CHAPTER 9

■ ■ ■

New Input/Output

QUESTIONS AND EXERCISES

1. What is new input/output?

Answer:

New Input/Output (NIO) is an Input/Output API to provide faster I/O compared to the standard I/O API. NIO uses buffers and channels for I/O operations.

2. What is a buffer? Name three classes that represent three different types of buffers.

Answer:

A buffer contains data to be written to a file or data that is read from a file. ByteBuffer, ShortBuffer and IntBuffer.

3. Define the capacity, position, and limit of a buffer. Write the invariant that must be true all the time for these three properties of a buffer.

Answer:

The capacity of a buffer is the maximum number of elements that it can hold. Capacity is fixed when buffer is created. A buffer's position is the index of the next element to be read or written. A buffer's limit is the index of the first element that should not be read or written. The following invariant must hold during the lifetime of a buffer:

```
0 <= mark <= position <= limit <= capacity
```

4. What is the difference between relative read and absolute read from a buffer?

Answer:

In an absolute read, an index is specified to read the data and the position is unchanged after the read. In a relative read, the number of data elements to be read is specified (current position determines which elements) and the position is incremented by one after reading each data element.

5. After you have written into a `Buffer`, what method do you need to call on the `Buffer` before you start reading the written data using a relative read?

Answer:

The `flip()` method

6. What is the difference between the `remaining()` and `hasRemaining()` methods of the `Buffer` class?

Answer:

The `hasRemaining()` method returns `true` if relative `get()` or `put()` can be called on the buffer to read/write at least one element. The `remaining()` method returns maximum number of elements that can be read/written using relative `get()` or `put()`.

7. What is the effect of calling the `clear()` method of a `Buffer`?

Answer:

The `clear()` method of a buffer sets the position to zero, limit to its capacity, and discards its mark. It does not change any data in the buffer.

8. What is the effect of calling the `reset()` method of a `Buffer`?

Answer:

The `reset()` method sets the position of a buffer equal to its mark. It does not affect the limit and data of the buffer.

9. What is the effect of calling the `rewind()` method of a `Buffer`?

Answer:

The `rewind()` method sets the position of the buffer to zero and discards its mark. It does not affect the limit and data of the buffer.

10. Write the output when the following code for a `TestReadOnlyBufferTest` class is run. This exercise is to test your knowledge about the properties of a read-only buffer.

```
// ReadOnlyBufferTest.java
package com.jdojo.nio;

import java.nio.IntBuffer;

public class ReadOnlyBufferTest {
    public static void main(String[] args) {
        // Create an IntBuffer of capacity 1
        IntBuffer data = IntBuffer.allocate(1);
        System.out.println(data.isReadOnly());

        // Get a read-only copy of the IntBuffer
        IntBuffer copy = data.asReadOnlyBuffer();
        System.out.println(copy.isReadOnly());

        // Print the contents of the read-only buffer
        System.out.println(copy.get());

        // Write into the original buffer
        data.put(64);

        // Print the contents of the read-only buffer again
        copy.rewind();
        System.out.println(copy.get());
    }
}
```

Answer:

```
false
true
0
```

64

11. Suppose you have an IntBuffer. What is the difference between creating two copies of the IntBuffer, one by using the asReadOnlyBuffer() method and another by using the duplicate() method?

Answer:

In both cases, the new buffers share the contents of the original buffer and they maintain position, mark, and limit independently. The asReadOnlyBuffer() method always returns a read-only view whereas the duplicate() method returns a copy, which is read-only when the original buffer is read-only.

12. What do instances of the following classes represent: Charset, CharsetEncoder, and CharsetDecoder?

Answer:

An instance of the java.nio.charset.Charset class represents a character set and a character-encoding scheme. The CharsetEncoder class performs the character encoding (converting a character into a sequence of bytes based on encoding scheme). The CharsetDecoder class performs the decoding (converting a sequence of bytes into a character based on an encoding scheme).

13. What is a channel? What is the fully qualified name of the interface that every implementation of a channel implements? If you have a reference to a channel, how would you tell if the channel is open?

Answer:

A channel is an open connection between a data source (or a data sink) and a Java program to perform some I/O operations. Every channel implements the java.nio.channels.Channel interface. The isOpen() method returns true if a channel is open.

14. When do you use instances of the GatheringByteChannel and ScatteringByteChannel?

Answer:

A GatheringByteChannel (which writes data from multiple byte buffers to a data sink) is used to write data in a format that is grouped in some fixed-length headers, followed by a variable length body.

A ScatteringByteChannel (which reads data from a data source into multiple byte buffers) is used to read data from a known file format or a similar data source, where data is supplied in some fixed-length headers followed by a variable length body.

15. Suppose you have a file named test.txt in your current directory. Write a snippet of code to get a FileChannel for this file in read-write mode.

Solution:

```
// Open file in a read-write mode
RandomAccessFile raf = new RandomAccessFile("test.txt", "rw");

// A read-write channel
FileChannel rafReadWrite = raf.getChannel();
```

16. What is memory-mapped file I/O? Name a class whose instances are used to work with memory-mapped file I/O.

Answer:

Memory-mapped file I/O is mapping a region of the file into physical memory and treating it as a memory array. This is the fastest way available to perform file I/O in Java. Special kind of byte buffer called MappedByteBuffer is used to perform memory-mapped file I/O.

17. What is the difference in using the lock() and tryLock() methods of the FileChannel class while locking a region of a file?

Answer:

The lock() method blocks if the lock on the requested region of the file is not available. The tryLock() method does not block; it returns immediately. It returns an object of the FileLock class if the lock was acquired; otherwise, it returns null.

18. Write a snippet of code that prints the byte order (little-endian or big-endian) of the current machine.

Solution:

```
ByteOrder b = ByteOrder.nativeOrder();
if (b.equals(ByteOrder.BIG_ENDIAN)) {
    System.out.println("Big endian");
} else {
   System.out.println("Little endian");
}
```