联系我们

请扫描二维码联系客服
webmaster@csdn.net
400-660-0108
QQ客服 客服论坛

关于 招聘 广告服务 百度
©1999-2018 CSDN版权所有
京ICP证09002463号

经营性网站备案信息
网络110报警服务
中国互联网举报中心
北京互联网违法和不良信息举报中心

等级： 博客 7 访问量： 162万+
积分： 1万+ 排名： 685

ng
输出

目录视图 摘要视图 RSS 订阅

Java 集合深入理解（4）：List<E> 接口

标签： java 集合

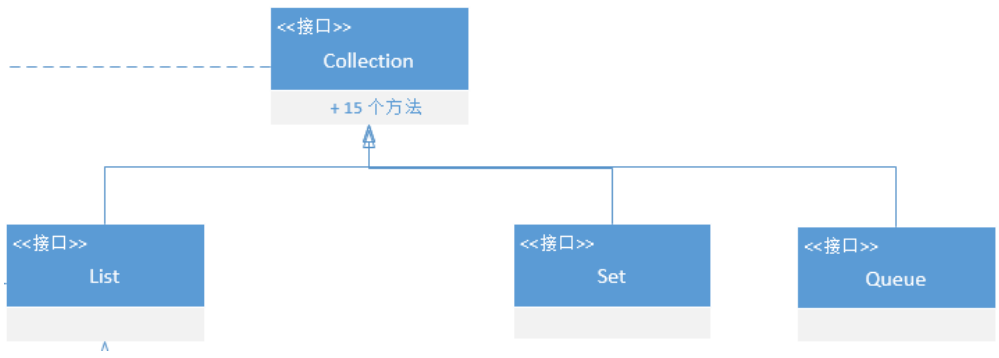
2016年10月13日 01:34:49 30592人阅读 评论(6) 收藏 举报

分类： Java 集合框架（17）

版权声明：转载前请留言获得作者许可，转载后标明作者 张拭心 与 原文链接。大家都是成年人，创作不易，感谢您的支持！
https://blog.csdn.net/u011240877/article/details/52802849

点击查看 Java 集合框架深入理解 系列，-(゜-゜)つロ 乾杯~

> 蓝瘦！香菇！ 连着加班几天，醉了。学学 List 放松下！



在 Java 集合深入理解：Collection 中我们熟悉了 Java 集合框架的基本概念和优点，也了解了根接口之一的 Collection，这篇文章来加深 Collection 的子接口之一 List 的熟悉。

java.util

Interface List<E>

Type Parameters:

E - the type of elements in this list

All Superinterfaces:

Collection<E>, Iterable<E>

List 接口

一个 List 是一个元素有序的、可以重复、可以为 null 的集合（有时候我们也叫它“序列”）。

Java 集合框架中最常使用的几种 List 实现类是 ArrayList，LinkedList 和 Vector。在各种 List 中，最好的做法是以 ArrayList 作为默认选择。 当插入、删除频繁时，使用 LinkedList，Vector 总是比 ArrayList 慢，所以要尽量避免使用它，具体实现后续文章介绍。

为什么 List 中的元素 “有序”、“可以重复”呢?

首先，List 的数据结构就是一个序列，存储内容时直接在内存中间，然后将空间地址与索引对应。

其次根据官方文档 ：

欢迎微信关注安卓进化论

Android 性能优化：使用 Lint... (33223)

联系我们

请扫描二维码联系客服

webmaster@csdn.net

400-660-0108

QQ客服 客服论坛

关于 招聘 广告服务 百度

©1999-2018 CSDN版权所有

京ICP证09002463号

经营性网站备案信息

网络110报警服务

中国互联网举报中心

北京互联网违法和不良信息举报中心

谷海涛：感谢分享！

并发编程3：线程池的使用与执行流程
梁山boy ：大佬，这个吞吐量不是很理解，如何去比较LinkedBlockingQueue、Synchronou...

Android 框架学习4：一次读...
梁山boy ：[reply]u011240877[/reply] 还是我看漏了？在哪个类里面啊？我看的不是rel...

Android 框架学习4：一次读...
拭心 ：[reply]a153614131[/reply] 我这个分析就是基于目前最新的 release ...

