Дек (Deque) в Java представляет собой двухстороннюю очередь, которая позволяет добавлять и удалять элементы с обоих концов. Дек является интерфейсом в Java, а его реализации предоставляются классами ArrayDeque и LinkedList. Обе реализации поддерживают операции добавления, удаления и доступа к элементам, а также позволяют использовать дек в качестве стека или очереди.

Вот некоторые основные операции, которые можно выполнять со структурой данных Дек:

1. Добавление элемента в начало: addFirst(E element) или offerFirst(E element)
   Эти методы добавляют элемент в начало дека. Если дек имеет ограниченный размер и достигает предела, методы addFirst выбрасывают исключение, в то время как offerFirst возвращает специальное значение (например, false или null), указывающее на успешность или неуспешность добавления элемента.
2. Добавление элемента в конец: addLast(E element) или offerLast(E element)
   Эти методы добавляют элемент в конец дека. Поведение методов аналогично addFirst и offerFirst соответственно.
3. Удаление элемента из начала: removeFirst() или pollFirst()
   Эти методы удаляют и возвращают элемент с начала дека. Если дек пуст, метод removeFirst выбрасывает исключение, а pollFirst возвращает специальное значение (например, null), указывающее на отсутствие элементов.
4. Удаление элемента из конца: removeLast() или pollLast()
   Эти методы удаляют и возвращают элемент с конца дека. Поведение методов аналогично removeFirst и pollFirst соответственно.
5. Получение элемента из начала: getFirst() или peekFirst()
   Эти методы возвращают элемент с начала дека без его удаления. Если дек пуст, метод getFirst выбрасывает исключение, а peekFirst возвращает специальное значение (например, null), указывающее на отсутствие элементов.
6. Получение элемента из конца: getLast() или peekLast()
   Эти методы возвращают элемент с конца дека без его удаления. Поведение методов аналогично getFirst и peekFirst соответственно.
   Дек предлагает эффективные операции добавления и удаления элементов как из начала, так и из конца, и обеспечивает надежное хранение и доступ к данным. Зависимости от конкретной реализации, некоторые операции в Деке могут иметь различную сложность времени выполненения, например, методы LinkedList обычно требуют больше времени, чем методы ArrayDeque.