***1. Как устроена структура данных «Динамический массив». какой интерфейс она предоставляет. В чем ее преимущества и недостатки.***

***//*** Динамический массив — это массив, который умеет изменять свой размер во время выполнения программы. В Java эту роль играют в основном классы ArrayList и LinkedList. ArrayList — это коллекция, которая реализует интерфейс IList<T> и использует массив для хранения элементов. Как и связный список, ArrayList может хранить произвольное число элементов (ограниченное только объемом доступной памяти), но в остальном ведет себя как массив.  
Интерфейс IList<T> предоставляет все методы ICollection<T> и дополнительно — методы для чтения, вставки и удаления элементов по индексу.

***2. Как устроена структура данных «Двусвязный список». Какой интерфейс она предоставляет. В чем ее преимущества и недостатки.***

***//*** Двусвязный список - структура данных, которая хранит информацию в памяти распределено и состоит из узлов, которые хранят полезные данные и указатели на предыдущий узел и следующий узел. Структура выглядит так: Хранятся данные , ссылка на следующий и предыдущий элементы. Ссылки указывают на другие элементы двусвязного списка.   
  
Интерфейс:

1. add - Добавляет элемент в конец списка,

2. сlear - Удаляет все элементы из списка,

3. AddFirst - Добавляет переданный элемент в начало списка.

4. AddLast - Добавляет переданный элемент в конец списка.

5. Remove - Удаляет первый элемент списка со значением, равным переданному.   
  
Двусвязный список является противоположностью массива ,какие-то операции в нем делаются быстрее массива,а какие-то дольше.Что и является преимуществами и недостатками данной структуры данных. Также достоинства: операции вставки и удаления принадлежит к классу O(1),Независимо от текущего элемента спуска и его емкости, для вставки или удаления элемента всегда требуется одно и то же время.Основным недостатком связных списков является то, что получение доступа к их элементам принадлежит к классу О(n). В этом случае важно количество элементов в списке: при поиске n-ного элемента мы начинаем с некоторой позиции в списке и переходим по ссылкам вплоть до искомого элемента. Чем больше элементов в списке, тем больше переходов придется совершить.По сравнению с классом TList связные списки требуют большего объема памяти.

***3. Что такое логическая структура данных? Опишите известные вам логические структуры данных. Какой интерфейс они предоставляют, как реализуются?***

***//*** Логические структуры данных упаковывают некоторые физические структуры данных снабжая их либо дополнительным, либо новым интерфейсом характерным для этой структуры данных.   
Логические структуры данных:   
1. stack   
абстрактный тип данных, представляющий собой список элементов, организованных по принципу LIFO (last in — first out, «последним пришёл — первым вышел»).   
  
Интерфейс:   
Push – добавление элемента на вершину стека.   
Pop – получить значение элемента на вершине стека и извлечь его.   
isEmpty — метод, который будет проверять стек на пустоту.   
Peek – получить значение элемента на вершие без извлечения.   
  
Реализация на массиве:   
Push:   
1) проверка на переполнение   
2) увеличивается top   
3) положить элемент в ячейку (Top)   
Peek:   
1) Проверка на пустоту (если пустое - ошибка)   
2) Возвращает элемент с номером (если не пустой)   
Pop:   
1) Првоерка на пустоту (если пустое - ошибка)   
2) Возвращает элемент с номером Top   
Реализация на списке:   
Push:   
1) Вызывает добавлеие в конец (обовляет Top)   
Peek:   
1) Возвращает данные из top   
Pop:   
1) Возвращает данные из top   
2) Удаляет top из data   
3) Обовляет ссылку top   
2. Queue   
Подобно стекам, очередь — хранит элемент последовательным образом. Существенное отличие от стека – использование FIFO (First in First Out) вместо LIFO.   
Интерфейс:   
Enqueue — вставляет элемент в конец очереди   
  
Dequeue — удаляет элемент из начала очереди   
  
Peek – получить значение без извлечения   
  
Size – фактическое кол-во элементов в очереди   
Реализация:   
(сложная)   
Deque (двустороняя очередь):   
Контейнер отрытый с двух сторон для добавления и удаляения элементов.   
Интерфейс:   
1) push-front   
2) peek-front   
3) pop-front   
4) push-back   
5) peek-back   
6) pop-back   
Реализация на списке:   
1) push-back – добавление в конец списка   
2) pop-back – последний элемент удаляется из списка и возращается его значение.