SIM-PELKA (Simulasi Manajemen Pelayanan Kapal di Pelabuhan) Menggunakan Bahasa Pemrograman Python.

---

1. Konsep OOP (Object-Oriented Programming)

   Pada Program ini digunakan paradigma Pemrograman Berorientasi Objek (OOP) dengan membuat sebuah class bernama Kapal.

   a. **Class Kapal** merepresentasikan objek kapal dengan atribut dan perilaku tertentu.

   ```
   class Kapal:
   ```

   b. Atribut (Properties) :

   ```
   self.nama = nama
   self.jenis = jenis
   self.tonase = tonase
   self.status = "Antri"
   self.muatan = 0
   ```

   - **nama** = nama kapal
   - **jenis** = jenis kapal (kargo, tanker, penumpang, dll).
   - **tonase** = kapasitas maksimum muatan kapal.
   - **status** = kondisi kapal saat ini (Antri, Bersandar, Selesai).
   - **muatan** = jumlah muatan yang sedang ditangani.

   c. Method (Behaviors) :

   - **bersandar()** = mengubah status kapal dari Antri ke Bersandar.

     ```
     def bersandar(self):
     ```

   - **bongkar_muat(volume)** = menambah muatan kapal sesuai volume yang diinput (dengan batas maksimum tonase).

     ```
     def bongkar_muat(self, volume):
     ```

   - **selesai_layanan()** = mengubah status kapal menjadi Selesai setelah proses layanan.

     ```
     def selesai_layanan(self):
     ```

   Konsep OOP ini memungkinkan program lebih modular, mudah dipahami, dan bisa dikembangkan lebih lanjut.

2. Function

Program memiliki beberapa fungsi di luar class:

a. **tampilkan_daftar_kapal(daftar_kapal)**

```
def tampilkan_daftar_kapal(daftar_kapal):
    print("\nDaftar Kapal di Pelabuhan:")
    for i, kapal in enumerate(daftar_kapal, 1):
        print(f"{i}. {kapal.nama} | Jenis: {kapal.jenis} | Tonase: {kapal.tonase} ton | "
              f"Status: {kapal.status} | Muatan: {kapal.muatan} ton")
```

Menampilkan seluruh kapal di pelabuhan dalam format terstruktur menggunakan for loop.

b. **simulasikan_bongkar_muat(daftar_kapal)**

```
def simulasikan_bongkar_muat(daftar_kapal):
    shift = 1
    while shift <= 2:
        print(f"\n--- Shift {shift} ---")
        for kapal in daftar_kapal:
            if kapal.status == "Antri":
                kapal.bersandar()
            elif kapal.status == "Bersandar":
                volume = int(input(f"Masukkan volume bongkar/muat untuk {kapal.nama}: "))
                kapal.bongkar_muat(volume)
                kapal.selesai_layanan()
        tampilkan_daftar_kapal(daftar_kapal)
        shift += 1
```

Menjalankan proses simulasi bongkar muat kapal dalam dua shift dengan memanfaatkan while loop dan if statement.

- Pada Pada setiap shift, program akan mengecek status kapal: apakah masih Antri atau sudah Bersandar.
- Jika kapal bersandar, maka pengguna diminta memasukkan volume bongkar/muat, lalu kapal diproses hingga status Selesai.

c. **if __name__ == "__main__":**

```
if __name__ == "__main__":
    daftar_kapal = [
        Kapal("Meratus Jaya", "Kargo", 5000),
        Kapal("Samudra Indah", "Kargo", 8000),
        Kapal("Nusantara", "Kargo", 3000)
    ]

    tampilkan_daftar_kapal(daftar_kapal)
    simulasikan_bongkar_muat(daftar_kapal)
    print("\nSimulasi selesai. Terima kasih telah menggunakan SIM-PELKA.\n")
```

Bagian ini berfungsi sebagai titik awal program. Di dalamnya dibuat daftar kapal, lalu program menampilkan daftar awal dan menjalankan simulasi.

