public class ArrayList<E> extends AbstractList<E>  
 implements List<E>, RandomAccess, Cloneable, java.io.Serializable  
{

|  |  |
| --- | --- |
| **Методы *ArrayList*** | |
| aList.add("Sunday") | Добавление элемента |
| aList.add(int index, E element) | Добавление по индексу, со сдвигом остальных элементов вправо |
| aList.addAll(Collection <?extends E> c) | Добавляет список в конец списка. Элементы выстраиваются по порядку. |
| aList.addAll(int index, Collection <?extends E> c) | Добавляет коллекцию в список с указанного индекса, тем самым остальные элементы пополняемой коллекции сдвигая вправо, изменяя их индексы |
| aList.removeRange(int fromIndex, int toIndex) | Удаление элементов списка в диапазоне индексов (включительно) |
| aList.size() | Длина (размер) коллекции |
| aList.get(1) | Получить конкретный элемент по индексу |
| aList.get(0) | Получить первый элемент коллекции по индексу |
| aList.get(aList.size()-1) | Получить последний элемент коллекции по индексу |
| aList.clear() | Удалить все элементы из списка |
| aList.removeAll(aList) | Также удаляет все элементы из списка |
| aList.remove("Monday") | Удалить (объект) |
| aList.remove(1) | Удалить по индексу |
| Collections.sort(aList) | Отсортировать список |
| Collections.sort(aList,Collections.reverseOrder()) | Отсортировать в обратном порядке |
| aList.contains("Monday") | Проверить, существует ли значение. Возвращает true or false |
| aList.indexOf("Monday") | Проверить наличие объекта. Возвращает его индекс или -1 |
| aList.lastIndexOf("Monday") | Проверить наличие объекта. Возвращает его индекс или -1 |
| String[] arr = (String[])aList.toArray(new String[aList.size()]) | Преобразование списка в массив |
| ArrayList < String> aList= new ArrayList < String>(Arrays.asList(array)) | Преобразование массива в список |
| String result = String.join(",", aList) | Преобразовать список в строку |
| List list = aList | Преобразовать список в список |
| aList.isEmpty() | Проверить, пуст ли список. Возвращаете true or false |
| aList.set(1, newInt) | Элемент по индексу 1 будет заменен на значение newInt |
| Integer old = aList.set(1, newInt) | Элемент по индексу 1 будет заменен на значение newInt, а начальное значение присвоится ссылке old |
| aList.retainAll(Collection<?> c) | Оставляет в списке только те элементы, которые входят в список-аргумент |
| Void trimToSize() | Сделать размер массива списка соответствующим колличеству элементов в списке |
|  |  |

**Как добавить элементы в ArrayList?**

# Python

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | import Java.util.\*;  class TestClass  {  public static void main (String[] args) throws Java.lang.Exception  {     // создаем список массивов Object     ArrayList aList = new ArrayList();     aList.add("Sunday"); // добавление элемента     aList.add("Monday");     aList.add("Tuesday");  }  } |

**Как напечатать элемент arraylist?**

Python

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | // создаем список массивов Object  ArrayList aList = new ArrayList();  aList.add("Sunday");  aList.add("Monday");  aList.add("Tuesday");  Iterator ir=aList.iterator();    while(ir.hasNext()){  System.out.println(ir.next());  } |

# *Вывод:Python*

|  |  |
| --- | --- |
| 1  2  3 | Sunday  Monday  Tuesday |

**Как найти длину / размер ArrayList в Java?**

Используя метод size() класса ArrayList, мы можем легко определить размер ArrayList. Этот метод возвращает количество элементов в объекте ArrayList.Python

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | // создаем список массивов Object  ArrayList aList = new ArrayList();  aList.add("Sunday");  aList.add("Monday");  aList.add("Tuesday");  System.out.println(aList.size()); |

# *Вывод:*Python

|  |  |
| --- | --- |
| 1 | 3 |

**Как получить конкретный элемент ArrayList?**

Python

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | // создаем список массивов Object  ArrayList aList = new ArrayList();  aList.add("Sunday");  aList.add("Monday");  aList.add("Tuesday");  System.out.println(aList.get(1)); |

# *Вывод:*Python

|  |  |
| --- | --- |
| 1 | Monday |

Доступ к элементам в ArrayList можно получить с помощью целочисленного индекса, а индексы ArrayList начинаются с нуля. Таким образом, aList.get (1) возвращает второй элемент из ArrayList.

