1. Doubly Linked List

```
public class DoublyLinkedList {
    private Node head;

    // Other methods...

    /**
     * Adds a node at the end of the list.
     *
     * @param newData The data to be added to the new node.
     */
    public void append(int newData) {
        // 1. Allocate node
        // 2. Put in the data
        Node newNode = new Node(newData);

        Node last = head; // Used in step 5

        // 3. This new node is going to be the last node, so
        // make next of it as null
        newNode.setNext(null);

        // 4. If the Linked List is empty, then make the new
        // node as head
        if (head == null) {
            newNode.setPrev(null);
            head = newNode;
            return;
        }

        // 5. Else traverse till the last node
        while (last.getNext() != null)
            last = last.getNext();

        // 6. Change the next of the last node
        last.setNext(newNode);

        // 7. Make the last node as previous of the new node
        newNode.setPrev(last);
    }
}

class Node {
    private int data;
    private Node prev;
    private Node next;

    /**
     * Constructor for the Node class.
```

```
 *
 * @param newData The data to be stored in the node.
 */
public Node(int newData) {
    data = newData;
    prev = null;
    next = null;
}

// Getter and setter methods for data, prev, and next...

/**
 * Gets the data stored in the node.
 *
 * @return The data stored in the node.
 */
public int getData() {
    return data;
}

/**
 * Sets the data in the node.
 *
 * @param newData The new data to be set.
 */
```

```
public void setData(int newData) {
   data = newData;
}

/**
 * Gets the previous node.
 *
 * @return The previous node.
 */
public Node getPrev() {
   return prev;
}

/**
 * Sets the previous node.
 *
 * @param newPrev The new previous node.
 */
public void setPrev(Node newPrev) {
   prev = newPrev;
}

/**
 * Gets the next node.
 *
```

```
     * @return The next node.
     */
    public Node getNext() {
        return next;
    }

    /**
     * Sets the next node.
     *
     * @param newNext The new next node.
     */
    public void setNext(Node newNext) {
        next = newNext;
    }
}
```

2. DLL.java

```
public class DLL<T> {
    private Node<T> head; // head of list

    // ... Other methods ...

    public static void main(String[] args) {
        /* Start with the empty list */
```

```
DLL<Integer> intDLL = new DLL<>();


// Insert 6. So linked list becomes 6->NULL

intDLL.append(6);


// Insert 7 at the beginning. So linked list becomes 7->6->NULL

intDLL.push(7);


// Insert 1 at the beginning. So linked list becomes 1->7->6->NULL

intDLL.push(1);


// Insert 4 at the end. So linked list becomes 1->7->6->4->NULL

intDLL.append(4);


// Insert 8, after 7. So linked list becomes 1->7->8->6->4->NULL

intDLL.InsertAfter(intDLL.head.getNext(), 8);


System.out.println("Created DLL of Integers is: ");

intDLL.printlist(intDLL.head);


// Similarly, you can create DLLs for Double and String

DLL<Double> doubleDLL = new DLL<>();

doubleDLL.append(2.0);

doubleDLL.push(6.0);

doubleDLL.push(12.0);
```

```
        doubleDLL.InsertAfter(doubleDLL.head.getNext(), 8.0);

        System.out.println("\nCreated DLL of Doubles is: ");
        doubleDLL.printlist(doubleDLL.head);

        DLL<String> stringDLL = new DLL<>();
        stringDLL.append("Dog");
        stringDLL.push("Cat");
        stringDLL.push("Horse");
        stringDLL.InsertAfter(stringDLL.head.getNext(), "Dog");

        System.out.println("\nCreated DLL of Strings is: ");
        stringDLL.printlist(stringDLL.head);
    }
}

class Node<T> {
    private T data;
    private Node<T> prev;
    private Node<T> next;

    // Constructor
    public Node(T newData) {
        data = newData;
        prev = null;
    next = null;

}


// Getter and setter methods for data, prev, and next...


public T getData() {

    return data;

}


public void setData(T newData) {

    data = newData;

}


public Node<T> getPrev() {

    return prev;

}


public void setPrev(Node<T> newPrev) {

    prev = newPrev;

}


public Node<T> getNext() {

    return next;

}
```

```
    public void setNext(Node<T> newNext) {

        next = newNext;

    }

}
```