重温数据结构：哈希 哈希函数 哈希表
qq_36993025 ：讲的很详细，让我又多了解点哈希，多谢大佬，但我用的不多，不够熟练。

Android 框架学习4：一次读...
梁山boy ：看了下最新的代码里，根据网络状态调整线程数 adjustThreadCount() 好像已经没有了

Android 进阶15：Hand...
geelaro ：和你的代码一样，点了开始后，DownloadThread只启动了onLooperPrepared(...

Android 框架学习4：一次读...
qq_31970041 ：Mua

一句话解决RecyclerView...
AM小可乐 ：貌似并不是这样子的

文章存档

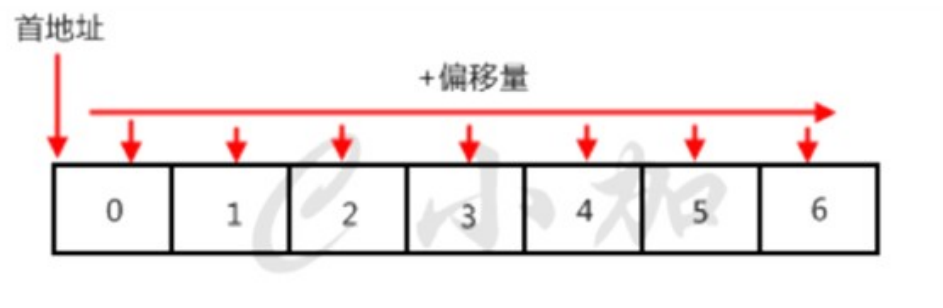
2018年3月 (1)
2018年1月 (1)
2017年11月 (2)
2017年10月 (1)
2017年9月 (2)

展开

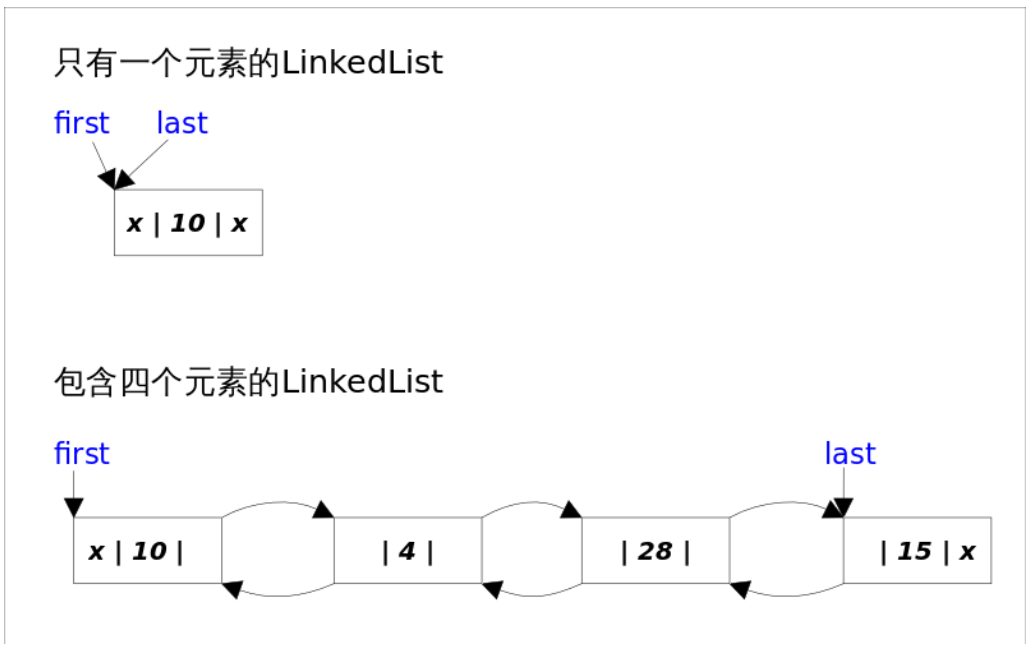
> The user of this interface has precise control over where in the list each element is inserted. The user can access elements by their integer index (position in the list), and search for elements in the list.

