```python
# Definisi kelas Node untuk merepresentasikan satu elemen dalam double linked-list
class Node:
    def __init__(self, data):
        self.data = data         # Menyimpan data pada node
        self.prev = None         # Pointer ke node sebelumnya, awalnya None
        self.next = None         # Pointer ke node berikutnya, awalnya None

# Definisi kelas DoublyLinkedList untuk mengelola node-node secara berantai dua arah
class DoublyLinkedList:
    def __init__(self):
        self.head = None         # Pointer ke node pertama (head) dalam list, awalnya None (kosong)

    # Fungsi untuk menambahkan node baru di akhir list
    def append(self, data):
        new_node = Node(data)    # Membuat node baru dengan data yang diterima
        if not self.head:        # Jika list masih kosong (head None)
            self.head = new_node # Maka node baru menjadi head
            return
        last = self.head         # Mulai dari head
        while last.next:         # Iterasi sampai node terakhir (yang next-nya None)
            last = last.next
        last.next = new_node     # Update next node terakhir ke node baru
        new_node.prev = last     # Node baru prev-nya adalah last
# Fungsi untuk menghapus node pertama (head) dari list
def delete_first(self):
    if not self.head:        # Jika list kosong, tidak ada yang dihapus
        return
    if self.head.next:       # Jika lebih dari 1 node
        self.head = self.head.next   # Geser head ke node berikutnya
        self.head.prev = None          # Prev di head baru menjadi None karena dia sekarang node pertama
    else:
        self.head = None      # Jika hanya 1 node saja, list jadi kosong setelah dihapus

# Fungsi untuk menghapus node terakhir dari list
def delete_last(self):
    if not self.head:         # Jika list kosong, tidak ada yang dihapus
        return
    if not self.head.next:    # Jika hanya 1 node dalam list
        self.head = None      # Langsung hapus head (list kosong)
        return
    last = self.head
    while last.next:          # Cari node terakhir
        last = last.next
    last.prev.next = None     # Node sebelum last, next-nya di-set None (menandakan jadi node akhir)
# Fungsi untuk menghapus node berdasarkan nilai data tertentu
def delete_by_value(self, value):
    if not self.head:          # Jika list kosong, langsung return
        return
    current = self.head        # Mulai dari node pertama
    while current:
        if current.data == value:   # Jika data node sama dengan nilai yang dicari
            if current.prev:        # Jika node bukan head
                current.prev.next = current.next  # Hubungkan node prev ke node next dari current
            if current.next:        # Jika node bukan tail
                current.next.prev = current.prev  # Hubungkan node next ke node prev dari current
            if current == self.head:  # Jika node yang ingin dihapus adalah head
                self.head = current.next  # Update head ke node setelah current
            return                 # Hentikan proses setelah menghapus node
        current = current.next     # Lanjut ke node berikutnya

# Fungsi untuk menampilkan isi linked list dari depan ke belakang
def display(self):
    current = self.head          # Mulai dari head
    while current:
        print(current.data, end=" <-> ")  # Cetak data node dan tanda panah ganda
        current = current.next               # Lanjut ke node berikutnya
    print("None")                             # Akhiri tampilan dengan None (tanda akhir list)
```

```
# Contoh pemakaian kelas DoublyLinkedList dan fungsi penghapusan:
dll = DoublyLinkedList()  # Buat objek double linked-list baru
dll.append(1)             # Tambah node dengan data 1
dll.append(2)             # Tambah node dengan data 2
dll.append(3)             # Tambah node dengan data 3
dll.append(4)             # Tambah node dengan data 4

print("List sebelum penghapusan:")
dll.display()             # Tampilkan list saat ini

dll.delete_first()        # Hapus node pertama
print("List setelah menghapus node pertama:")
dll.display()             # Tampilkan list lagi

dll.delete_last()         # Hapus node terakhir
print("List setelah menghapus node terakhir:")
dll.display()             # Tampilkan list lagi

dll.delete_by_value(2)    # Hapus node dengan data 2
print("List setelah menghapus node dengan nilai 2:")
dll.display()             # Tampilkan list terakhir
```

Penjelasan:
Baris 1 : Definisi kelas Node untuk merepresentasikan satu elemen dalam double linked-list
Baris 4 : Menyimpan data pada node
Baris 5 : Pointer ke node sebelumnya, awalnya None
Baris 6 : Pointer ke node berikutnya, awalnya None
Baris 8 : Definisi kelas DoublyLinkedList untuk mengelola node-node secara
Baris 11 : Pointer ke node pertama (head) dalam list, awalnya None (kosong)
Baris 13 : Fungsi untuk menambahkan node baru di akhir list
Baris 15 : Membuat node baru dengan data yang diterima
Baris 16 : Jika list masih kosong (head None)
Baris 17 : Maka node baru menjadi head
Baris 19 : Mulai dari head
Baris 20 : Iterasi sampai node terakhir (yang next-nya None)
Baris 22 : Update next node terakhir ke node baru
Baris 23 : Node baru prev-nya adalah last
Baris 25 ; Fungsi untuk menghapus node pertama (head) dari list
Baris 27 : Jika list kosong, tidak ada yang dihapus
Baris 29 : Jika lebih dari 1 node
Baris 30 : Geser head ke node berikutnya
Baris 31 : Prev di head baru menjadi None karena dia sekarang node pertama
Baris 33 : Jika hanya 1 node saja, list jadi kosong setelah dihapus
Baris 35 : Fungsi untuk menghapus node terakhir dari list
Baris 37 : Jika list kosong, tidak ada yang dihapus
Baris 39 : Jika hanya 1 node dalam list
Baris 40 : Langsung hapus head (list kosong)
Baris 43 : Cari node terakhir

Baris 45 : Node sebelum last, next-nya di-set None (menandakan jadi node akhir)
Baris 47 : Fungsi untuk menghapus node berdasarkan nilai data tertentu
Baris 49 : Jika list kosong, langsung return
Baris 51 : Mulai dari node pertama
Baris 53 : Jika data node sama dengan nilai yang dicari
Baris 54 : Jika node bukan head
Baris 55 : Hubungkan node prev ke node next dari current
Baris 56 : Jika node bukan tail
Baris 57 : Hubungkan node next ke node prev dari current
Baris 58 : Jika node yang ingin dihapus adalah head
Baris 59 : Update head ke node setelah current
Baris 60 : Hentikan proses setelah menghapus node
Baris 61 : Lanjut ke node berikutnya
Baris 63 : Fungsi untuk menampilkan isi linked list dari depan ke belakang
Baris 65 : Mulai dari head
Baris 67 : Cetak data node dan tanda panah ganda
Baris 68 : Lanjut ke node berikutnya
Baris 69 : Akhiri tampilan dengan None (tanda akhir list)
Baris 71 : Contoh pemakaian kelas DoublyLinkedList dan fungsi penghapusan:
Baris 72 : Buat objek double linked-list baru
Baris 73 : Tambah node dengan data 1
Baris 74 : Tambah node dengan data 2
Baris 75 : Tambah node dengan data 3
Baris 76 : Tambah node dengan data 4
Baris 79 : Tampilkan list saat ini
Baris 81 : Hapus node pertama
Baris 83 : Tampilkan list lagi
Baris 85 : Hapus node terakhir
Baris 87 : Tampilkan list lagi
Baris 89 : Hapus node dengan data 2
Baris 91 : Tampilkan list terakhir