```
class Node:
    def __init__(self, data):
        self.data = data
        self.prev = None
        self.next = None
```

Penjelasan per baris:

- class Node: Mendefinisikan kelas Node, yang mewakili satu simpul dalam linked list.
- def __init__(self, data): Konstruktor yang menerima data dan otomatis dipanggil saat objek Node dibuat.
- self.data = data: Menyimpan nilai ke dalam node.
- self.prev = None: Mengatur pointer ke node sebelumnya (awalnya kosong).
- self.next = None: Mengatur pointer ke node selanjutnya (awalnya kosong).

```
class DoubleLinkedList:
    def __init__(self):
        self.head = None
```

Penjelasan per baris:

- class DoubleLinkedList: Kelas utama untuk menyimpan dan mengelola seluruh linked list.
- def __init__(self): Konstruktor yang dipanggil saat objek dibuat.
- self.head = None: Menunjukkan bahwa list masih kosong (tidak ada node).

```
def append(self, data):
        new_node = Node(data)
        if self.head is None:
            self.head = new_node
            return
        cur = self.head
        while cur.next:
            cur = cur.next
        cur.next = new_node
        new_node.prev = cur
```

Penjelasan per baris:

- def append(self, data): Fungsi untuk menambahkan node baru ke akhir list.

- new_node = Node(data): Membuat node baru dengan data yang diberikan.
- if self.head is None: Jika list masih kosong...
- self.head = new_node: ...node baru jadi node pertama (head).
- Return: Keluar dari fungsi setelah menambahkan node pertama.
- cur = self.head: Jika tidak kosong, mulai dari head.
- while cur.next: Selama belum sampai di node terakhir...
- cur = cur.next: ...lanjutkan ke node berikutnya.
- cur.next = new_node: Setelah sampai terakhir, sambungkan node baru.
- new_node.prev = cur: Node baru menunjuk balik ke node sebelumnya.

```
def display(self):
    cur = self.head
    while cur:
        print(cur.data, end=" <-> " if cur.next else "\n")
        cur = cur.next
```

Penjelasan per baris:

- def display(self): Fungsi untuk mencetak seluruh isi linked list
- cur = self.head: Mulai dari node pertama.
- while cur: Selama node saat ini masih ada...
- print(cur.data, end=" <-> " if cur.next else "\n"): Cetak datanya. Tambahkan " <-> " jika bukan node terakhir.
- cur = cur.next: Lanjutkan ke node berikutnya.

```
def delete_first(self):
    if self.head is None:
        print("List kosong")
        return
    print(f"Menghapus node awal: {self.head.data}")
    self.head = self.head.next
    if self.head:
        self.head.prev = None
```

Penjelasan per baris:

- def delete_first(self): Fungsi untuk menghapus node pertama.
- if self.head is None: Jika list kosong...
- print("List kosong"): ...tampilkan pesan.

- Return: Keluar dari fungsi.
- print(f"Menghapus node awal: {self.head.data}"): Tampilkan data node yang akan dihapus.
- self.head = self.head.next: Geser head ke node berikutnya.
- if self.head: Jika masih ada node...
- self.head.prev = None: ...putuskan link ke node sebelumnya.

```
def delete_last(self):
    if self.head is None:
        print("List kosong")
        return
    cur = self.head
    if cur.next is None:
        print(f"Menghapus node terakhir: {cur.data}")
        self.head = None
        return
    while cur.next:
        cur = cur.next
    print(f"Menghapus node terakhir: {cur.data}")
    cur.prev.next = None
```

Penjelasan per baris:

- def delete_last(self): Fungsi untuk menghapus node terakhir.
- if self.head is None: Jika list kosong...
- print("List kosong"): ...tampilkan pesan.
- Return: Keluar dari fungsi.
- cur = self.head: Mulai dari node pertama.
- if cur.next is None: Jika hanya ada satu node...
- print(f"Menghapus node terakhir: {cur.data}"): Tampilkan data yang dihapus.
- self.head = None: Hapus satu-satunya node.
- Return: Keluar dari fungsi.
- while cur.next: Telusuri ke node terakhir.
- cur = cur.next: Pindah ke next.
- print(f"Menghapus node terakhir: {cur.data}"): Tampilkan data yang dihapus.
- cur.prev.next = None: Putuskan hubungan dari node sebelumnya.

```
def delete_by_value(self, value):
    if self.head is None:
        print("List kosong")
        return
    cur = self.head
    while cur:
        if cur.data == value:
            print(f"Menghapus node dengan nilai: {value}")
            if cur.prev:
                cur.prev.next = cur.next
            else:
                self.head = cur.next
            if cur.next:
                cur.next.prev = cur.prev
            return
        cur = cur.next
    print(f'Data {value} tidak ditemukan dalam list.")
```

Penjelasan per baris:

- def delete_by_value(self, value): Fungsi untuk menghapus node berdasarkan nilai tertentu.
- if self.head is None: Jika list kosong...
- print("List kosong"): ...tampilkan pesan.
- Return: Keluar dari fungsi.
- cur = self.head: Mulai pencarian dari node pertama.
- while cur: Telusuri node satu per satu.
- if cur.data == value: Jika ditemukan nilai yang cocok...
- print(f"Menghapus node dengan nilai: {value}"): Tampilkan pesan.
- if cur.prev: Jika bukan node pertama...
- cur.prev.next = cur.next: Sambungkan node sebelumnya ke node setelahnya.
- else: Jika node adalah head...
- self.head = cur.next: Geser head ke node berikutnya.
- if cur.next: Jika node bukan yang terakhir...
- cur.next.prev = cur.prev: Sambungkan node setelahnya ke node sebelumnya.
- Return: Keluar dari fungsi.
- cur = cur.next: Lanjutkan ke node berikutnya.

- print(f"Data {value} tidak ditemukan dalam list."): Jika tidak ada data yang cocok, tampilkan pesan.

```
dll = DoubleLinkedList()

dll.append(10)

dll.append(20)

dll.append(30)

dll.append(40)
```

Penjelasan per baris:

- dll = DoubleLinkedList(): Membuat objek baru dll dari kelas DoubleLinkedList.
- dll.append(10): Menambahkan node dengan nilai 10.
- dll.append(20): Menambahkan node dengan nilai 20.
- dll.append(30): Menambahkan node dengan nilai 30.
- dll.append(40): Menambahkan node dengan nilai 40.
- Sekarang list berisi: 10 <-> 20 <-> 30 <-> 40

```
print("Isi awal:")

dll.display()
```

Penjelasan:

- print("Isi awal:"): Menampilkan teks sebagai keterangan output.
- dll.display(): Memanggil fungsi display() untuk mencetak isi list: 10 <-> 20 <-> 30 <-> 40

```
dll.delete_first()

dll.display()
```

Penjelasan:

- dll.delete_first(): Menghapus node pertama (nilai 10).
- dll.display(): Cetak isi list: 20 <-> 30 <-> 40

```
dll.delete_last()

dll.display()
```

Penjelasan:

- dll.delete_last(): Menghapus node terakhir (nilai 40).
- dll.display(): Cetak isi list: 20 <-> 30

```
dll.delete_by_value(20)

dll.display()
```

Penjelasan:

- dll.delete_by_value(20): Menghapus node dengan nilai 20.
- dll.display(): Cetak isi list: 30

```
dll.delete_by_value(99)  # Kasus data tidak ditemukan
```

Penjelasan:

- dll.delete_by_value(99): Mencoba menghapus node dengan nilai 99, tapi tidak ditemukan.
- Maka akan muncul: Data 99 tidak ditemukan dalam list.