**Как получить первый элемент Arraylist?**

Python

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | // создаем список массивов Object  ArrayList aList = new ArrayList();  aList.add("Sunday");  aList.add("Monday");  aList.add("Tuesday");  System.out.println(aList.get(0)); |

# *Вывод:*Python

|  |  |
| --- | --- |
| 1 | Sunday |

Поскольку индексы ArrayList начинаются с нуля, aList.get (0) возвращает первый элемент ArrayList.

**Как получить последний элемент Arraylist?**

Python

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | // создаем список массивов Object  ArrayList aList = new ArrayList();  aList.add("Sunday");  aList.add("Monday");  aList.add("Tuesday");  System.out.println(aList.get(aList.size()-1)); |

AList.size() возвращает итоговые элементы в ArrayList. Поскольку индексы ArrayList начинаются с нуля, aList.size() - 1 возвращает последний элемент ArrayList.

**Как удалить все элементы из Java ArrayList?**

Вы можете использовать два разных метода для удаления arraylist в Java. Это ArrayList.clear() и ArrayList.removeAll()Python

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | //ArrayList.clear():  // создаем список массивов Object  ArrayList aList = new ArrayList();  aList.add("Sunday");  aList.add("Monday");  aList.add("Tuesday");  System.out.println(aList.size());  aList.clear();  System.out.println(aList.size()); |

# *Вывод:*Python

|  |  |
| --- | --- |
| 1  2 | 3  0 |

**Пример ArrayList.removeAll():**Python

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | // создаем список массивов Object  ArrayList aList = new ArrayList();  aList.add("Sunday");  aList.add("Monday");  aList.add("Tuesday");  System.out.println(aList.size());  aList.removeAll(aList);  System.out.println(aList.size()); |

# *Вывод:*Python

|  |  |
| --- | --- |
| 1  2 | 3  0 |

В случае removeAll вы должны передать тот же ArrayList, что и аргумент.

Методы **clear()** и **removeAll (коллекция)** служат в двух разных целях. Метод clear() будет проходить через лежащий в основе массив и установить каждую запись в null, а removeAll (collection) будет проходить проверку ArrayList для сбора и удаления (Object), если она существует. Таким образом, это подтверждает, что clear() выполняется намного быстрее, поскольку ему не нужно иметь дело со всеми этими вызовами дополнительных методов.

Как удалить определенный элемент из ArrayList? В общем случае объект может быть удален двумя способами из ArrayList (или вообще любого Списка), по индексу (remove (int)) и по объекту (remove (Object)).

**Удалить объект (удалить (объект)):**

Python

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | // создаем список массивов Object  ArrayList aList = new ArrayList();  aList.add("Sunday");  aList.add("Monday");  aList.add("Tuesday");  aList.remove("Monday"); |

Здесь мы передаем аргумент как строковый объект «Понедельник», поэтому он будет удален из коллекции.

**Удалить по индексу:**

Python

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | // создаем список массивов Object  ArrayList aList = new ArrayList();  aList.add("Sunday");  aList.add("Monday");  aList.add("Tuesday");  aList.remove(1); |

Здесь мы передаем аргумент как index remove (1) «Monday», поэтому он будет удален из коллекции.

