1.ByteBuffer类型化的put和get方法

```
public class NIOTest5 {
 public static void main(String[] args) {

 ByteBuffer buffer = ByteBuffer.allocate(64);

 buffer.putInt(15);
 buffer.putLong(500000000L);
 buffer.putDouble(14.123456);
 buffer.putChar('你');
  buffer.putShort((short)2);
  buffer.putChar('我');

  buffer.flip();

  System.out.println(buffer.getInt());
  System.out.println(buffer.getLong());
  System.out.println(buffer.getDouble());
  System.out.println(buffer.getChar());
  System.out.println(buffer.getShort());
  System.out.println(buffer.getChar());
  }
}
```

2.buffer.slice(类似分割)

调用slice方法返回一个新的Buffer，sliceBuffer和buffer的position和limit互不相关，相互独立，但是底层共享数据(数组)

```
public class NioTest6 {
 public static void main(String[] args) {
 ByteBuffer buffer = ByteBuffer.allocate(10);

 for (int i = 0; i < buffer.capacity(); ++i) {
 buffer.put((byte) i);
 }
 buffer.position(2);
 buffer.limit(6);
  ByteBuffer sliceBuffer = buffer.slice();
  for (int i = 0; i < sliceBuffer.capacity(); ++i) {
  byte b = sliceBuffer.get();
  b *= 2;
```

```
 sliceBuffer.put(i, b);
 }
 //sliceBuffer和buffer的position和limit互不相干,是独立的
 System.out.println("buffer的position: "+buffer.position());
 System.out.println("buffer的limit: "+buffer.limit());
 System.out.println("sliceBuffer的position: "+sliceBuffer.position());
 System.out.println("sliceBuffer的limit: "+sliceBuffer.limit());

 buffer.position(0);
 buffer.limit(buffer.capacity());
 //sliceBuffer和buffer底层共享数据(底层数组)
 while (buffer.remaining()>0){
 System.out.println(buffer.get());
 }
 }
}
```

3.buffer.asReadOnlyBuffer()

任何一个buffer可以调用asReadOnlyBuffer()可以返回一个只读buffer，反过来不可以

```
public class NioTest7 {
 public static void main(String[] args) {
 ByteBuffer buffer = ByteBuffer.allocate(10);
 System.out.println(buffer.getClass());
 for (int i = 0; i < buffer.capacity(); i++) {
 buffer.put((byte) i);
 }
 ByteBuffer readonlyBuffer = buffer.asReadOnlyBuffer();
 System.out.println(readonlyBuffer.getClass());
 readonlyBuffer.position(0);
 System.out.println(readonlyBuffer.put((byte)2));//会抛出异常
 }
}
```