Laporan Praktikum

Mata Kuliah Pemrograman Berorientasi Objek (PBO)

Pertemuan 5. Praktikum 5

“Polymorphism”

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

Disusun Oleh:
Putri Wahyuni
2310271

PROGRAM STUDI SISTEM INFORMASI KELAUTAN
UNIVERSITAS PENDIDIKAN INDONESIA
2024

I. PENDAHULUAN

Polimorfisme (Polymorphism) dalam pemrograman berorientasi objek (OOP) adalah konsep di mana objek yang berbeda tipe (subclass) dapat diperlakukan atau dipanggil dengan cara yang sama melalui interface yang sama (method yang sama), namun menghasilkan perilaku yang berbeda-beda sesuai dengan tipe objeknya. Polimorfisme memungkinkan sebuah method atau fungsi memproses objek dari kelas yang berbeda secara seragam, tetapi menghasilkan keluaran atau respons yang spesifik tergantung pada objek yang diproses.

II. ALAT DAN BAHAN

- Laptop
- Aplikasi Visual Studio Code
- Google Chrome

III. PENJELASAN

Berikut adalah penjelasan mengenai kodingannya. Kode ini menunjukkan implementasi polimorfisme dengan menggunakan class dan subclass dalam JavaScript. Berikut langkah-langkah yang terjadi:

1. Kelas `Kapal`

Kelas `Kapal` adalah kelas dasar (superclass) yang memiliki beberapa atribut dan metode dasar yang umum untuk semua jenis kapal.
- Constructor: Dalam konstruktor, kita mendefinisikan empat atribut utama: `nama`, `jenis`, `panjang`, dan `lebar`. Selain itu, atribut `_status` diinisialisasi dengan nilai "Berlabuh".
- Metode `berlayar()`: Metode ini mengubah status kapal menjadi "Berlayar" dan mengembalikan pesan yang menyatakan bahwa kapal telah berlayar.
- Metode `berhenti()`: Metode ini mengubah status kapal kembali menjadi "Berlabuh" dan mengembalikan pesan bahwa kapal telah berlabuh.
- Metode `infoKapal()`: Metode ini mengembalikan string yang berisi informasi lengkap tentang kapal, termasuk nama, jenis, ukuran, dan status saat ini.

2. Subkelas `KapalPenumpang`
Kelas `KapalPenumpang` adalah turunan dari kelas `Kapal`, yang menambahkan fungsionalitas khusus untuk kapal penumpang.
- Constructor: Menggunakan `super()` untuk memanggil konstruktor dari kelas induk (`Kapal`) dan menambahkan atribut `kapasitasPenumpang`.
- Metode `naikPenumpang(jumlahPenumpang)`: Memeriksa apakah jumlah penumpang melebihi kapasitas maksimum. Jika ya, mengembalikan pesan bahwa kapasitas terlampaui. Jika tidak, status kapal diubah menjadi "Penuh".
- Metode `turunPenumpang(jumlahPenumpang)`: Mengubah status kapal menjadi "Kosong" dan mengembalikan pesan bahwa penumpang telah turun.

- Metode `aksi()`: Mengembalikan pesan yang menyatakan kegiatan kapal terkait penumpang.
- Metode `infoKapal()`: Memanggil metode `infoKapal()` dari kelas induk dan menambahkan informasi tentang kapasitas penumpang.

3. Subkelas `KapalKargo`
Kelas `KapalKargo` adalah turunan dari kelas `Kapal` yang khusus untuk kapal pengangkut barang.
- Constructor: Memanggil konstruktor dari kelas induk dan menambahkan atribut `kapasitasMuatan`.
- Metode `aksi()`: Mengembalikan pesan yang menyatakan bahwa kapal sedang memuat atau membongkar kargo.
- Metode `infoKapal()`: Memanggil metode `infoKapal()` dari kelas induk dan menambahkan informasi tentang kapasitas muatan.

4. Subkelas `KapalTanker`
Kelas `KapalTanker` adalah turunan dari kelas `Kapal` untuk kapal yang mengangkut minyak.
- Constructor: Menambahkan atribut `kapasitasMinyak`.
- Metode `aksi()`: Mengembalikan pesan bahwa kapal sedang mengisi atau mengosongkan minyak.
- Metode `infoKapal()`: Menambahkan informasi tentang kapasitas minyak.

