【干货】用大白话聊聊JavaSE — ArrayList 深入剖析和Java基础知识详解（一）

java.util
Class ArrayList<E>

java.lang.Object
  └ java.util.AbstractCollection<E>
    └ java.util.AbstractList<E>
      └ **java.util.ArrayList<E>**

All Implemented Interfaces:
Serializable, Cloneable, Iterable<E>, Collection<E>, List<E>, RandomAccess

Direct Known Subclasses:
AttributeList, RoleList, RoleUnresolvedList

对Java程序开发而言，ArrayList 的使用频率是非常高的，尤其在进行 JavaWeb 开发的时候，ArrayList 和 HashMap 这两个类，相信你一定不会陌生，因为天天都在用嘛。

本系列对 ArrayList 做一个解析，同时把Java基础知识个串连进去。一开始我会对如何使用 ArrayList 做一个简要的说明，然后，我们来仿照 ArrayList 封装一个自己的集合框架MyList，通过练习，来一步一步猜想ArrayList 可能的实现方式。

最后，深入到ArrayList 的源码进行解读。

为什么要学习源码？

很简单，一个知道源码的人和一个不知道源码的人，虽然都能使用 ArrayList，但是，他们在使用的时候，心态是完全不一样的。

只有当你深入了源码，然后你才会对它的一些细节有更充分的认识。这是一本万利的事情。

当然，对于初学者，还是尽量以使用为主，因为源码的话，毕竟有一定的难度。如果一味地追求这些东西，可能会大大降低自己的学习兴趣和热情。

##1、 ArrayList 概述

副本难度：一颗星
经验值：500

首先来看一下文档，

All Implemented Interfaces:
Serializable, Cloneable, Iterable<E>, Collection<E>, List<E>, RandomAccess

All Implemented Interfaces: Serializable, Cloneable, Iterable

从图中可以看到，ArrayList 实现了Iterable接口，这个接口表示一种迭代的能力。

既然是ArrayList，那么肯定和 List 有关，所以它果然继承了List接口。

接口的知识点在这里就用上了。

其他接口我们暂且不谈，先继续。

然后，我们看一下官方对ArrayList 做出的说明。

文档中有这么一句：

Resizable-array implementation of the List interface.

这句话是说，ArrayList 是对List接口的一个实现，实现方式是利用一个可改变尺寸的数组，也就是说，它的底层就是一个数组。而且是可改变尺寸的数组，说明这个数组是动态的。

哦，难怪它叫ArrayList，Array就是数组的意思，那么它肯定和数组有关。

Implements all optional list operations, and permits all elements, including null.

ArrayList 实现了list接口的所有方法，并且允许空元素。

前半句是肯定的，因为在Java中，如果一个类实现了一个接口，那么就必须要重写该接口里所有的抽象方法。

我们知道，接口里只有方法的声明，没有方法的实现。

我对接口的理解，总结以后就只有一句话：

Java类实现接口，就是给这个类本身添加了一个新的身份。

ArrayList是一个类，那么它的身份就是ArrayList，你问他“你叫什么名字呀？”，他肯定会毫不犹豫地告诉你，“我叫ArrayList！”。

白天，我们大家都以为它是ArrayList，结果到了夜晚。

ArrayList竟然摇身一变，成为了Iterable！

原来，ArrayList实现了Iterable，所以它也就拥有了Iterable的身份。

接口其实就是这个意思。网上对接口的解释众说纷纭，反正我总结下来就是这么一句话，也不想搞那么复杂了。

ArrayList不仅有Iterable的身份，还拥有其他好几种身份，比如List，Collection等。

这个情况也叫作多态。

新建一个测试类。

```
public class TestArrayList1 {

    public static void main(String[] args) {

    }
}
```