**Как отсортировать ArrayList в Java?**

В ArrayList элементы помещаются по мере их вставки. Но при кодировании вы часто нуждаетесь в них в некотором порядке. для сортировки ArrayList используется метод sort() класса Collections.Python

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | // создаем список массивов Object  ArrayList aList = new ArrayList();  aList.add("Sunday");  aList.add("Monday");  aList.add("Tuesday");  System.out.println("Before sort.....");  Iterator ir=aList.iterator();  while(ir.hasNext()){    System.out.println(ir.next());  }  Collections.sort(aList);  System.out.println("After sort.....");  ir=aList.iterator();  while(ir.hasNext()){    System.out.println(ir.next());  } |

# *Вывод:*Python

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | Before sort.....  Sunday  Monday  Tuesday  After sort.....  Monday  Sunday  Tuesday |

**Сортировка ArrayList в обратном порядке?**

Python

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | // создаем список массивов Object  ArrayList aList = new ArrayList();  aList.add("Sunday");  aList.add("Monday");  aList.add("Tuesday");  Collections.sort(aList,Collections.reverseOrder());  Iterator ir=aList.iterator();  while(ir.hasNext()){    System.out.println(ir.next());  } |

# *Вывод:*Python

|  |  |
| --- | --- |
| 1  2  3 | Tuesday  Sunday  Monday |

**Поиск элемента в Java ArrayList**

Вы можете проверить, существует ли значение в Java ArrayList, используя следующие методы: ArrayList.contains(), ArrayList.indexOf() и ArrayList.lastIndexOf()Python

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25 | // создаем список массивов Object  ArrayList aList = new ArrayList();  aList.add("Sunday");  aList.add("Monday");  aList.add("Tuesday");  // Использование метода contains  if (aList.contains("Monday")) {    System.out.println("Account found");  } else {    System.out.println("Account not found");  }  // использование метода indexOf  int val = aList.indexOf("Monday");  if (val !=-1) {    System.out.println("Item found");  } else {    System.out.println("Item not found");  }  // использование метода lastIndexOf  int val = aList.lastIndexOf("Monday");  if (val !=-1) {    System.out.println("Item found");  } else {    System.out.println("Item not found");  } |

**Преобразование массива в массив**

Класс ArrayList имеет метод, называемый toArray(), который мы можем использовать для преобразования ArrayList в массивы.Python

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | // создаем список массивов Object  ArrayList aList = new ArrayList();  aList.add("Sunday");  aList.add("Monday");  aList.add("Tuesday");  String[] arr = (String[])aList.toArray(new String[aList.size()]);  for(String item : arr)    System.out.println(item); |

# *Вывод:*Python

|  |  |
| --- | --- |
| 1  2  3 | Sunday  Monday  Tuesday |

**Преобразование массива в ArrayList**

Python

|  |  |
| --- | --- |
| 1  2  3  4  5 | // создаем массив Object  String days[]={"Sunday", "Monday", "Tuesday"};  ArrayList < String> arrList= new ArrayList < String>(Arrays.asList(days));  for(String item : arrList)    System.out.println(item); |

# *Вывод:*Python

|  |  |
| --- | --- |
| 1  2  3 | Sunday  Monday  Tuesday |

**Преобразование ArrayList в строку**

Метод String join() поможет вам преобразовать ArrayList в String. Метод Java string join() возвращает строку, связанную с данным разделителем. В методе объединения строк для каждого элемента копируется разделитель.

Python

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | // создаем список массивов Object  ArrayList aList = new ArrayList();  aList.add("Sunday");  aList.add("Monday");  aList.add("Tuesday");  String result = String.join(",", aList);  System.out.println(result); |

# *Вывод:*Python

|  |  |
| --- | --- |
| 1 | Sunday,Monday,Tuesday |

**Преобразование ArrayList в список**

Python

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | // создаем список массивов Object  ArrayList aList = new ArrayList();  aList.add("Sunday");  aList.add("Monday");  aList.add("Tuesday");  List list = aList;  Iterator ir=list.iterator();  while(ir.hasNext()){    System.out.println(ir.next()); |

# *Вывод:*Python

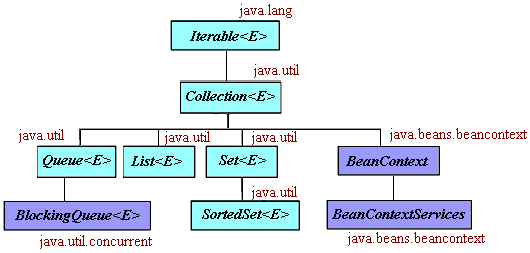
|  |  |
| --- | --- |
| 1  2  3 | Sunday  Monday  Tuesday |

