استخدام □□□ في جافا

أفهم كيف يمكن استخدام □□□ □□□□ □□□□□□□□□□□□□□ في □□□□ مع بعض الأمثلة. العملية □□□□ □□□ هو □□□ □□□ غير متزامن ويوفر طريقة أكثر كفاءة للتعامل مع عمليات □□□ مقارنة بالباقة التقليدية □□□□.□□.

بسيط: ومثال الرئيسية المكونات تحليل هو انه

المكونات الرئيسية في □□□

1. □□□□□□□□ال: أوعية للبيانات مثل□ ByteBuffer□.
2. □□□□□□□□□: قنوات الكتابة□للقراءة للبيانات مثل□ FileChannel, SocketChannel□.
3. □□□□□□□□□□: لتعدد القنوات في الوضع غير المتزامن.

مثال: قراءة أساسي ملف باستخدام □□□

هكذا يمكنك قراءة ملف باستخدام FileChannel و ByteBuffer:

```java
import java.nio.ByteBuffer;
import java.nio.channels.FileChannel;
import java.nio.file.Paths;
import java.nio.file.StandardOpenOption;

public class NIOFileReader {
    public static void main(String[] args) {
        try {
            //
            FileChannel fileChannel = FileChannel.open(
                Paths.get("example.txt"),
                StandardOpenOption.READ
            );

            //
            ByteBuffer buffer = ByteBuffer.allocate(1024);

            //
            while (fileChannel.read(buffer) > 0) {
                //
                buffer.flip();

                //
                while (buffer.hasRemaining()) {
```

```java
                System.out.print((char) buffer.get());
            }

            //
            buffer.clear();
        }

        //
        fileChannel.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
  }
}
```

فيك لمعي

1. □□□□□□□□□□□□□: حتفي الًاصتا .فلملاب
2. □□□□□□□□□□□□: لمعي نزخك تقؤم .تانايبلل كنكمي صيصخت همجح لثم□ 1024 .□تياب
3. تايلمع ةظفاحلا:
   - □ `flip()`: زهجي ةظفاحلا ةءارقلل دعب ةباتك تانايبلا .اهيف
   - □ `clear()`: ديعي نييعت ةظفاحلا ةعطقلل ةيلاتلا نم .تانايبلا
4. ةءارقلا: ةقيرط `read()` ءيلمت ،ةظفاحلا جرختستو تانايبلا اثًياب .اثًياب

ةباتك يف فلم مادختساب □□□

انه لاثم ةباتكلل يف :فلم

```java
import java.nio.ByteBuffer;
import java.nio.channels.FileChannel;
import java.nio.file.Paths;
import java.nio.file.StandardOpenOption;

public class NIOFileWriter {
    public static void main(String[] args) {
        try {
            //
            FileChannel fileChannel = FileChannel.open(
```

```java
                Paths.get("output.txt"),
                StandardOpenOption.CREATE,
                StandardOpenOption.WRITE
            );

            //
            String data =     " NIO!";
            ByteBuffer buffer = ByteBuffer.allocate(1024);

            //
            buffer.put(data.getBytes());

            //
            buffer.flip();

            //
            fileChannel.write(buffer);

            //
            fileChannel.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

مثال غير متزامن باستخدام □□□□□□□□

للبرمجة الشبكية مثال□ ،□الخادم يمكنك استخدام Selector مع SocketChannel:

```java
import java.nio.ByteBuffer;
import java.nio.channels.SelectionKey;
import java.nio.channels.Selector;
import java.nio.channels.ServerSocketChannel;
import java.nio.channels.SocketChannel;
import java.net.InetSocketAddress;
import java.util.Iterator;

public class NIOServer {
```

```java
public static void main(String[] args) throws Exception {
    //    Selector
    Selector selector = Selector.open();

    //          Socket
    ServerSocketChannel serverChannel = ServerSocketChannel.open();
    serverChannel.bind(new InetSocketAddress(8080));
    serverChannel.configureBlocking(false);

    //           Selector
    serverChannel.register(selector, SelectionKey.OP_ACCEPT);

    System.out.println( "             "8080);

    while (true) {
        selector.select(); //
        Iterator<SelectionKey> keys = selector.selectedKeys().iterator();

        while (keys.hasNext()) {
            SelectionKey key = keys.next();
            keys.remove();

            if (key.isAcceptable()) {
                //
                SocketChannel client = serverChannel.accept();
                client.configureBlocking(false);
                client.register(selector, SelectionKey.OP_READ);
                System.out.println( "      "    ;(
            } else if (key.isReadable()) {
                //
                SocketChannel client = (SocketChannel) key.channel();
                ByteBuffer buffer = ByteBuffer.allocate(1024);
                int bytesRead = client.read(buffer);
                if (bytesRead == -1) {
                    client.close();
                } else {
                    buffer.flip();
                    System.out.println( " :      " + new String(buffer.array(), 0, bytesRead));
                    buffer.clear();
                }
```

```
                    }
                }
            }
        }
}
```

حئاصن مادختسال

- **الحافظة إدارة**: دائمًا `flip()` بعد كتابة في الحافظة وقبل القراءة، و `clear()` عند إعادة استخدامها.
- **متزامن غير**: استخدم `configureBlocking(false)` و `Selector` للبرامج الشبكية قابلة للتوسيع.
- **الأداء**: عملي في السيناريوهات التي تحتوي على العديد من الإتصالات المتزامنة أو نقل البيانات الكبيرة.

أعلمني إذا كنت ترغب في غوص أعمق في أي جزء معين!