现在我新建一个ArrayList类的实例

```
ArrayList list = new ArrayList();
```

这肯定是没问题的。

文档上说，ArrayList 实现了Iterable接口，那么也就是说，它的另一个身份是Iterable。

所以，我这样写是不是也没有问题啊。

```
Iterable iterable = new ArrayList();
```

就好像白天是普通的上班族，一旦到了夜晚，就。。。

总之，

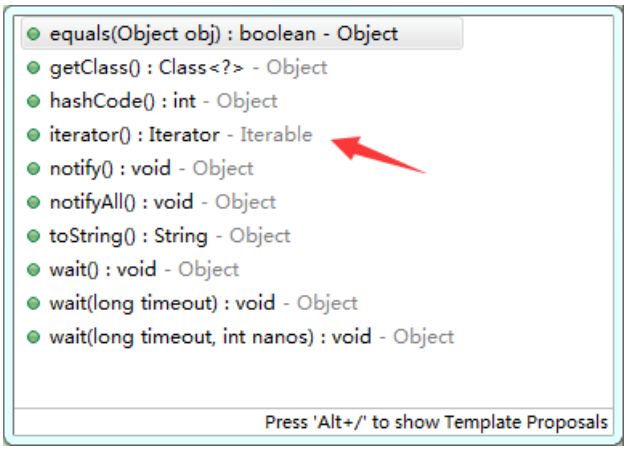
Iterable iterable = new ArrayList();

这句话的含义就是说，它原来是普通的ArrayList，一旦情况需要，就变身成为Iterable。

好了，既然变成了 Iterable，那么它是不是也就拥有了 Iterable 的能力？

我们来看一下 Iterable 有哪些方法？

去除继承自Object类的方法不谈，Iterable 只有一个方法，就是 **iterator()**



它返回的是一个 Iterator，这是一个迭代器。

通过这个迭代器，我们是不是就可以遍历 ArrayList 中所有的数据了呀？

首先，我们需要一个迭代对象：

```
Iterable iterable = new ArrayList();
Iterator it = iterable.iterator();
```

当然，这个例子中，ArrayList里面没有数据。

现在思考一个问题，是不是我非得把 ArrayList 对象改变成 Iterable 身份，才可以调用 iterator() 方法呢？

当然不是了，ArrayList 有一个身份是 Iterable，所以它具有 Iterable 的所有能力（方法），这没问题。那么难道 ArrayList 不变身，就没有 Iterable 的能力了吗？

答案自然是否定的。

一个小说家如果有一天转行去写代码了，那么你觉得他还会不会写小说呢？

肯定会嘛，这没有什么好怀疑的。

所以，代码这么写是不是也没关系？

```
ArrayList list = new ArrayList();

for (int i = 0; i < 100; i++) {
    list.add(i);
}

Iterator it = list.iterator();
```

循环输出

```
while(it.hasNext()){
    System.out.println(it.next());
}
```

```
87
88
89
90
91
92
93
94
95
96
97
98
99
```

