```
Class BasicDoubleLinkedList<T>:
  Class Node<T>:
    data: T        // Data in the node
    prev: Node<T>   // Previous node
    next: Node<T>   // Next node
  head: Node<T>      // First node
  tail: Node<T>      // Last node
  size: int          // Number of nodes

  Constructor():
    head = tail = null
    size = 0

  // Add to front
  Method addToFront(data: T):
    newNode = new Node(data)
    if head is null:
      head = tail = newNode
    else:
      newNode.next = head
      head.prev = newNode
      head = newNode
    size++

  // Add to end
  Method addToEnd(data: T):
```

```
    newNode = new Node(data)
    if head is null:
        head = tail = newNode
    else:
        tail.next = newNode
        newNode.prev = tail
        tail = newNode
    size++

// Iterator for the list
Class Iterator:
    current: Node<T>

    Constructor():
        current = head

    Method next() -> T:
        if current is null: throw NoSuchElementException
        data = current.data
        current = current.next
        return data

    Method hasNext() -> boolean:
        return current != null

    Method previous() -> T:
```

```
        if current is null: throw NoSuchElementException
        data = current.data
        current = current.prev
        return data

    Method hasPrevious() -> boolean:
        return current != null

Method iterator() -> Iterator:
    return new Iterator()
```

```
Class SortedDoubleLinkedList<T> extends BasicDoubleLinkedList<T>:
    comparator: Comparator<T>  // Comparator for sorting

    Constructor(comparator: Comparator<T>):
        this.comparator = comparator

    // Add in sorted order
    Method add(data: T):
        newNode = new Node(data)
        if head is null:
            head = tail = newNode
        else:
            current = head
            while current != null and comparator.compare(data, current.data) > 0:
                current = current.next
            if current is null:  // Insert at end
                tail.next = newNode
                newNode.prev = tail
                tail = newNode
            else:  // Insert before current
                newNode.next = current
                newNode.prev = current.prev
                if current.prev is null:
                    head = newNode
                else:
                    current.prev.next = newNode
```

```
        current.prev = newNode
    size++

// Block unsorted additions
Method addToFront(data: T):
    throw UnsupportedOperationException

Method addToEnd(data: T):
    throw UnsupportedOperationException
```