

# Java语言基础

【Day13】

# List集合

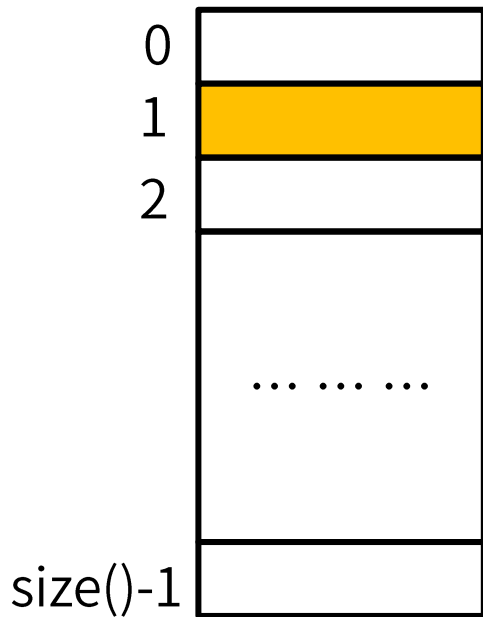
# List集合的概述

- List接口是Collection的子接口，用于定义线性表数据结构；可以将List理解为存放对象的数组，只不过其元素个数可以动态的增加或减少。
- List接口的两个常见实现类为ArrayList和LinkedList，分别用动态数组和链表的方式实现了List接口。
- 可以认为ArrayList和LinkedList的方法在逻辑上完全一样，只是在性能上有一定的差别，ArrayList更适合于随机访问而LinkedList更适合于插入和删除；在性能要求不是特别苛刻的情形下可以忽略这个差别。

# 实现类的底层原理

ArrayList

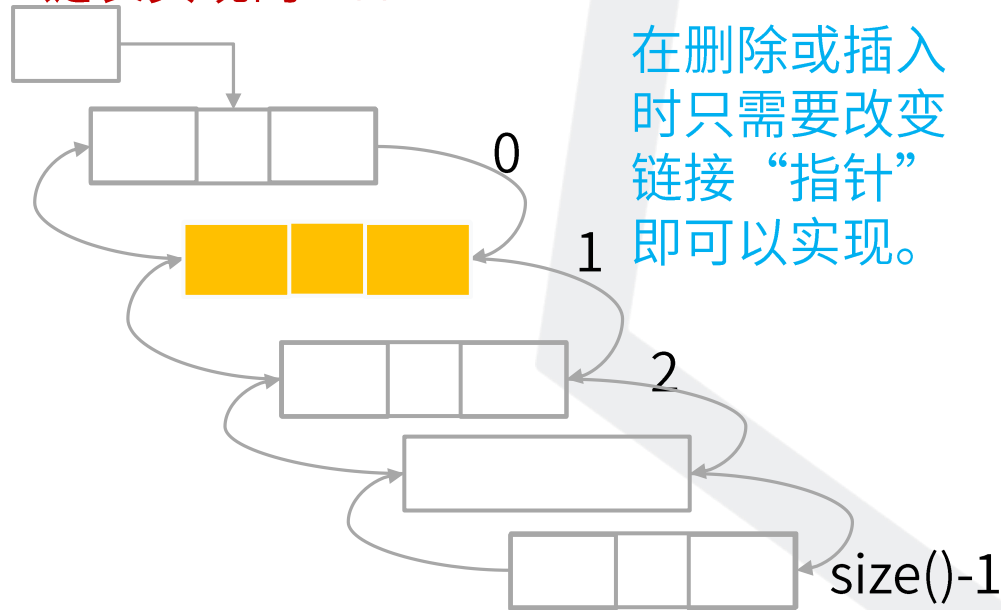
——动态数组实现的List



可以通过下标迅速的索引到对应的元素，但在删除和插入时移动较多元素。

LinkedList

——链表实现的List



在删除或插入时只需要改变链接“指针”即可以实现。

# List集合的常用方法

- List除了继承Collection定义的方法外，还根据其线性表的数据结构定义了一系列方法，其中最常用的就是基于下标的get和set方法。

|   |              |
|---|--------------|
| <code>void add(int index, E element)</code>                             | 向集合中指定位置添加元素 |
| <code>boolean addAll(int index, Collection&lt;? extends E&gt; c)</code> | 向集合中添加所有元素   |
| <code>E get(int index)</code>   | 从集合中获取指定位置元素 |
| <code>E set(int index, E element)</code>                                | 修改指定位置的元素    |
| <code>E remove(int index)</code>  | 删除指定位置的元素    |