总结一下，ArrayList 它只要实现 Iterable 接口，那么它就必须要拥有 Iterable 规定的所有能力，也就是方法。而接口中，我们知

道它里面只有方法的声明，没有方法的实现，所以 ArrayList 需要实现这些方法，实现接口就是这么个意思。

##2、 ArrayList常用方法

副本难度：三颗星
经验值：800

2.1、 属性和方法的调用问题

在调用ArrayList的方法之前，我们需要先获得一下ArrayList的实例对象，除了静态方法，其他所有的方法，都只能由对象来调用。

ArrayList是一个类，我更愿意把它称为一个** 数据模板**。它只是一个模板而已，不是一个实实在在的对象，这一点首先要确定。

就好像工厂生产一个产品，首先是不是要有一个模板和设计图纸，这个模板决定了你这个产品是一种怎样的形状，以及可能会具备哪些功能？图纸则决定了功能的具体实现。

比如生成一部手机，模板开出来就是一个扁平的长方体的样子，可是光有模板还不行，你还得规定它的一些具体细节。

这些细节就好比是Java类的构造方法，以及其他的一些方法实现。

但是，你光给客户模板和图纸行吗？

一般来说是不行的。

至于静态方法，我们知道，我们调用静态方法的时候，不需要先生成一个实例，可以通过类名直接调用。

这就相当于，在弄模版的时候，这些功能就已经定制在里面了。

你买手机的时候，里面不是经常有一些内置的应用吗？有些删都删不掉，**这不就相当于静态方法**吗？

（我只是举一个例子啊，你不要非得较真说我可以ROOT一下啊）

如果模板里面已经有了一些做好的功能，今后任何根据这个模具生成出来的产品也自带了这些功能。

如果模板里面已经做好了一些功能，那么我的确可以使用这个模板，而不需要真正拿到一个产品。

比如生产一部手机，它的模板里面已经做好了一个手电筒的功能，那么，你即便不给我一个真正的产品，仅仅给我一个模板，我是不是也可以用它的手电筒功能呢？

这就是静态方法。

所以我们常说，静态方法和静态属性为所有实例共用，不就是这个道理吗？

所以，正常情况下，我们调用一个类的非静态方法，是不是必须要先new一个对象？

好的，我们现在来 **根据 ArrayList 模板生产一个 ArrayList 产品**。

这样，我们才能调用它里面所有的非私有方法。

怎么生产呢，是不是new一下就可以啦？

```
ArrayList  arrayList = new ArrayList();
```

###2.2、 add方法
ArrayList是一个集合，既然是一个集合，那么它肯定是可以往里头添东西的。

怎么往里面添，用add，用add方法往里面加。

```
arrayList.add("HelloWorld"); //添加一个字符串
arrayList.add(new Integer(100));//添加一个Integer类型的数字
```

add的参数就是一个object，这就是多态，多态就是多种形态，**多种身份**的意思。object可以有多种形态。它可以是String，也可以是Integer，还可以是用户自定义的类型。

