

Laporan Praktikum
Mata Kuliah Pemrograman Berorientasi Objek



Pertemuan 5. Praktikum 5
"Polymorphism"

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

Disusun Oleh :
Angga Ardiansyah
2307300/A3

PROGRAM STUDI SISTEM INFORMASI KELAUTAN
UNIVERSITAS PENDIDIKAN INDONESIA
2024

I. PENDAHULUAN


Polymorphism merupakan suatu konsep pemrograman yang memungkinkan objek-objek dari kelas berbeda untuk merespons pemanggilan metode dengan nama yang sama secara berbeda. Dalam praktikum ini, kita akan mengimplementasikan konsep polymorphism dalam konteks pemrograman. Tujuan utama dari praktikum ini adalah untuk memahami cara kerja polymorphism dapat diimplementasikan.

II. ALAT DAN BAHAN

1. Laptop
2. Mouse
3. Visual Code Studio
4. Kode index.html dan javascript

III. LANGKAH KERJA

1. Kode dibawah ini berisi mengenai: Kapal merupakan superclass dari semua jenis kapal. Properti berisi nama, jenis, panjang, dan lebar diinisialisasi melalui constructor. Method yang digunakan yaitu, infokapal() yang berfungsi untuk mengembalikan informasi tentang kapal berupa nama, jenis, panjang, dan lebar.



```
1 // Superclass Kapal
2 class Kapal {
3   constructor(nama, jenis, panjang, lebar) {
4     this.nama = nama;
5     this.jenis = jenis;
6     this.panjang = panjang;
7     this.lebar = lebar;
8   }
9
10  infokapal() {
11    return `Kapal ${this.nama} merupakan jenis ${this.jenis} yang berukuran ${this.panjang} m x ${this.lebar} m.`;
12  }
13 }
14
```

2. Kode dibawah ini berisi mengenai: KapalPenumpang adalah subclass dari Kapal. Method yang digunakan yaitu, infokapal() menambahkan informasi kapasitas penumpang ke method infokapal() dari superclass. Selain itu, subclass ini juga menggunakan method getKapasitas() untuk mengembalikan kapasitas penumpang kapal.

```

Codeium: Refactor | Explain
15 class KapalPenumpang extends Kapal {
Codeium: Refactor | Explain | Generate JSDoc | X
16 constructor(nama, panjang, lebar, kapasitasPenumpang) {
17     super(nama, "kapal feri", panjang, lebar);
18     this.kapasitasPenumpang = kapasitasPenumpang;
19 }
20
Codeium: Refactor | Explain | Generate JSDoc | X
21 infokapal() {
22     return `${super.infokapal()} Kapal ini memiliki kapasitas ${this.kapasitasPenumpang} orang.`;
23 }
24
Codeium: Refactor | Explain | Generate JSDoc | X
25 getKapasitas() {
26     return this.kapasitasPenumpang;
27 }
28 }
29

```

3. Kode dibawah ini berisi mengenai: Subclass Tiket yang menambahkan properti hargaTiket untuk menyimpan informasi harga tiket. Method yang digunakan yaitu, infoTiket() untuk memberikan informasi harga tiket untuk kapal.

```

Codeium: Refactor | Explain
30 class Tiket extends KapalPenumpang {
Codeium: Refactor | Explain | Generate JSDoc | X
31 constructor(nama, panjang, lebar, kapasitasPenumpang, hargaTiket) {
32     super(nama, panjang, lebar, kapasitasPenumpang);
33     this.hargaTiket = hargaTiket;
34 }
35
Codeium: Refactor | Explain | Generate JSDoc | X
36 infoTiket() {
37     return `Harga tiket untuk kapal ${this.nama} adalah Rp ${this.hargaTiket}.`;
38 }
39 }
40

```

4. Kode dibawah ini berisi mengenai: Subclass ini berisi PenumpangNaik dan PenumpangTurun yang berfungsi untuk mengidentifikasi pelabuhan tempat penumpang naik atau turun dari kapal.

```

Codeium: Refactor | Explain
41 class PenumpangNaik extends KapalPenumpang {
Codeium: Refactor | Explain | Generate JSDoc | X
42 constructor(nama, panjang, lebar, kapasitasPenumpang, pelabuhanNaik) {
43     super(nama, panjang, lebar, kapasitasPenumpang);
44     this.pelabuhanNaik = pelabuhanNaik;
45 }
46
Codeium: Refactor | Explain | Generate JSDoc | X
47 infoNaik() {
48     return `Penumpang naik di pelabuhan ${this.pelabuhanNaik}.`;
49 }
50 }
51
Codeium: Refactor | Explain
52 class PenumpangTurun extends KapalPenumpang {
Codeium: Refactor | Explain | Generate JSDoc | X
53 constructor(nama, panjang, lebar, kapasitasPenumpang, pelabuhanTurun) {
54     super(nama, panjang, lebar, kapasitasPenumpang);
55     this.pelabuhanTurun = pelabuhanTurun;
56 }
57
Codeium: Refactor | Explain | Generate JSDoc | X
58 infoTurun() {
59     return `Penumpang turun di pelabuhan ${this.pelabuhanTurun}.`;
60 }
61 }
62

