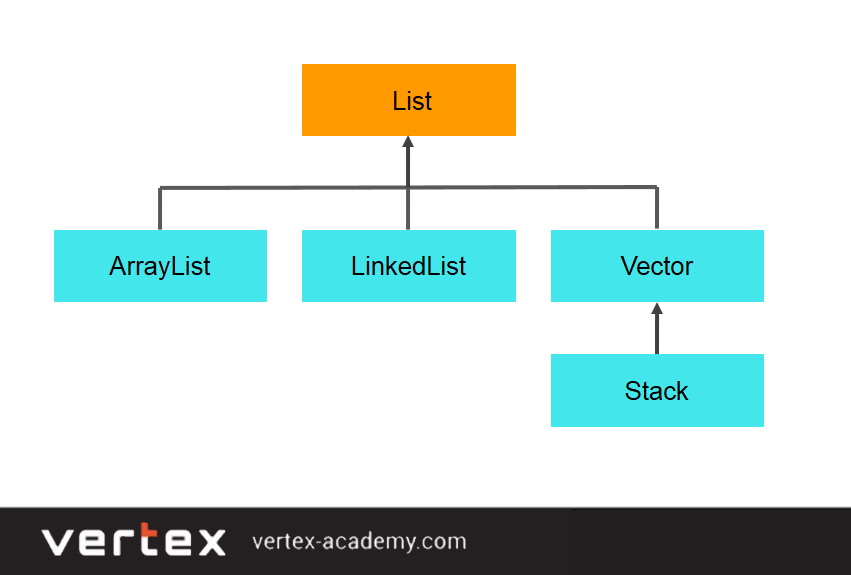
**Что такое список (List) в Java**

До этого Вы, скорее всего, были знакомы только с одним способом хранения данных - массивом. Давайте рассмотрим и другие способы. Один из таких "способов" - это хранение данных в списке.

List упорядочен в порядке добавления.

**Виды списков**

Списков существует несколько. **List**-это только название интерфейса, который объединяет несколько реализаций:



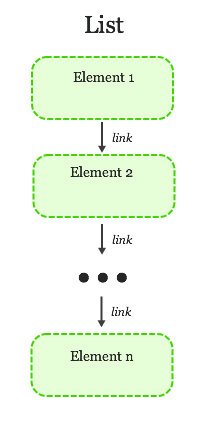
На практике мы чаще всего используем:

* **ArrayList**( мы также рассматривали в статье ["Что такое ArrayList"](https://vertex-academy.com/tutorials/ru/arraylist-v-java/))
* **LinkedList**

На **Vector** можете не смотреть

**Как хранятся данные**

Так что же все-таки такое **List**? В памяти его можно представить так:



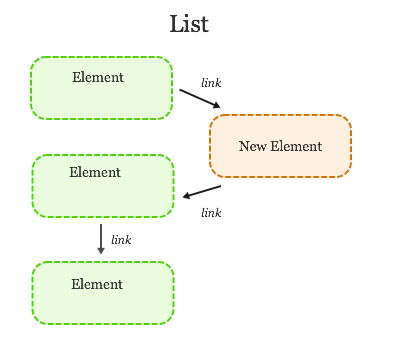
То есть список, в классическом смысле - это тип данных, в котором каждый элемент содержит какой-то контент (например, **String**, **Integer** или вообще какой-нибудь объект), а также ссылку на следующий элемент списка.

**Зачем?**

Основное преимущество списка по сравнению с массивом - его можно легко **расширять**.

Если у нас есть массив на 3 элемента, а нам вдруг понадобилось положить туда 10 элементов, нам просто придется создавать **новый массив** на 10 элементов. Только так и никак иначе! ;(

Но со списками все по-другому. Мы можем **свободно добавлять элементы** в начало, середину или конец списка. Все потому, что для массива нам нужен один "сплошной" участок памяти, и чтобы его расширить, придется искать другой "целый" кусок памяти. Но с **List** проще - мы просто меняем ссылки:

[](https://vertex-academy.com/tutorials/wp-content/uploads/2018/05/2.bmp)

Итак, перейдем от теории к практике.

**Отличие ArrayList и LinkedList**

Не вдаваясь в подробности, запомните, что:

* **LinkedList** лучше использовать, если Вы знаете, что будете часто добавлять и удалять элементы из списка.
* **ArrayList** лучше использовать, если Вам главное иметь быстрый доступ к элементам списка.