可以看到，List 接口的实现类在实现插入元素时，都会根据索引进行排列。

比如 ArrayList，本质是一个数组：



LinkedList, 双向链表：



由于 List 的元素在存储时互不干扰，没有什么依赖关系，自然可以重复（这点与 Set 有很大区别）。

List 接口定义的方法

List 中除了继承 Collection 的一些方法，还提供以下操作：

- 位置相关：`List` 和 数组一样，都是从 0 开始，我们可以根据元素在 list 中的位置进行操作，比如说 `get`，`set`，`add`，`addAll`，`remove`；
- 搜索：从 list 中查找某个对象的位置，比如 `indexOf`，`lastIndexOf`；
- 迭代：使用 Iterator 的拓展版迭代器 ListIterator 进行迭代操作;
- 范围性操作：使用 `subList` 方法对 list 进行任意范围的操作。

Collection 中 提供的一些方法就不介绍了，不熟悉的可以去看一下。

集合的操作

- `remove(Object)`
  - 用于删除 list 中头回出现的 指定对象；
- `add(E)`，`addAll(Collection<? extends E>)`
  - 用于把新元素添加到 list 的尾部，下面这段语句使得 list3 等于 list1 与 list2 组合起来的内容：

    ```
    List list3 = new ArrayList(list1);
    list3.addAll(list2);
    ```

  注意：上述使用了 `ArrayList` 的转换构造函数:

  public ArrayList(Collection

`Object` 的 `equlas`() 方法默认和 `==` 一样，比较的是地址是否相

欢迎微信关注安卓进化论

联系我们

请扫描二维码联系客服

webmaster@csdn.net

400-660-0108

QQ客服 客服论坛

关于 招聘 广告服务 百度

©1999-2018 CSDN版权所有

京ICP证09002463号

经营性网站备案信息

网络110报警服务

中国互联网举报中心

北京互联网违法和不良信息举报中心

```
public boolean equals(Object o) {
    return this == o;
}
```

因此和 Set，Map 一样，List 中如果想要根据两个对象的内容而不是地址比较是否相等时，需要重写 `equals()` 和 `hashCode()` 方法。 `remove()`，`contains()`，`indexOf()` 等等方法都需要依赖它们：

```
@Override
public boolean contains(Object object) {
    Object[] a = array;
    int s = size;
    if (object != null) {
        for (int i = 0; i < s; i++) {
            //需要重载 Object 默认的 equals
            if (object.equals(a[i])) {
                return true;
            }
        }
    } else {
        for (int i = 0; i < s; i++) {
            if (a[i] == null) {
                return true;
            }
        }
    }
    return false;
}

@Override
 public int indexOf(Object object) {
    Object[] a = array;
    int s = size;
    if (object != null) {
        for (int i = 0; i < s; i++) {
            if (object.equals(a[i])) {
                return i;
            }
        }
    } else {
        for (int i = 0; i < s; i++) {
            if (a[i] == null) {
                return i;
            }
        }
    }
    return -1;
}
```

两个 List 对象的所有位置上元素都一样才能相等。

位置访问，搜索

基础的位置访问操作方法有：

- `get`，`set`，`add`，`remove`
  - set, remove 方法返回的是 **被覆盖** 或者 **被删除** 的元素；
- `indexOf`，`lastIndexOf`
  - 返回指定元素在 list 中的首次出现/最后一次出现的位置（获取 lastIndexOf 是通过倒序遍历查找）；
- `addAll(int,Collection)`
  - 在特定位置插入指定集合的所有元素。这些元素按照迭代器 Iterator 返回的先后顺序进行插入；

下面是一个简单的 List 中的元素交换方法：

```
public static <E> void swap(List<E> a, int i, int j) {
    E tmp = a.get(i);
    a.set(i, a.get(j));
    a.set(j, tmp);
}
```

不同的是它是多态的，允许任何 List 的子类使用。 Collections 中的 shuffle 就有用到
法：

欢迎微信关注安卓进化论

联系我们

请扫描二维码联系客服

webmaster@csdn.net

400-660-0108

QQ客服 客服论坛

关于 招聘 广告服务 百度

©1999-2018 CSDN版权所有

京ICP证09002463号

经营性网站备案信息

网络110报警服务

中国互联网举报中心

北京互联网违法和不良信息举报中心

```
public static void shuffle(List<?> list, Random rnd) {
    for (int i = list.size(); i > 1; i--)
        swap(list, i - 1, rnd.nextInt(i));
}
```