```

5. Kode dibawah ini berisi mengenai: subclass-subclass jenis kapal (cargo, selam, pesiar, dan patroli)

- KapalCargo adalah subclass yang merepresentasikan kapal kargo dengan properti kapasitasMuatan (muatan dalam ton).
- KapalSelam mengkhususkan kapal yang dapat menyelam dengan kedalaman maksimal tertentu (kedalamanMaksimal).
- KapalPesiar menambahkan properti fasilitas, yang merupakan array dari fasilitas yang ada di kapal pesiar.

- KapalPatroli mengkhususkan kapal patroli dengan properti areaPatroli yang menunjukkan area patroli kapal.

```

63 Codeium: Refactor | Explain
class KapalCargo extends Kapal {
64 Codeium: Refactor | Explain | Generate JSDoc | X
  constructor(nama, panjang, lebar, kapasitasMuatan) {
65     super(nama, "kapal kargo", panjang, lebar);
66     this.kapasitasMuatan = kapasitasMuatan;
67   }
68
69 Codeium: Refactor | Explain | Generate JSDoc | X
  infokapal() {
70     return `${super.infokapal()} Kapasitas muatan: ${this.kapasitasMuatan} ton.`;
71   }
72 }
73
74 Codeium: Refactor | Explain
class KapalSelam extends Kapal {
75 Codeium: Refactor | Explain | Generate JSDoc | X
  constructor(nama, panjang, lebar, kedalamanMaksimal) {
76     super(nama, "kapal selam", panjang, lebar);
77     this.kedalamanMaksimal = kedalamanMaksimal;
78   }
79
80 Codeium: Refactor | Explain | Generate JSDoc | X
  infokapal() {
81     return `${super.infokapal()} Kapal ini dapat menyelam hingga kedalaman ${this.kedalamanMaksimal} meter.`;
82   }
83 }
84
85 Codeium: Refactor | Explain
class KapalPesiar extends Kapal {
86 Codeium: Refactor | Explain | Generate JSDoc | X
  constructor(nama, panjang, lebar, fasilitas) {
87     super(nama, "kapal pesiar", panjang, lebar);
88     this.fasilitas = fasilitas;
89   }
90
91 Codeium: Refactor | Explain | Generate JSDoc | X
  infokapal() {
92     return `${super.infokapal()} Fasilitas onboard: ${this.fasilitas.join(", ")}.`;
93   }
94 }
95
96 Codeium: Refactor | Explain
class KapalPatroli extends Kapal {
97 Codeium: Refactor | Explain | Generate JSDoc | X
  constructor(nama, panjang, lebar, areaPatroli) {
98     super(nama, "kapal patroli", panjang, lebar);
99     this.areaPatroli = areaPatroli;
100   }
101
102 Codeium: Refactor | Explain | Generate JSDoc | X
  infokapal() {
103     return `${super.infokapal()} Area patroli: ${this.areaPatroli}`;
104   }
105 }
106

```

6. Kode dibawah ini berisi mengenai: Daftar kapal dari subclass yang berbeda (KapalPenumpang, KapalCargo, dll.) ditambahkan ke dalam array kapalList. Masing-masing objek memanggil method infokapal() yang sesuai dengan implementasi di subclass masing-masing (polymorphism).

```

107 // Polymorphism
108 const kapalList = [
109   new KapalPenumpang("Kapal Penumpang 1", 300, 50, 500),
110   new KapalCargo("Kapal Cargo 1", 400, 60, 10000),
111   new KapalSelam("Kapal Selam 1", 70, 10, 300),
112   new KapalPesiar("Kapal Pesiar 1", 500, 80, ["Kolam renang", "Restoran", "Teater"]),
113   new KapalPatroli("Kapal Patroli 1", 120, 20, "Laut Selatan")
114 ];
115
116 kapalList.forEach(kapal => {
117   console.log(kapal.infokapal());
118 });
119

```

IV. KESIMPULAN

Melalui praktikum ini, mahasiswa dapat menambah wawasan tentang polymorphism bahwa konsep ini dapat memungkinkan objek-objek dari kelas yang berbeda untuk merespons pemanggilan metode dengan nama yang sama secara berbeda. Konsep ini dapat memberikan fleksibilitas dan ekstensibilitas pada kode.