主要内容

# 缓冲区基础

概念上，**缓冲区**是包在一个对象内的基本数据元素**数组**。Buffer类相比一个简单数组的优点是：它将关于数据的数据内容和信息包含在一个单一的对象中。Buffer类以及它专有的子类定义了一个用于处理数据缓冲区的API。

所有的缓冲区都具有四个属性来提供关于其所包含的数据元素的信息。它们是：

1.容量（Capacity）2.上界（Limit）3.位置（Position）4.标记（Mark）

**容量（Capacity）**

缓冲区能够容纳的数据元素的最大数量。这一容量在缓冲区创建时被设定，并且永远不能被改变。

**上界（Limit）**

缓冲区的第一个不能被读或写的元素。或者说，缓冲区中现存元素的计数。

**位置（Position）**

下一个要被读或写的元素的索引。位置会自动由相应的get()和put()函数更新。

**标记（Mark）**

一个备忘位置。调用mark()来设定 mark = position。调用reset()设定position = mark。标记在设定前是未定义的（undefined）。

缓冲区的分类：

1.字节缓冲区ByteBuffer

2.字符缓冲区CharBuffer

3.双精浮点型（double）缓冲区DoubleBuffer

4.单精浮点型（float）缓冲区FloatBuffer

5.整型（int）缓冲区IntBuffer

6.长整型（long）缓冲区LongBuffer

7.短整型（short）缓冲区ShortBuffer

注：没有布尔类型的缓冲区

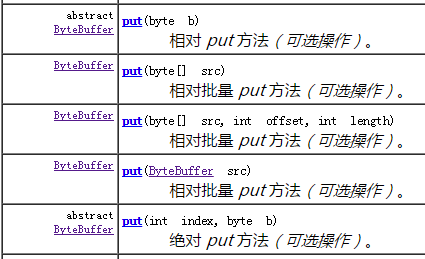
上述的各类型的缓冲区 都提供了读和写的方法 get和put方法，也提供了一些批量的put和get方法。而且缓冲区可以通过allocation创建，此方法通过wrapping将一个现有（数据类型）数组包装到缓冲区中来为缓冲区内容分配空间，或者通过创建现有字节缓冲区的视图来创建。

缓冲区的操作：

1.存：通过相关的Buffer类和put方法进行操作。

2.取：通过相关的Buffer类的get方法进行操作。

还有其他一些get/put方法的重载方式，参见API文档



# 创建缓冲区

Java.nio软件包中有七种主要的缓冲区类，每一种都具有一种Java语言中的非布尔类型的原始类型数据。这些类没有一种能够直接实例化。他们都是抽象类，但是都包含静态工厂方法用来创建相应类的新实例。

虽然有七种类型的缓冲区类，但是他们的创建方式基本都是类似的。

新的缓冲区是由分配或包装操作创建的。**分配操作**创建一个缓冲区对象并分配一个私有的空间来储存容量大小的数据元素。**包装操作**创建一个缓冲区对象但是不分配任何空间来储存数据元素。它使用您所提供的数组作为存储空间来储存缓冲区中的数据元素。

要分配一个容量为100个char变量的CharBuffer:

CharBuffer charBuffer = CharBuffer.allocate(100);

如果您想提供您自己的数组用作缓冲区的备份存储器，请调用wrap()函数：

char[] myArray = new char[100];

CharBuffer charbuffer = CharBuffer.wrap(myArray);

通过allocate()或者wrap()函数创建的缓冲区通常都是间接的。

间接的缓冲区使用备份数组，像我们之前讨论的，您可以通过上面流出的API函数获得对这些数组的存取权。Boolean型函数hasArray()告诉您这个缓冲区是否有一个可存取的备份数组。如果这个函数的返回true，array()函数会返回这个缓冲区对象所使用的数组存储空间的引用。

# 复制缓冲区

如我们刚刚所讨论的那样，可以创建描述从外部存储到数组中的数据元素的缓冲区对象。但是缓冲区不限于管理数组中的外部数据。它们也能管理其他缓冲区中的外部数据。当一个管理其他缓冲器所包含的数据元素的缓冲器被创建时，这个缓冲器被称为视图缓冲器。大多数的视图缓冲器都是ByteBuffer的视图。在继续前往字节缓冲器的细节之前，我们先将注意力放在所有缓冲器类型的共同视图上。

视图存储器总是通过调用已存在的存储器实例中的函数来创建。使用已存在的存储器实例中的工厂方法意味着视图对象为原始存储器的内部实现细节私有。数据元素可以直接存取，无论它们是存储在数组中还是以一些其他的方式，而不需经过原始缓冲区对象的get()/put() API。如果原始缓冲区是直接缓冲区，该缓冲区的视图会具有同样的效率优势。映像缓冲区也是如此。

Duplicate()函数创建了一个与原始缓冲区相似的新缓冲区。两个缓冲区共享数据元素，拥有同样的容量，但每个缓冲区拥有各自的位置，上界和标记属性。对一个缓冲区内的数据元素所做的改变会反应在另外一个缓冲区上。这一副本缓冲区具有与原始缓冲区同样的数据视图。如果原始的缓冲区为只读，或者直接缓冲区，新的缓冲区将继续这些属性。

注意：复制一个缓冲区会创建一个新的Buffer对象，但并不复制数据。原始缓冲区和副本都会操作同样的数据元素。