这种算法使用指定的随机算法，从后往前重复的进行交换。和一些其他底层 shuffle 算法不同，这个算法更加公平（随机方法够随机的话，所有元素的被抽到的概率一样），同时够快（只要 `list.size() -1` ）次交换。

局部范围操作

List.subList(int fromIndex, int toIndex) 方法返回 List 在 fromIndex 与 toIndex 范围内的子集。注意是左闭右开，[fromIndex,toIndex)。

注意！ `List.subList` 方法并没有像我们想的那样：创建一个新的 List，然后把旧 List 的指定范围子元素拷贝进新 List，根！本！不！是！
subList 返回的扔是 List 原来的引用，只不过把开始位置 offset 和 size 改了下，见 List.subList() 在 AbstractList 抽象类中的实现：

```
public List<E> subList(int start, int end) {
    if (start >= 0 && end <= size()) {
        if (start <= end) {
            if (this instanceof RandomAccess) {
                return new SubAbstractListRandomAccess<E>(this, start, end);
            }
            return new SubAbstractList<E>(this, start, end);
        }
        throw new IllegalArgumentException();
    }
    throw new IndexOutOfBoundsException();
}
```

SubAbstractListRandomAccess 最终也是继承 SubAbstractList,直接看 SubAbstractList：

```
    SubAbstractList(AbstractList<E> list, int start, int end) {
        fullList = list;
        modCount = fullList.modCount;
        offset = start;
        size = end - start;
    }
```

可以看到，的确是保持原来的引用。

所以，重点来了！

由于 subList 持有 List 同一个引用，所以对 subList 进行的操作也会影响到原有 List，举个栗子：

```
List list = new ArrayList();
list.add("a");
list.add("b");
list.add("c");
list.add("d");

List subList = list.subList(0, 2);
subList.clear();
System.out.println(list);
```

你猜运行结果是什么？

```
"C:\Program Fi
[c, d]
```

验证了上述重点。

所以，我们可以使用 subList 对 List 进行范围操作，比如下面的代码，一句话实现了删除 shixinList 部分元素的操作：

```
shixinList.subList(fromIndex, toIndex).clear();
```

还可以查找某元素在局部范围内的位置：

```
int i = list.subList(fromIndex, toIndex).indexOf(o);
int j = list.subList(fromIndex, toIndex).lastIndexOf(o);
```

欢迎微信关注安卓进化论

联系我们

请扫描二维码联系客服

webmaster@csdn.net

400-660-0108

QQ客服 客服论坛

关于 招聘 广告服务 *百度*

©1999-2018 CSDN版权所有

京ICP证09002463号

经营性网站备案信息

网络110报警服务

中国互联网举报中心

北京互联网违法和不良信息举报中心

List 与 Array 区别？

List 在很多方面跟 Array 数组感觉很相似，尤其是 ArrayList，那 List 和数组究竟哪个更好呢？

- 相似之处：
  - 都可以表示一组同类型的对象
  - 都使用下标进行索引
- 不同之处：
  - 数组可以存任何类型元素
  - List 不可以存基本数据类型，必须要包装
  - 数组容量固定不可改变；List 容量可动态增长
  - 数组效率高; List 由于要维护额外内容，效率相对低一些

容量固定时优先使用数组，容纳类型更多，更高效。

在容量不确定的情景下， List 更有优势，看下 ArrayList 和 LinkedList 如何实现容量动态增长：

ArrayList 的扩容机制:

```
public boolean add(E object) {
    Object[] a = array;
    int s = size;
    //当放满时，扩容
    if (s == a.length) {
        //MIN_CAPACITY_INCREMENT 为常量，12
        Object[] newArray = new Object[s +
                (s < (MIN_CAPACITY_INCREMENT / 2) ?
                 MIN_CAPACITY_INCREMENT : s >> 1)];
        System.arraycopy(a, 0, newArray, 0, s);
        array = a = newArray;
    }
    a[s] = object;
    size = s + 1;
    modCount++;
    return true;
}
```

可以看到：

- 当 ArrayList 的元素个数小于 6 时，容量达到最大时，元素容量会扩增 12；
- 反之，增加 当前元素个数的一半。

LinkedList 的扩容机制：

```
public boolean add(E object) {
    return addLastImpl(object);
}

private boolean addLastImpl(E object) {
    Link<E> oldLast = voidLink.previous;
    Link<E> newLink = new Link<E>(object, oldLast, voidLink);
    voidLink.previous = newLink;
    oldLast.next = newLink;
    size++;
    modCount++;
    return true;
}
```