这里是不是又多态了。。。

这就是多态的一个用法。

###2.3、 get 方法

既然能够往里面添加东西，是不是肯定还要拿出来啊。

怎么拿出来，用get方法拿出来，而且一次只能拿一个。不要多拿哦。

get 方法需要传入一个 int 类型的数字，这个数字就是元素对应的下标。

我们刚才第一个放进去的是 "HelloWorld"，一个字符串。那么对应的下标就是0 。

第二个放进去的是new Integer(100)，这是一个Integer对象，是一个实实在在的东西了。那么对应的下标就是1 。

现在，我取出第1个元素，应该是100。

试试看

```
Integer i = arrayList.get(1);
```

报错了，因为 get 方法返回的是一个Object对象，而我们拿Integer 去接，就出问题了。

这是咋回事呢？

很简单，比如张三是一个医生，同时他还拥有一个人类的身份，可并不是所有的人类都是医生啊？

注意我下面分析的用词，能帮助你理解。

同理，Integer 是一个整数类型，同时它还拥有一个Object的身份，可并不是所有的 Object 都是Integer 啊？

同接口一样，A继承B，可以看成A同时拥有了B的身份。

如果A继续保持A的身份，那么它不仅拥有自己本身的能力，也拥有B的能力。只要A愿意，他完全可以展现B的能力。

如果A变身成为了B，那么它肯定不希望别人知道他是A。

就好比虽然他知道他可以变身成为奥特曼，但是一般情况下，他都不愿意曝露自己的身份。

所以如果A变身成为了B，那么A就只会使用B的能力，同时隐藏他本身的能力。

如果你是初学者，请仔细体会一下这其中的韵味。慢慢地，你就会对多态有一个更深入的理解。一段时间后，你再重新去看以前写的代码，会有不一样的感觉。

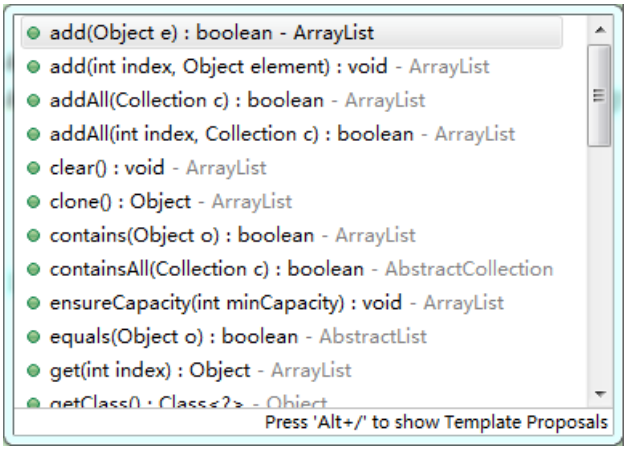
再举一个例子，帮助你理解。

我 new 一个 ArrayList：（在eclipse中）

```
ArrayList  arrayList = new ArrayList();
```

换行，写上arrayList 。

我直接在arrayList 右边加一个点，然后会有提示：

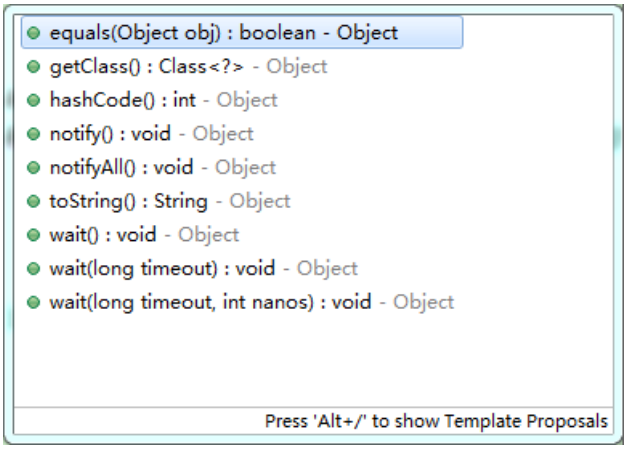


这些都是它可以调用的方法和属性，哇，这么多。

如果我这样写呢？

```
Object  arrayList = new ArrayList();
```

我也在arrayList 右边加一个点



不好意思，你现在只能调用 Object 类的属性和方法了。

嗯，再体会一下。

初学者在面向对象方面的理解总是会走弯路，如果你能把这些东西理清，对今后的学习会有巨大的好处。

继续。

我们这里就强转一下吧，因为我们知道 index 为1的元素是一个Integer类型的。

```
Integer i = (Integer) arrayList.get(1);
System.out.println(i);
```

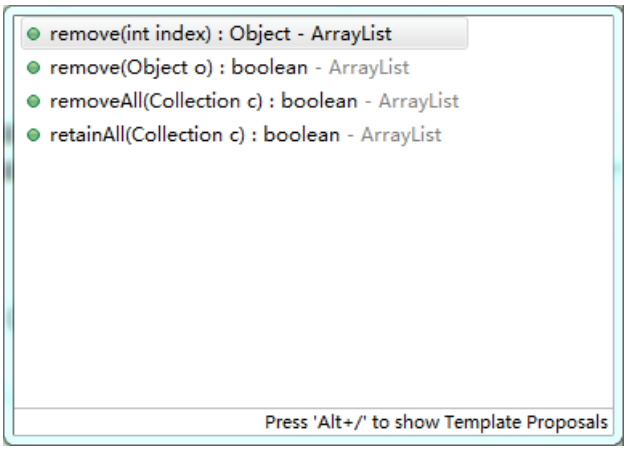
结果

```
100
```

好的，是正确的。

###2.4、 remove 方法

remove 方法可以删除集合中的元素，ArrayList给我们提供了很多删除元素的方法。



我们这里先看一下第一个方法。

这是通过数组下标来删除某一个特定的元素，我们刚才给ArrayList添加了两个元素，下标分别为 0,1 ，那么，如果我删除第0个元素，会怎么样呢？

首先，ArrayList的列表长度会不会改变？

我们可以通过size()方法来获取ArrayList当前的列表长度。

测试：

```
arrayList.remove(0);
System.out.println(arrayList.size());
```

打印出来是1，看来的确是影响长度了。

###2.5、 toArray 方法

这个方法可以将ArrayList转换成一个Object数组。

例：

```
Object[] objs = arrayList.toArray();

for (int i = 0; i < objs.length; i++) {
    System.out.println(objs[i].toString());
}
```

需要注意的是，哪怕你给ArrayList全部添加Integer类型的元素，也不能采用这样的代码：

```
Integer[] objs = (Integer[]) arrayList.toArray();
```

这是错误的，虽然编译的时候不会报错，但是运行无法通过。

因为ArrayList的add方法可以添加任意类型的参数，Java运行机制无法获知ArrayList中的元素是否可以都强制转换为你指定的类型。所以这种写法是不被允许的。