3. IF Statement

Program menggunakan if statement untuk pengambilan keputusan :

a. Pada method **bersandar()** Logika ini mengecek apakah kapal masih antri. Jika iya, status diubah menjadi Bersandar. Jika tidak, ditolak.

```
if self.status == "Antri":
    self.status = "Bersandar"
    print(f"{self.nama} telah bersandar di dermaga.")
else:
    print(f"{self.nama} tidak dapat bersandar. Status saat ini: {self.status}")
```

b. Pada method **bongkar_muat()**. Kapal hanya bisa melakukan bongkar/muat jika statusnya Bersandar.

```
if self.status == "Bersandar":
    if self.muatan + volume > self.tonase:
        volume = self.tonase - self.muatan
    self.muatan += volume
    print(f"{self.nama} melakukan bongkar/muat sebanyak {volume} ton.")
else:
    print(f"{self.nama} tidak dapat bongkar/muat. Status saat ini: {self.status}")
```

Tambahan Validasi. Bagian ini memastikan muatan tidak boleh melebihi tonase kapal.

```
if self.muatan + volume > self.tonase:
    volume = self.tonase - self.muatan
```

c. Pada method **selesai_layanan()**. Kapal hanya bisa menyelesaikan layanan jika statusnya masih Bersandar.

```
if self.status == "Bersandar":
    self.status = "Selesai"
    print(f"{self.nama} selesai layanan dan siap berangkat.\n")
else:
    print(f"{self.nama} tidak dapat menyelesaikan layanan. Status saat ini: {self.status}")
```

4. FOR Loop

Digunakan pada **fungsi tampilkan_daftar_kapal** :

```
for i, kapal in enumerate(daftar_kapal, 1):
    print(f"{i}. {kapal.nama} | Jenis: {kapal.jenis} | Tonase: {kapal.tonase} ton | "
          f"Status: {kapal.status} | Muatan: {kapal.muatan} ton")
```

Loop ini menampilkan daftar semua kapal yang ada di pelabuhan. **enumerate** dipakai untuk memberikan nomor urut pada setiap kapal yang ditampilkan.

5. WHILE Loop

Digunakan pada fungsi **simulasikan_bongkar_muat** :

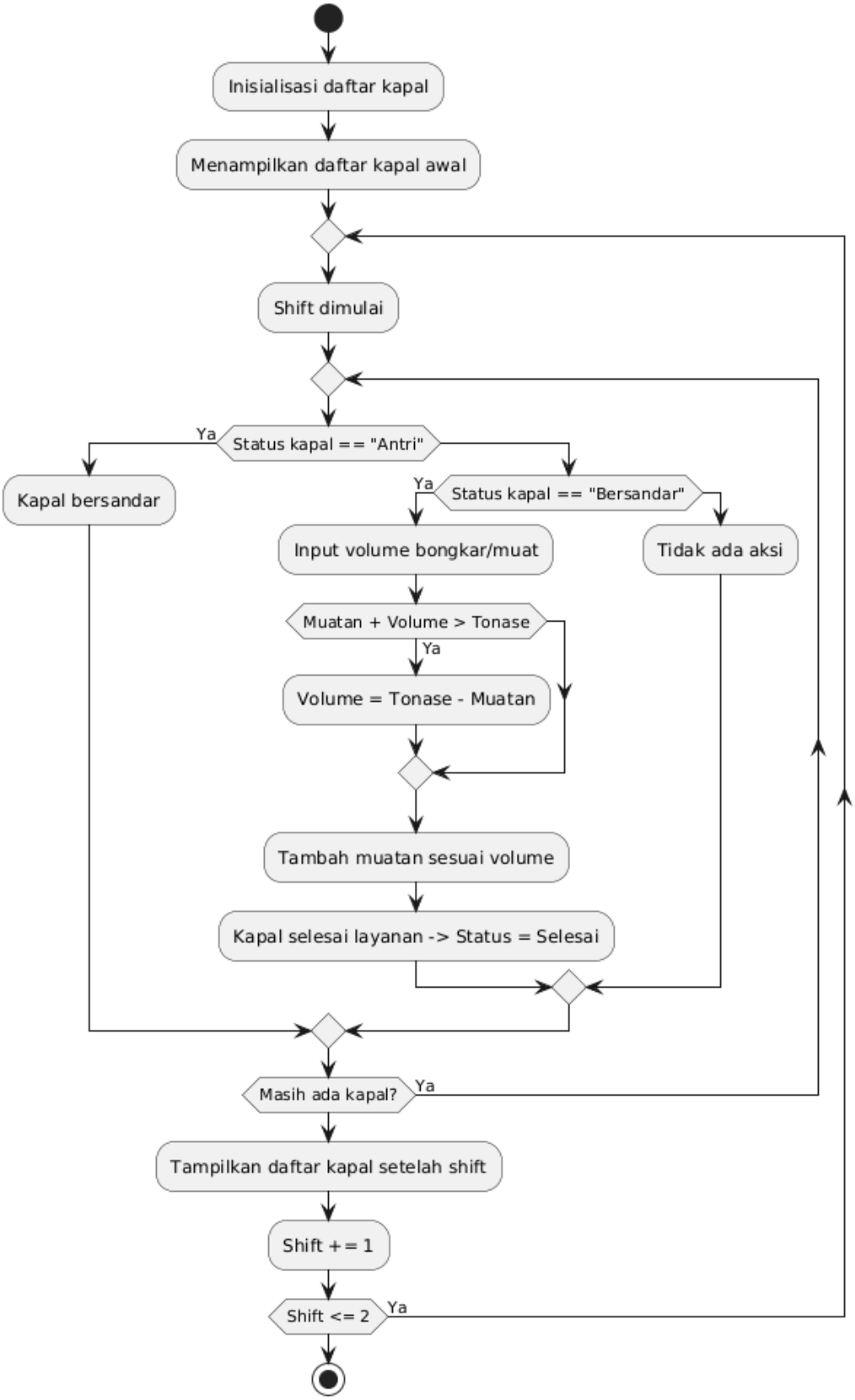
```
while shift <= 2:
    print(f"\n--- Shift {shift} ---")
    for kapal in daftar_kapal:
        if kapal.status == "Antri":
            kapal.bersandar()
        elif kapal.status == "Bersandar":
            volume = int(input(f"Masukkan volume bongkar/muat untuk {kapal.nama}: "))
            kapal.bongkar_muat(volume)
            kapal.selesai_layanan()
    tampilkan_daftar_kapal(daftar_kapal)
    shift += 1
```