可以看到，没！有！扩容机制！

这是由于 LinedList 实际上是一个双向链表，不存在元素个数限制，使劲加就行

```
transient Link<E> voidLink;

private static final class Link<ET> {
    ET data;

    Link<ET> previous, next;
```

欢迎微信关注安卓进化论

联系我们

请扫描二维码联系客服

webmaster@csdn.net

400-660-0108

QQ客服 客服论坛

关于 招聘 广告服务 百度

©1999-2018 CSDN版权所有

京ICP证09002463号

经营性网站备案信息

网络110报警服务

中国互联网举报中心

北京互联网违法和不良信息举报中心

```
    Link(ET o, Link<ET> p, Link<ET> n) {
        data = o;
        previous = p;
        next = n;
    }
}
```

List 与 Array 之间的转换

在 List 中有两个转换成 数组 的方法：

- Object[] toArray()
  - 返回一个包含 List 中所有元素的数组；
- T[] toArray(T[] array)
  - 作用同上，不同的是当 参数 array 的长度比 List 的元素大时，会使用参数 array 保存 List 中的元素；否则会创建一个新的 数组存放 List 中的所有元素；

ArrayList 中的实现：

```
public Object[] toArray() {
    int s = size;
    Object[] result = new Object[s];
    //这里的 array 就是 ArrayList 的底层实现，直接拷贝
    //System.arraycopy 是底层方法，效率很高
    System.arraycopy(array, 0, result, 0, s);
    return result;
}

public <T> T[] toArray(T[] contents) {
    int s = size;
    //先判断参数能不能放下这么多元素
    if (contents.length < s) {
        //放不下就创建个新数组
        @SuppressWarnings("unchecked") T[] newArray
            = (T[]) Array.newInstance(contents.getClass().getComponentType(), s);
        contents = newArray;
    }
    System.arraycopy(this.array, 0, contents, 0, s);
    if (contents.length > s) {
        contents[s] = null;
    }
    return contents;
}
```

LinkedList 的实现：

```
public Object[] toArray() {
    int index = 0;
    Object[] contents = new Object[size];
    Link<E> link = voidLink.next;
    while (link != voidLink) {
        //挨个赋值，效率不如 ArrayList
        contents[index++] = link.data;
        link = link.next;
    }
    return contents;
}

@Override
@SuppressWarnings("unchecked")
public <T> T[] toArray(T[] contents) {
    int index = 0;
    if (size > contents.length) {
        Class<?> ct = contents.getClass().getComponentType();
        contents = (T[]) Array.newInstance(ct, size);
    }
    Link<E> link = voidLink.next;
    while (link != voidLink) {
        //还是比 ArrayList 慢
        contents[index++] = (T) link.data;
        link = link.next;
    }
    if (index < contents.length) {
        contents[index] = null;
```

欢迎微信关注安卓进化论

联系我们

请扫描二维码联系客服

webmaster@csdn.net

400-660-0108

QQ客服 客服论坛

关于 招聘 广告服务 百度

©1999-2018 CSDN版权所有

京ICP证09002463号

经营性网站备案信息

网络110报警服务

中国互联网举报中心

北京互联网违法和不良信息举报中心

```
    }
    return contents;
}
```

数组工具类 Arrays 提供了数组转成 List 的方法 `asList`：

```
@SafeVarargs
public static <T> List<T> asList(T... array) {
    return new ArrayList<T>(array);
}
```

使用的是 Arrays 内部创建的 ArrayList 的转换构造函数：

```
    private final E[] a;
    ArrayList(E[] storage) {
        if (storage == null) {
            throw new NullPointerException("storage == null");
        }
        //直接复制
        a = storage;
    }
```

迭代器 Iterator, ListIterator

List 继承了 Collection 的 iterator() 方法，可以获取 Iterator，使用它可以进行向后遍历。

在此基础上，List 还可以通过 listIterator(), listIterator(int location) 方法（后者指定了游标的位置）获取更强大的迭代器 ListIterator。