Следующая программа Java иллюстрирует большинство вышеупомянутых методов в одной программеPython

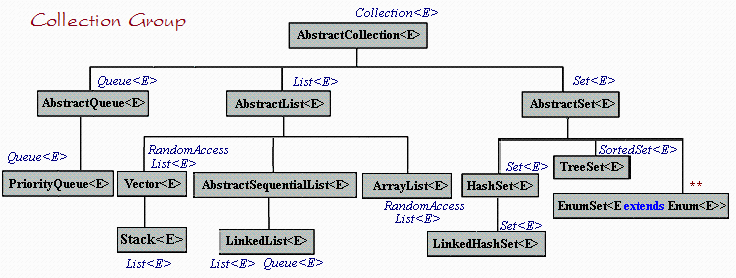
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37 | import Java.util.\*;  class TestClass  {    public static void main (String[] args) throws Java.lang.Exception    {      // создаем новый объект ArrayList        ArrayList aList = new ArrayList();        aList.add("Sunday");        aList.add("Monday");        aList.add("Wednesday");        // использование Iterator для просмотра всех элементов в ArrayList        Iterator < String> itr = aList.iterator();            while(itr.hasNext()){            System.out.println(itr.next());        }        // добавление элемента по указанному индексу        aList.add(2,"Tuesday");        System.out.println(aList);        // Извлечение элементов ArrayList по индексу        System.out.println(aList.get(2));        // Поиск элемента в ArrayList        if(aList.contains("Monday"))            System.out.println("Item Found");        else            System.out.println("Item not Found");        // Удаление одного элемента из ArrayList по индексу        aList.remove(1);        System.out.println(aList);        // Удаление всех элементов из ArrayList        aList.clear();        // Проверка вектора - это ArrayList или нет.        if(aList.isEmpty())            System.out.println("ArrayList Empty !!");        else            System.out.println("ArrayList Contains: " + aList.size() + " elements  !!");    }  } |

### java.util.Collection Interface

Наследственная связь:



### Наследственные отношения между классами в группе Collection



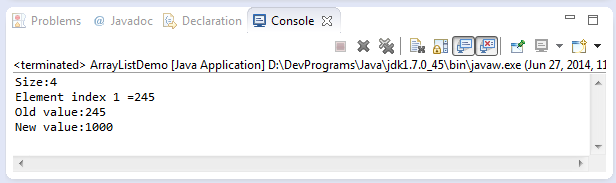
**ArrayList** имеет полные функции интерфейса  **List**. Одновременно, имеет возможность получать доступ к случайным элементам (Из-за наследства от интерфейса  **RandomAccess**)

Это в принципе похоже на класс  **Vector**, отличается в том, что метод  **Vector** синхронизирован, а **ArrayList** нет. **ArrayList** подходит для одноточного приложения (1 Thread).

**ArrayListDemo.java**

|  |
| --- |
| **package** org.o7planning.tutorial.javacollection.list;    **import** java.util.ArrayList;    **public** **class** ArrayListDemo {    **public** **static** **void** main(String[] args) {            // Объявить объект ArrayList содержаций элемент вида Integer.          ArrayList<Integer> list = **new** ArrayList<Integer>(10);            // Добавить элементы.          list.add(123);          list.add(245);          list.add(**new** Integer(345));            // ArrayList позволяет добавить элементы null .          // (Особенность List)          list.add(**null**);            // Распечатаь элементы ArrayList.          System.out.println("Size:" + list.size());// =4            // Получить случайный доступ в элемент индекса 1.          Integer i = list.get(1);          System.out.println("Element index 1 =" + i);// =245          Integer newInt = 1000;            // Заменить элемент в позиции с индексом 1.          // Этот метод возвращает старый элемент.          Integer old = list.set(1, newInt);          //          System.out.println("Old value:" + old);// =245 .          System.out.println("New value:" + list.get(1));// =1000 .      }    } |

Результаты запуска примера:



### Основные харрактеристики ArrayList

**ArrayList** – это динамичный массив, который по мере надобности в процессе работы увеличивает свой размер. При заполнении исходного массива создаётся новый массив большего размера (по формуле: (размер старого массива \* 3)/2+1 ) и в него копируются данные из старого массива.