# List集合的常用方法

- List还提供有类似于String的indexOf和lastIndexOf方法，用于在集合中检索某个对象，其判断逻辑为：`(o==null ? get(i)==null : o.equals(get(i)))`。
- toArray方法是继承自Collection的方法，可以将集合中的对象序列以对象数组的形式返回。

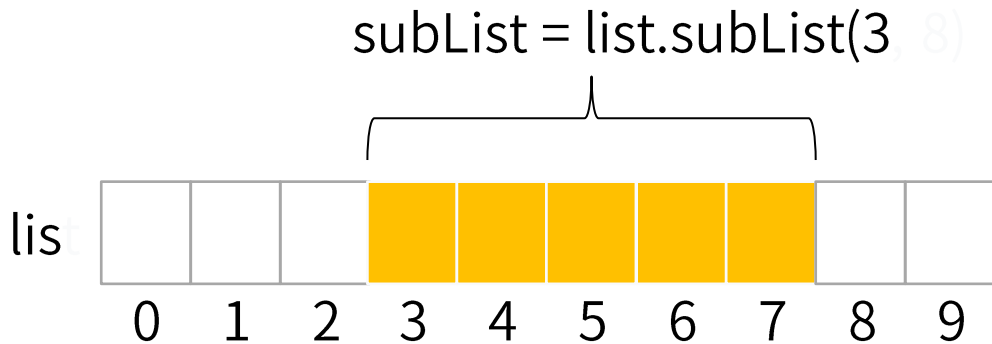
```
List list = new ArrayList();  
list.add("one");  
list.add("two");  
list.add("three");  
String[] array = (String[])list.toArray(new String[]{});  
System.out.println(Arrays.toString(array));
```

# List集合的常用方法

- List的subList方法用于获取子List。
- 需要注意的是，subList获取的List与原List占有相同的存储空间，对子List的操作会影响到原List。

List<E> subList(int fromIndex, int toIndex);

fromIndex和toIndex是截取子List的首尾下标（前包括，后不包括）



子list和原占用相同的存储空间；可以使用如下方式清除list中的某一段数据：  
list.subList(i, j).clear();

# Java泛型机制



# 泛型机制的概述

- 泛型是Java SE 1.5引入的特性，泛型的本质是参数化类型。在类、接口和方法的定义过程中，所操作的数据类型被传入的参数指定。

```
public class ArrayList<E> {  
    ... ..  
    public boolean add(E e) {···};  
    public E get(int index) {···};  
}
```

例如：ArrayList类<E>中的E为泛型参数，在创建对象时可以将类型作为参数传递，此时，类定义所有的E将被替换成传入的参数；

```
ArrayList<String> list = new ArrayList<String>();  
list.add("One");
```

`list.add(100);` Java 编译器类型检查错误，该方法应传入String类型

# 集合中的泛型机制

- Java泛型机制广泛的应用在集合框架中。所有的集合类型都带有泛型参数，这样在创建集合时可以指定放入集合中的对象类型。Java编译器可以据此进行类型检查，这样可以减少代码在运行时出现错误的可能性。

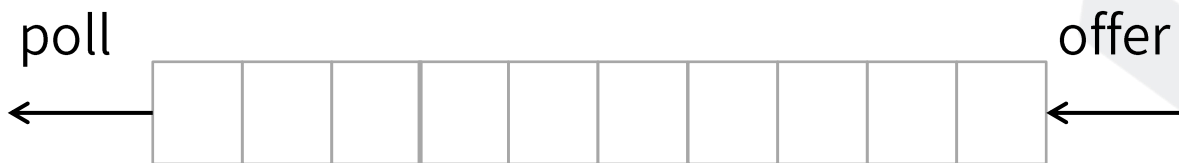
```
List<Point> pointList = new LinkedList<Point>();
```

