenis, tahun pembuatan, dan kapasitas. Kemudian, beberapa kelas turunan seperti KapalKargo, KapalPenumpang, KapalPerang, dan KapalLautDalam dibuat untuk mengimplementasikan perilaku khusus dari masing-masing jenis kapal. Setiap kelas turunan memiliki atribut dan metode spesifik yang menggambarkan karakteristik kapal tersebut, namun tetap menggunakan metode umum dari kelas dasar Kapal melalui konsep *polymorphism*. Implementasi ini ditujukan untuk menunjukkan bagaimana sebuah metode umum dapat diadaptasi untuk berbagai jenis objek tanpa mengubah struktur utama program.

II. ALAT DAN BAHAN

- Komputer atau Laptop
- Editor kode (misalnya: Visual Studio Code)
- Browser untuk menjalankan JavaScript

III. HASIL DAN PEMBAHASAN

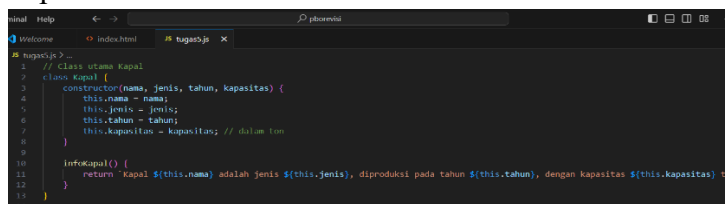
PENJELASAN KODE PROGRAM 1. Kelas Kapal {superclass}

Kelas ini adalah kelas utama yang digunakan untuk membuat objek kapal secara umum.

Atribut yang ada di dalamnya adalah:

- Nama: menyimpan nama kapal
- Jenis: menyimpan merk kapal
- Tahun: menyimpan tahun produksi kapal
- Kapasitas: menyimpan jenis kapal (kargo, penumpang, perang, laut dalam)

Selain itu, ada metode infoKapal(), yang bertugas menampilkan informasi lengkap tentang kapal tersebut.



```
1 // Class utama Kapal
2 class Kapal {
3   constructor(nama, jenis, tahun, kapasitas) {
4     this.nama = nama;
5     this.jenis = jenis;
6     this.tahun = tahun;
7     this.kapasitas = kapasitas; // dalam ton
8   }
9
10  infoKapal() {
11    return `Kapal ${this.nama} adalah jenis ${this.jenis}, diproduksi pada tahun ${this.tahun}, dengan kapasitas ${this.kapasitas} ton`;
12  }
13 }
```

2. Kelas Turunan (subclass)

Class ini bertindak sebagai superclass yang berisi atribut dasar dari sebuah kapal, seperti nama, jenis, tahun, dan kapasitas. Terdapat metode infoKapal() yang mengembalikan informasi dasar tentang kapal, seperti nama kapal, jenis, tahun produksi, dan kapasitas.

- **KapalKargo** merupakan subclass dari Kapal yang menambahkan atribut jenisMuatan untuk menyimpan informasi tentang jenis muatan yang dibawa oleh kapal kargo. Metode infoKapal() di-override untuk menambahkan informasi mengenai jenis muatan yang dibawa oleh kapal.

```
14
15 // Subclass KapalKargo
16 class KapalKargo extends Kapal {
17     constructor(nama, jenis, tahun, kapasitas, jenisMuatan) {
18         super(nama, jenis, tahun, kapasitas);
19         this.jenisMuatan = jenisMuatan;
20     }
21
22     infoKapal() {
23         return `${super.infoKapal()} kapal ini membawa muatan berupa ${this.jenisMuatan}.`;
24     }
25 }
26
```

- **KapalPenumpang** juga merupakan subclass dari Kapal, dengan tambahan atribut jumlahPenumpang untuk menyimpan informasi kapasitas penumpang kapal. Metode infoKapal() di-override untuk menampilkan jumlah penumpang yang dapat ditampung oleh kapal.

```
27 // Subclass KapalPenumpang
28 class KapalPenumpang extends Kapal {
29     constructor(nama, jenis, tahun, kapasitas, jumlahPenumpang) {
30         super(nama, jenis, tahun, kapasitas);
31         this.jumlahPenumpang = jumlahPenumpang;
32     }
33
34     infoKapal() {
35         return `${super.infoKapal()} kapal ini dapat menampung ${this.jumlahPenumpang} penumpang.`;
36     }
37 }
38
```

- **KapalPerang** Subclass ini memiliki atribut tambahan persenjataan, yang menggambarkan jenis senjata yang digunakan oleh kapal perang. Metode infoKapal() menambahkan deskripsi mengenai persenjataan kapal.

```
39 // Subclass KapalPerang
40 class KapalPerang extends Kapal {
41     constructor(nama, jenis, tahun, kapasitas, persenjataan) {
42         super(nama, jenis, tahun, kapasitas);
43         this.persenjataan = persenjataan;
44     }
45
46     infoKapal() {
47         return `${super.infoKapal()} kapal ini dipersenjatai dengan ${this.persenjataan}.`;
48     }
49 }
50
```

- **KapalLautDalam**: Kelas ini menambahkan atribut kedalamanMaks, yang menyimpan informasi tentang kemampuan kapal dalam menyelam hingga kedalaman maksimum tertentu. Metode infoKapal() menambahkan informasi mengenai kemampuan kapal menyelam.

```
51 // Subclass KapalLautDalam
52 class KapalLautDalam extends Kapal {
53     constructor(nama, jenis, tahun, kapasitas, kedalamanMaks) {
54         super(nama, jenis, tahun, kapasitas);
55         this.kedalamanMaks = kedalamanMaks;
56     }
57
58     infoKapal() {
59         return `${super.infoKapal()} kapal ini dapat menyelam hingga kedalaman maksimum ${this.kedalamanMaks} meter.`;
60     }
61 }
62
```

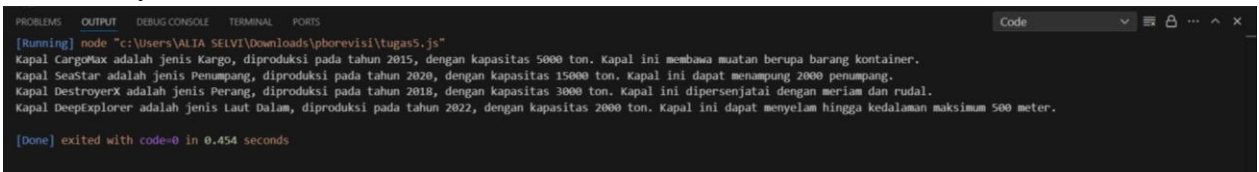
3. Penerapan Polimorfisme

Pada bagian ini, beberapa objek dibuat menggunakan kelas KapalKargo, KapalPenumpang, KapalPerang, dan KapalLautDalam. Setiap objek disimpan dalam sebuah array kapalList, dan kemudian metode infoKapal() dipanggil secara polimorfik untuk setiap objek dalam array. Konsep polimorfisme memungkinkan pemanggilan metode infoKapal() pada setiap objek meskipun objek tersebut berasal dari subclass yang berbeda. Metode ini akan memberikan output yang berbeda berdasarkan jenis objeknya.

```
62 // Polimorfisme
63 const kapalkargo = new KapalKargo("CargoMax", "Kargo", 2015, 5000, "barang kontainer");
64 const kapalpenumpang = new KapalPenumpang("SeaStar", "Penumpang", 2020, 15000, 2000);
65 const kapalperang = new KapalPerang("DestroyerX", "Perang", 2018, 3000, "meriam dan rudal");
66 const kapallautdalam = new KapalLautDalam("DeepExplorer", "Laut Dalam", 2022, 2000, 500);
67
```

4. Hasil Output

Hasil output akan menampilkan informasi lengkap tentang setiap kapal sesuai dengan tipe subclass-nya.



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
[Running] node "c:\Users\ALIA SELVI\Downloads\pborevis\tugas5.js"
Kapal CargoMax adalah jenis Kargo, diproduksi pada tahun 2015, dengan kapasitas 5000 ton. Kapal ini membawa muatan berupa barang kontainer.
Kapal SeaStar adalah jenis Penumpang, diproduksi pada tahun 2020, dengan kapasitas 15000 ton. Kapal ini dapat menampung 2000 penumpang.
Kapal DestroyerX adalah jenis Perang, diproduksi pada tahun 2018, dengan kapasitas 3000 ton. Kapal ini dipersenjatai dengan meriam dan rudal.
Kapal DeepExplorer adalah jenis Laut Dalam, diproduksi pada tahun 2022, dengan kapasitas 2000 ton. Kapal ini dapat menyelam hingga kedalaman maksimum 500 meter.
[Done] exited with code=0 in 0.454 seconds
```

Pembahasan

Dengan menggunakan konsep *inheritance* (pewarisan) dan *polymorphism* (polimorfisme), kode ini untuk mengelola berbagai jenis kapal dengan perilaku dan properti yang berbeda, namun tetap menggunakan metode yang sama (infoKapal)

IV. KESIMPULAN

Melalui praktek pemrograman web ini, kami berhasil mengimplementasikan konsep *polymorphism* pada sebuah sistem yang terdiri dari berbagai jenis kapal. Dengan menggunakan kelas dasar Kapal dan beberapa kelas turunan seperti KapalKargo, KapalPenumpang, KapalPerang, dan KapalLautDalam, kami menunjukkan bagaimana metode yang sama dapat berfungsi secara berbeda berdasarkan jenis objek yang memanggilnya. Konsep *polymorphism* tidak hanya memudahkan pengelolaan kode yang lebih rapi dan terstruktur, tetapi juga memperluas fleksibilitas aplikasi dalam menangani berbagai variasi objek tanpa harus menulis ulang kode yang sama. Penerapan konsep ini juga sangat relevan dalam pengembangan aplikasi web modern, di mana skalabilitas dan kemudahan pemeliharaan kode sangat penting.