5. Subkelas `KapalLayar`
Kelas `KapalLayar` untuk kapal yang menggunakan layar.
- Constructor: Menambahkan atribut `jumlahLayar`.
- Metode `aksi()`: Mengembalikan pesan yang menyatakan bahwa kapal sedang mengatur layarnya.
- Metode `infoKapal()`: Menambahkan informasi tentang jumlah layar.

6. Subkelas `KapalPatroli`
Kelas `KapalPatroli` untuk kapal yang melakukan patroli.
- Constructor: Menambahkan atribut `kecepatanMaks`.
- Metode `aksi()`: Mengembalikan pesan bahwa kapal sedang melakukan patroli.
- Metode `infoKapal()`: Menambahkan informasi tentang kecepatan maksimum kapal.

7. Subkelas `KapalSelam`
Kelas `KapalSelam` untuk kapal selam.
- Constructor: Menambahkan atribut `kedalamanMaks`.
- Metode `aksi()`: Mengembalikan pesan bahwa kapal sedang menyelam atau muncul ke permukaan.
- Metode `infoKapal()`: Menambahkan informasi tentang kedalaman maksimal.

8. Subkelas `KapalPenangkapIkan`
Kelas `KapalPenangkapIkan` untuk kapal yang menangkap ikan.
- Constructor: Menambahkan atribut `kapasitasIkan`.
- Metode `aksi()`: Mengembalikan pesan bahwa kapal sedang menangkap ikan.
- Metode `infoKapal()`: Menambahkan informasi tentang kapasitas tangkapan ikan.

9. Subkelas `KapalFeri`

Kelas `KapalFeri` untuk kapal yang mengangkut kendaraan.

- Constructor: Menambahkan atribut `kapasitasKendaraan`.
- Metode `aksi()`: Mengembalikan pesan bahwa kapal sedang mengangkut kendaraan.
- Metode `infoKapal()`: Menambahkan informasi tentang kapasitas kendaraan.

10. Subkelas `KapalPenelitian`
Kelas `KapalPenelitian` untuk kapal yang digunakan untuk penelitian.
- Constructor: Menambahkan atribut `peralatanPenelitian`.
- Metode `aksi()`: Mengembalikan pesan bahwa kapal sedang melakukan penelitian.
- Metode `infoKapal()`: Menambahkan informasi tentang peralatan penelitian.

11. Fungsi tampilkanInfoKapal(kapal)
Menerima objek kapal dan menampilkan informasi serta aktivitas kapal melalui console.

12. Instansiasi Objek
Berbagai jenis kapal dibuat sebagai objek dengan memberikan parameter yang sesuai.
Setiap objek kapal akan memiliki perilaku dan informasi spesifik berdasarkan kelasnya.

13. Menampilkan Info Kapal
Memanggil fungsi tampilkanInfoKapal untuk setiap objek kapal yang telah dibuat, yang akan mencetak informasi dan aksi kapal tersebut ke console.

=> Penggunaan Polimorfisme dalam Fungsi tampilkanInfoKapal:

Dalam fungsi tampilkanInfoKapal(kapal), kita dapat memanggil metode infoKapal() dan aksi() tanpa mengetahui jenis spesifik kapal tersebut. Fungsi ini dapat menerima objek dari kelas apa pun yang merupakan turunan dari Kapal. Dengan cara ini, kita dapat menangani berbagai objek kapal dengan cara yang konsisten, yang merupakan esensi dari polimorfisme.

Contohnya, jika kita memanggil tampilkanInfoKapal(kapalTanker), metode infoKapal() dan aksi() yang dipanggil adalah yang didefinisikan di dalam kelas KapalTanker, tetapi jika kita memanggil tampilkanInfoKapal(kapalLayar), yang dipanggil adalah metode dari kelas KapalLayar. Ini menunjukkan bahwa meskipun objek yang berbeda, mereka dapat diperlakukan secara seragam melalui kelas induk Kapal.

Secara keseluruhan, polimorfisme di dalam kode ini memberikan fleksibilitas dan kemampuan untuk memperluas atau memodifikasi sistem dengan lebih mudah. Kita bisa

menambah jenis kapal baru dengan cara yang sama tanpa harus mengubah struktur yang sudah ada, selama kelas baru tersebut mengikuti pola yang ditentukan oleh kelas induk.

IV. KESIMPULAN

Kode ini mendemonstrasikan konsep pewarisan, di mana subclass mewarisi properti dan method dari class induk Kapal, serta polimorfisme, di mana satu fungsi (tampilkanInfoKapal()) dapat digunakan untuk menangani berbagai subclass dengan cara yang seragam namun memberikan keluaran yang berbeda sesuai dengan tipe objeknya.