**ArrayList** – это фактически, тот же массив так как внутренне данные хранятся как в обычном массиве.

**ArrayList** – при удалении объекта из списка все елементы справа от удаляемого объекта перемещаются на позицию влево, тем самым переписывая свои индексы. Фактическая длина массива при этом не меняется. Просто в хвосте образуются элементы типа null.

**ArrayList** – также при добавлении элемента в начало или в середине списка все элементы правее смещаются на позицию, что вызывает их переиндексацию.

**ArrayList** – позволяет хранить одинаковые элементы в своих ячейках. Т.е. информация может дублироваться.

**ArrayList** – это упорядоченная коллекция. Т.е. все элементы находятся в том поряке, в котором их туда поместили.

**ArrayList** - как и все другие виды коллекций из Framework Collection может хранить в себе только объекты и классы-оболочки для примитивов. Нельзя в него поместить примитивные типы данных.

**ArrayList** – позволяет хранить элементы null в любом колличестве.

### java.util.List Interface

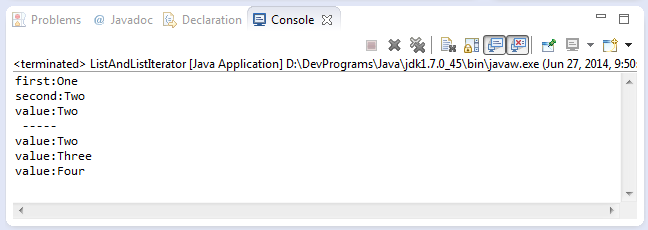
***List*** это подинтерфейс в  ***Collection*** который имеет полные функции  ***Collection*** .

Помимо доступа, используя  ***Iterator*** можно получить доступ используя  ***ListIterator***. ***ListIterator*** позволяет перемещать позицию курсора вперед или назад.

|  |
| --- |
| // Возвращает объект ListIterator, чтобы просмотреть элементы списка.  **public** ListIterator<E> listIterator()    // Возвращает объект ListIterator,  // для просмотра элементов с позиции с 'index' до конца списка.  **public** ListIterator<E> listIterator(**int** index) |

**ListAndListIterator.java**

Результаты запуска примера ниже:



|  |
| --- |
| **package** org.o7planning.tutorial.javacollection.list;    **import** java.util.ArrayList;  **import** java.util.List;  **import** java.util.ListIterator;    **public** **class** ListAndListIterator {    **public** **static** **void** main(String[] args) {            // Создать объект List (Содержащий только String).          List<String> list = **new** ArrayList<String>();          list.add("One");          list.add("Two");          list.add("Three");          list.add("Four");            // Получить объект ListIterator для просмотра элементов списка.          // (Theo đúng trình tự).          // Возвращает список iterator элементов данного списка          // (в определенном порядке)          ListIterator<String> listIterator = list.listIterator();            // В данный момент курсор стоит на первой позиции Iterator.          // Стоит в позиции с индексом 0.          // Получить первый элемент в Iterator, курсор движется вперед на один шаг.          String first = listIterator.next();          System.out.println("first:" + first);// -->"One"            // В данный момент курсор стоит в позиции с индексом 1.          // Получить следующий элемент.          String second = listIterator.next();          System.out.println("second:" + second);// -->"Two"            // Проверить может ли курсор перепрыгнуть назад на 1 позицию или нет.  **if** (listIterator.hasPrevious()) {              // Перепрыгнуть назад на 1 шаг.              String value = listIterator.previous();              System.out.println("value:" + value);// -->"Two"          }            System.out.println(" ----- ");    **while** (listIterator.hasNext()) {              String value = listIterator.next();              System.out.println("value:" + value);          }      }    } |