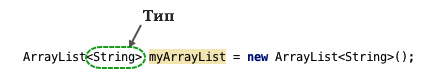
**Синтаксис**

|  |  |
| --- | --- |
|  | ArrayList<String> myArrayList = new ArrayList<String>(); |

Или

|  |  |
| --- | --- |
|  | LinkedList<String> myLinkedList = new LinkedList<String>(); |

Обратите внимание, что в скобках мы задаем тип данных, который будет храниться в списке:



В наших примерах это **String**, но Вы можете задать любой другой.

Кстати, далее мы будем приводить примеры на **ArrayList**.

**Добавление элемента**

Как добавлять элемент в список? Для этого используется метод**add()**:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | public class Test {        public static void main(String[] args) {            ArrayList<String> myArrayList = new ArrayList<String>();          LinkedList<String> myLinkedList = new LinkedList<String>();            myArrayList.add("We add an element to ArrayList!");          myLinkedList.add("We add an element to LinkedList!");      }  } |

Кстати, мы можем добавлять в список не только элемент класса, который мы указали, но и всех его классов-наследников.

Например, представим, что у нас есть класс **Animal**, у которого есть наследник **Cat**:

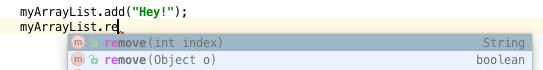
|  |  |
| --- | --- |
| 1  2  3 | class Cat extends Animal {      // ...  } |

Тогда, если указать как принимаемый тип **Animal**, а положить внутрь **Cat** - ничего плохого не произойдет:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | public class Test {        public static void main(String[] args) {            ArrayList<Animal> myArrayList = new ArrayList<Animal>();            Cat myCat = new Cat();          myArrayList.add(myCat);      }  } |

**Удаление элемента**

Удалить элемент можно двумя способами - по индексу и по ссылке:



Например, представим, что мы добавляем в наш **ArrayList**строчку **"Hey"**:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | public class Test {        public static void main(String[] args) {            ArrayList<String> myArrayList = new ArrayList<String>();            myArrayList.add("Hey!");        }  } |

Чтобы удалить ее, мы можем обратиться по индексу (индекс первого элемента - ноль):

**myArrayList.remove(0);**

или написать **"Hey"** как аргумент:

**myArrayList.remove("Hey!");**

И так, и так наш элемент будет удален.

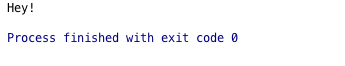
**Проход по списку**

Итак, давайте завершим наш урок проходом по списку. А то мы все добавляли, удаляли, а наглядно посмотреть работает оно или нет не могли

Давайте рассмотрим самый простой способ - пройтись по списку с помощью цикла **for**:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | public class Test {        public static void main(String[] args) {            ArrayList<String> myArrayList = new ArrayList<String>();            myArrayList.add("Hey!");            for (int i = 0; i < myArrayList.size(); i++)              System.out.println(myArrayList.get(i));      }  } |

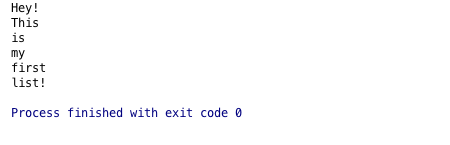
Здесь у нас в списке лежит только один элемент - строка "Hey". Давайте запустим:



Отлично! Добавим еще что-нибудь:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | public class Test {        public static void main(String[] args) {            ArrayList<String> myArrayList = new ArrayList<String>();            myArrayList.add("Hey!");          myArrayList.add("This");          myArrayList.add("is");          myArrayList.add("my");          myArrayList.add("first");          myArrayList.add("list!");            for (int i = 0; i < myArrayList.size(); i++)              System.out.println(myArrayList.get(i));      }  } |

Запускаем:



Замечательно - теперь удалим что-нибудь:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | public class Test {        public static void main(String[] args) {            ArrayList<String> myArrayList = new ArrayList<String>();            myArrayList.add("Hey!");          myArrayList.add("This");          myArrayList.add("is");          myArrayList.add("my");          myArrayList.add("first");          myArrayList.add("list!");            myArrayList.remove(0);          for (int i = 0; i < myArrayList.size(); i++)              System.out.println(myArrayList.get(i));      }  } |

Получим:

