Pro Java №6

1. **Линейные структуры данных**:
    - **Массив**: Коллекция элементов фиксированной длины, доступ к которым осуществляется по индексу.
    - **Список**: Коллекция элементов переменной длины, доступ к которым осуществляется по порядку их добавления.
    - **Очередь**: Структура данных, работающая по принципу FIFO (First In, First Out).
    - **Стек**: Структура данных, работающая по принципу LIFO (Last In, First Out).
2. **Нелинейные структуры данных**:
    - **Дерево**: Иерархическая структура данных, состоящая из узлов, где каждый узел может иметь несколько дочерних узлов.
    - **Граф**: Сеть узлов (вершин), связанных ребрами (дугами), которые могут быть направленными или ненаправленными.
3. **Статичные структуры данных**:
    - **Массив**: Размер массива фиксируется при создании и не может быть изменен.

Динамический массив (ArrayList): Массив, который автоматически увеличивает свой размер по мере необходимости.

```
// Создание ArrayList для хранения элементов типа Integer
        ArrayList<Integer> list = new ArrayList<>();

        // Добавление элементов в ArrayList
        list.add(1);
        list.add(2);
        list.add(3);
```

Внутренняя структура `ArrayList` в Java основана на массиве, и изначально его емкость по умолчанию составляет 10 элементов. Когда к `ArrayList` добавляется больше элементов, чем позволяет его текущая емкость, он автоматически увеличивает свою емкость и создает новый массив, копируя в него значения из старого массива.

Метод `size()` в классе `ArrayList` возвращает количество элементов, хранящихся в списке.

В Java коллекции могут содержать только ссылочные типы данных. Для примитивных типов данных, таких как `int`, `char`, `double` и т.д., используются классы-обертки (wrapper classes).

Чтобы вставить элемент в определенный индекс в `ArrayList` и сдвинуть остальные элементы вправо, можно использовать метод `add(int index, E element)`.

Метод `set(int index, E element)` в классе `ArrayList`.

Наследование по типу `List` вместо `ArrayList` рекомендуется для большей гибкости и поддержки полиморфизма.