感兴趣，交由 Selector 进行监听和管理。之后就可以调用 Selector 的 select()方法阻塞等待事件就绪，当有事件就绪时，就可以获取所有就绪的 SelectionKey 集合。最后根据不同的就绪事件执行不同的事件处理逻辑。

客户端代码清单：

```
package samples.javanio;

import java.io.IOException;
import java.net.InetSocketAddress;
import java.nio.ByteBuffer;
import java.nio.channels.SelectionKey;
import java.nio.channels.Selector;
import java.nio.channels.SocketChannel;
import java.util.Iterator;
import java.util.Set;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

public class Client {

    private final String serverHost;
    private final int serverPort;
    private Selector selector;
    private SelectionKey selectionKey;
    private ExecutorService executorService = Executors.newSingleThreadExecutor();

    private SocketChannel client;

    public Client(String serverHost, int serverPort) {
        this.serverHost = serverHost;
        this.serverPort = serverPort;
    }

    public void connect() throws IOException {
        // 获取一个客户端 Socket 通道
        SocketChannel socketChannel = SocketChannel.open();
        // 设置 Socket 为非阻塞方式
```