L oop ini menjalankan simulasi dalam 2 shift. Setiap shift akan memproses seluruh kapal (apakah antri, bersandar, atau selesai).

6. Alur Program (Flow)

1. Program dijalankan, daftar kapal ditampilkan.
2. Simulasi shift dimulai:
   - Kapal yang masih antri → bersandar.
   - Kapal yang bersandar → dilakukan bongkar/muat (dengan volume input).
   - Setelah bongkar/muat → status kapal diubah ke Selesai.
3. Daftar kapal ditampilkan kembali setelah setiap shift.
4. Proses berhenti setelah 2 shift.

7. Diagram Alur / Flowchat



8. CodeSnap

```python
# SIM-PELKA: Simulasi Manajemen Pelayanan Kapal di Pelabuhan

class Kapal:
    def __init__(self, nama, jenis, tonase):
        self.nama = nama
        self.jenis = jenis
        self.tonase = tonase
        self.status = "Antri"
        self.muatan = 0

    def bersandar(self):
        if self.status == "Antri":
            self.status = "Bersandar"
            print(f"{self.nama} telah bersandar di dermaga.")
        else:
            print(f"{self.nama} tidak dapat bersandar. Status saat ini: {self.status}")

    def bongkar_muat(self, volume):
        if self.status == "Bersandar":
            if self.muatan + volume > self.tonase:
                volume = self.tonase - self.muatan
            self.muatan += volume
            print(f"{self.nama} melakukan bongkar/muat sebanyak {volume} ton.")
        else:
            print(f"{self.nama} tidak dapat bongkar/muat. Status saat ini: {self.status}")

    def selesai_layanan(self):
        if self.status == "Bersandar":
            self.status = "Selesai"
            print(f"{self.nama} selesai layanan dan siap berangkat.\n")
        else:
            print(f"{self.nama} tidak dapat menyelesaikan layanan. Status saat ini: {self.status}")

def tampilkan_daftar_kapal(daftar_kapal):
    print("\nDaftar Kapal di Pelabuhan:")
    for i, kapal in enumerate(daftar_kapal, 1):
        print(f"{i}. {kapal.nama} | Jenis: {kapal.jenis} | Tonase: {kapal.tonase} ton | "
              f"Status: {kapal.status} | Muatan: {kapal.muatan} ton")

def simulasikan_bongkar_muat(daftar_kapal):
    shift = 1
    while shift <= 2:
        print(f"\n--- Shift {shift} ---")
        for kapal in daftar_kapal:
            if kapal.status == "Antri":
                kapal.bersandar()
            elif kapal.status == "Bersandar":
                volume = int(input(f"Masukkan volume bongkar/muat untuk {kapal.nama}: "))
                kapal.bongkar_muat(volume)
                kapal.selesai_layanan()
        tampilkan_daftar_kapal(daftar_kapal)
        shift += 1

if __name__ == "__main__":
    daftar_kapal = [
        Kapal("Meratus Jaya", "Kargo", 5000),
        Kapal("Samudra Indah", "Kargo", 8000),
        Kapal("Nusantara", "Kargo", 3000)
    ]

    tampilkan_daftar_kapal(daftar_kapal)
    simulasikan_bongkar_muat(daftar_kapal)
    print("\nSimulasi selesai. Terima kasih telah menggunakan SIM-PELKA.\n")
```