通过泛型参数的指定可以创建“专门”存储Point的List集合

# Queue集合

# Queue集合的概述

- 队列 (Queue) 是常用的数据结构, 可以将队列看成特殊的线性表, 队列限制了对线性表的访问方式: 只能从线性表的一端添加元素, 从另一端取出元素
- 队列遵循先进先出 (FIFO **First Input First Output**) 的原则。
- Java中提供了Queue接口, 同时使得LinkedList实现了该接口 (选择LinkedList实现Queue的原因在于Queue经常要进行插入和删除的操作, 而LinkedList在这方面效率较高) 。



# Queue集合的方法

- Queue接口中主要方法如下：

|                                 |                         |
|---------------------------------|-------------------------|
| <code>boolean offer(E e)</code> | 将一个对象添加至队尾，若添加成功则返回true |
| <code>E poll()</code>           | 从队首删除并返回一个元素            |
| <code>E peek()</code>           | 返回队首的元素（但并不删除）          |

# Queue集合的使用

- Queue集合中的方法使用如下：

```
Queue<String> queue = new LinkedList<String>();  
queue.offer("A"); queue.offer("B"); queue.offer("C");  
System.out.println(queue); // [A, B, C]  
System.out.println(queue.peek()); // A  
String obj = null;  
while ((obj = queue.poll()) != null) {    队列空时，poll方法返回null。  
    System.out.print(obj+" "); // A B C  
}
```

# Set集合

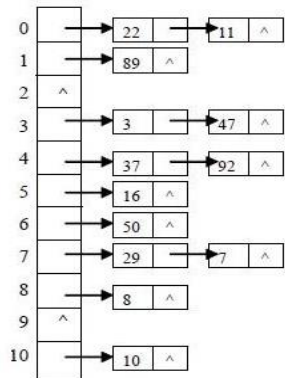
# Set集合的概述

- Set用于存储不重复的对象集合，任意两个对象 equals 比较为false。
- HashSet和TreeSet是Set集合两个常见的实现类，分别用哈希表和二叉树的方式实现了Set集合。



# HashSet集合放入元素的过程

- 将对象加入HashSet集合中时，需要获取对象的哈希码值通过哈希算法索引到对应的存储空间。



当生成的索引位置为2时：

由于该位置没有元素，则直接将新元素插入到索引2的位置。

当生成的索引位置为1时：

此时该位置有1个元素89，则使用新元素与89比较是否相等。

若相等，则放弃新元素的插入，保留旧元素；

若不相等，则将新元素插入到已有元素的后面；

... ..

# HashSet集合放入元素的过程

- 向HashSet中放元素的次序：

- 1 先调用元素的hashCode()方法得到哈希码，通过算法计算在哈希表中的位置。
- 2 如果该位置没有元素，直接放入即可。
- 3 如果该位置有元素，调用元素的equals()方法比较是不是相等。
- 4 如果相等，则保留旧元素丢弃新元素。
- 5 如果不相等，则放入该位置的链表中下一个元素。

# Set集合的遍历

- 所有Collection的实现类都实现了其iterator方法，该方法返回一个Iterator接口类型的对象，用于实现对集合元素的迭代遍历。
- Iterator接口的主要方法有：

|                   |                    |
|-------------------|--------------------|
| boolean hasNext() | 判断集合中是否有可以迭代/访问的元素 |
| E next()          | 用于取出一个元素并指向下一个元素   |
| void remove()     | 用于删除访问到的最后一个元素     |

# Set集合的遍历

- Java在5.0版本推出了增强型for循环语句，可以应用数组和集合的遍历。
- 是经典迭代的"简化版"

```
int[] arr = { 1, 2, 3, 4, 5, 6 };  
for (int i : arr) {  
    System.out.println(i);  
}
```

for(int i: arr)可以理解为每次循环从数组arr中取出一个元素赋值给循环变量i。

```
List<Point> pointList = new ArrayList<Point>();  
for (Point p : pointList) {  
    System.out.println(p);  
}
```

Java编译器在编译前会将其转换为迭代器的形式（因此不能对集合进行删除操作）

# 总结与答疑

技术资料





IT兄弟连  
ITXDL.CN

变态严管 让学习成为一种习惯