使用 ListIterator 可以对 List 进行向前、向后双向遍历，同时还允许进行 add, set, remove 等操作。

List 的实现类中许多方法都使用了 ListIterator，比如 List.indexOf() 方法的一种实现：

```
public int indexOf(E e) {
    for (ListIterator<E> it = listIterator(); it.hasNext(); )
        if (e == null ? it.next() == null : e.equals(it.next()))
            return it.previousIndex();
    // Element not found
    return -1;
}
```

ListIterator 提供了 add, set, remove 操作，他们都是对迭代器刚通过 next(), previous()方法迭代的元素进行操作。下面这个栗子中，List 通过结合 ListIterator 使用，可以实现一个多态的方法，对所有 List 的实现类都适用：

```
public static <E> void replace(List<E> list, E val, E newVal) {
    for (ListIterator<E> it = list.listIterator(); it.hasNext(); )
        if (val == null ? it.next() == null : val.equals(it.next()))
            it.set(newVal);
}
```

List 的相关算法：

集合的工具类 Collections 中包含很多 List 的相关操作算法：

- sort ，归并排序
- shuffle ，随机打乱
- reverse ，反转元素顺序
- swap ，交换
- binarySearch ，二分查找
- ……

具体实现我们后续介绍，感谢关注！

关联：

Collection, ListIterator, Collections

欢迎微信关注安卓进化论

联系我们

请扫描二维码联系客服

webmaster@csdn.net

400-660-0108

QQ客服 客服论坛

关于 招聘 广告服务 百度

©1999-2018 CSDN版权所有

京ICP证09002463号

经营性网站备案信息

网络110报警服务

中国互联网举报中心

北京互联网违法和不良信息举报中心

Thanks:

http://docs.oracle.com/javase/1.5.0/docs/api/java/util/List.html

https://docs.oracle.com/javase/tutorial/collections/interfaces/list.html

http://blog.csdn.net/mazhimazh/article/details/17759579#comments

http://www.blogjava.net/flysky19/articles/92775.html

- 上一篇 Java 集合深入理解（3）：Collection
- 下一篇 工作第十三周：身体掏空，精神饱满

您还没有登录,请[登录]或[注册]

查看评论

spring_beans 4楼 2018-03-23 21:02发表

当插入、删除频繁时，使用 LinkedList，Vector 总是比 ArrayList 慢？
是这样吗？

擦肩的阳光 Re: 2018-03-24 15:38发表

回复spring_beans：不是吧……LinkedList肯定比ArrayList快的

查看更多评论

Java学习笔记----------集合Set

hsj880921 2013年11月13日 14:41 17284

Java集合-----Set 集合：就像一种容器，可以把多个对象放进该容器中。 Java集合分为：Set、List、Map三种体系。 Set:无序的，不可重复的； List:有序的，可重复的；...

JAVA中list的用法

sisong 2008年02月27日 18:24 18310

java中List的通俗介绍和简单实例 List 经常用在对一组对象的存储和操作上，比如一组学生信息，一组帐号信息等等。 List是个集合接口，只要是集合类接口都会有个“迭代子”（ Iterator ...

List Control控件中及时捕获checkbox被选中的消息的解决方案

我的功能需求是：用户可以在List Control里添加item，当无选项被选中（即Checkbox没被选中），删除按钮不可用；当用户点击checkbox选中item时，删除按钮可用。 图示如下： ...

qiuyeting 2012年03月12日 15:31 4394

C++ STL 迭代器删除问题总结

ghosc 2011年11月24日 16:20 9180

iterator erase(iterator pos); 对C++ STL中迭代器的删除需要慎重，如上方法。 1、vector vector在遍历中进行 erase(pos)，这个时候i...

对List集合中的元素进行排序

veryisjava 2016年06月14日 22:08 138868

Collections对List集合中的数据进行排序 有时候需要对集合中的元素按照一定的规则进行排序，这就需要用到 Java中提供的对集合进行操作的工具类Collections，其中的so...

C++学习：list容器详解（一）

hao95165 2016年

C++学习：list容器详解（一）标签： listc++iteratorstring算法vector 2009-11-12 14:54 499...

Gson解析之字段名和实体类属性名不一致的处理

plq690816 2016年

当接口返回的json字段名和实体类的属性名不一致时的处理：示例json:public static final String jso
\",\"isbn...

欢迎微信关注安卓进化论

联系我们

请扫描二维码联系客服

webmaster@csdn.net

400-660-0108

QQ客服 客服论坛

关于 招聘 广告服务 百度

©1999-2018 CSDN版权所有

京ICP证09002463号

经营性网站备案信息

网络110报警服务

中国互联网举报中心

北京互联网违法和不良信息举报中心

C++ 字节对齐的总结(原因和作用)

sweetfather 2017年11月09日 11:21 234

一、介绍 什么是字节对齐 现代计算机中内存空间都是按照byte划分的，从理论上讲似乎对任何类型的变量的访问可以从任何地址开始，但实际情况是在访问特定类型变量的时候经常在特定的内存地址访问...

linux系统编程：线程同步-条件变量(cond)

zhangxiangDavaid 2015年07月26日 11:29 1157

线程同步-条件变量 生产者与消费者问题 再引入条件变量之前，我们先看下生产者和消费者问题：生产者不断地生产产品，同时消费者不断地在消费产品。 这个问题的同步在于两处：第一，消费者之间需要同步：同一件...

C#语法(List、方法的调用)

u014424282 2017年06月16日 11:54 573

class GroupSample1 { // The element type of the data source. public class Student { ...

劲爆！加拿大移民新政重开！

2018年加拿大移民门槛再创新低！

Java中List和ArrayList的区别(加入了个人见解)

erlian1992 2016年05月02日 22:25 23970

转载自：http://www.cnblogs.com/aisiteru/articles/1151874.html 第一次看这篇文章时，是在CSDN博客中看到的，作者写的初衷...

Java集合详解--什么是List

wz249863091 2016年10月19日 22:20 8006

简述上章简单介绍了什么是集合，集合有哪几种种类。 在这章中我们主要介绍Collection的其中一种实现方式，List。 什么是List 在上一章，我们已经了解了List主要分为3类，ArrayLi...

java中List的用法和实例详解

vaniice 2010年12月28日 10:05 175484

java中List的用法和实例详解List的用法List包括List接口以及List接口的所有实现类。因为List接口实现了Collection接口，所以List接口拥有Collection接口提供的...

java List(ArrayList)的5种遍历方式解析

xyc_csdn 2017年04月08日 17:47 3100

前言 对于ArrayList来说，从名字”数组列表”就知道它的底层其实是由数组实现的，同时ArrayList实现了RandomAccess接口，我们可以并且最好通过索引来访问ArrayList中的...

Java遍历List四种方法的效率对比

dengnanhua 2017年03月22日 00:53 10876

Java遍历List四种方法的效率对比 遍历方法简介 Java遍历List的方法主要有：（1）for each for(bject o :list) { ...

java.util.List学习

anyoneking 2007年04月17日 15:33 12228

接口List其上级接口 Collection已知实现类：ArrayList,LinkedList,Stack,Vector,AbstractList,AbstractSequentialList, ...

Java（十二）--List的添加，修改，删除

BenjaminYoung29 2015年08月23日 23:09 3481

首先我们先创建课程类/* * 课程类 */ public class Course { public String id; public String name; Cours...

java list三种遍历方法性能比较

jkh753 2013年09月13日 17:08 213234

从c/c++语言转向java开发，学习java语言list遍历的三种方法，顺便测试各种遍历方法的性能，测试方法为在ArrayList中插入1千万条记录，然后遍历ArrayList，发现了一个奇怪的现象...

劲爆！加拿大移民新政重开！

2018年加拿大移民门槛再创新低！

java List复制：浅拷贝与深拷贝

DeMonliuhui 2017年01

List浅拷贝众所周知，list本质上是数组，而数组的是以地址的形式进行存储。 如上图将list A浅拷贝给 贝，所以直接将A的内容复制给了B，java中相同内容的数组指向...

欢迎微信关注安卓进化论

联系我们

请扫描二维码联系客服

webmaster@csdn.net

400-660-0108

QQ客服 客服论坛

关于 招聘 广告服务 百度

©1999-2018 CSDN版权所有

京ICP证09002463号

经营性网站备案信息

网络110报警服务

中国互联网举报中心

北京互联网违法和不良信息举报中心

Java反转一个List或ArrayList

zhangphil 2017年01月13日 11:12 5307

package demo; import java.util.ArrayList; import java.util.Collections; /** * * 反转一个List，关键是使用C...

欢迎微信关注安卓进化论