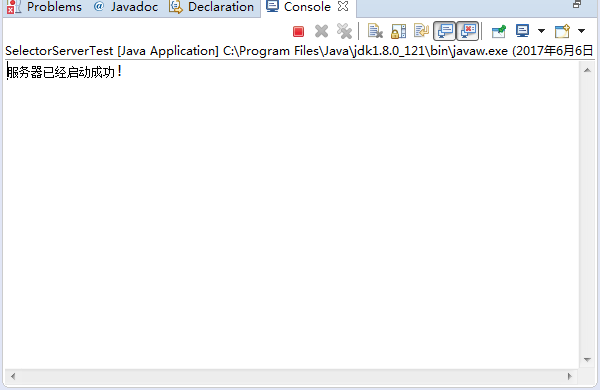
打开Eclipse，在JavaNIO项目创建selector包，并创建类SelectorServerTest，代码如下：

|  |
| --- |
| **package** selector;  **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.ServerSocketChannel;  **import** java.nio.channels.SocketChannel;  **import** java.util.Iterator;  **public** **class** SelectorServerTest {  **private** Selector selector;  **private** ServerSocketChannel serverChannel = **null**;  **private** **int** keys = 0;  /\*\*  \* 初始化服务端的连接通道和管理器，已经注册事件  \* **@throws** IOException  \*/  **public** **void** initServer() **throws** IOException {  **this**.selector = Selector.*open*();// SelectorProvider.provider().open();  serverChannel = ServerSocketChannel.*open*();  serverChannel.socket().bind(**new** InetSocketAddress("127.0.0.1", 8888));  serverChannel.configureBlocking(**false**);  // 把serverChannel这个通道注册到通道管理器对象acceptSelector中去，当有客户端连接时触发  SelectionKey key = serverChannel.register(**this**.selector, SelectionKey.***OP\_ACCEPT***);  }  /\*\*  \* 对客户端的请求(通道上面感兴趣的事件)进行监听  \* **@throws** IOException  \*/  **public** **void** listen() **throws** IOException {  System.***out***.println("服务器已经启动成功!");  **while** (**true**) {  // 让通道管理器至少选择一个通道  keys = **this**.selector.select();  Iterator it = **this**.selector.selectedKeys().iterator();  **if** (keys > 0) {  // 进行轮询  **while** (it.hasNext()) {  SelectionKey key = (SelectionKey) it.next();  it.remove();  // 客户端请求连接事件  **if** (key.isAcceptable()) {  serverChannel = (ServerSocketChannel) key.channel();  // 获得和客户端连接的通道  SocketChannel channel = serverChannel.accept();  channel.configureBlocking(**false**);// 设置为非阻塞模式    // 给客户端发送消息  channel.write(ByteBuffer.*wrap*(**new** String("hello client.").getBytes()));  // 还需要读取客户端过来的数据，所以注册一个去读取数据的事件  channel.register(**this**.selector, SelectionKey.***OP\_READ***);  }  **else** **if** (key.isReadable()) {  read(key);  }  }  }  **else** {  System.***out***.println("Select finished without any keys.");  }  }  }  /\*\*  \* 根据SelectionKey对象来读取客户端发送的到通道里面的数据  \* **@param** key  \* **@throws** IOException  \*/  **public** **void** read(SelectionKey key) **throws** IOException {  SocketChannel channel = (SocketChannel) key.channel();  // 缓冲区  ByteBuffer buff = ByteBuffer.*allocate*(1024);  **int** len = channel.read(buff);  String msg = "服务器收到的消息为：" + **new** String(buff.array(), 0, len);  System.***out***.println(msg);  }  /\*\*  \* 启动服务  \*/  **public** **void** start() {  **try** {  SelectorServerTest ns = **new** SelectorServerTest();  ns.initServer();  ns.listen();  } **catch** (IOException e) {  e.printStackTrace();  }  }  **public** **static** **void** main(String[] args) {  **new** SelectorServerTest().start();  }  } |

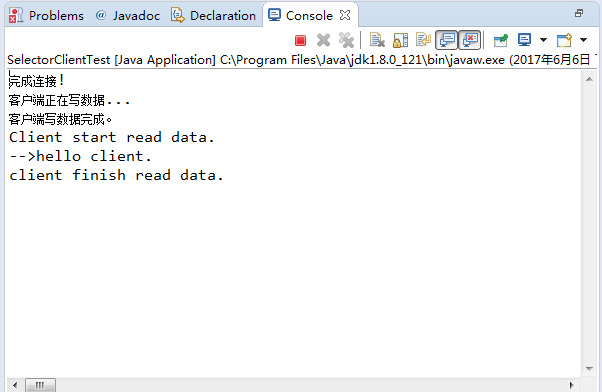
创建类SelectorClientTest，代码如下：

|  |
| --- |
| **package** selector;  **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;  /\*\*  \* NIO选择器的实例——客户端  \* **@author** lenovo  \*  \*/  **public** **class** SelectorClientTest {  **private** Selector selector;  **private** ByteBuffer outBuff = ByteBuffer.*allocate*(1024);  **private** ByteBuffer inBuff = ByteBuffer.*allocate*(1024);  **private** **int** keys = 0;  **private** SocketChannel channel = **null**;  **public** **void** initClient() **throws** IOException {  // 获得一个socket通道，并没有进行连接  channel = SocketChannel.*open*();  // 获得一个通道管理器  selector = Selector.*open*();  // 设置为非阻塞  channel.configureBlocking(**false**);  // 连接服务器  channel.connect(**new** InetSocketAddress("127.0.0.1", 8888));  // 注册客户端连接服务器的事件  channel.register(**this**.selector, SelectionKey.***OP\_CONNECT***);  }  /\*\*  \* 监听在通道上面进行注册的事件  \* **@throws** IOException  \*/  **public** **void** listen() **throws** IOException {  // 设置轮询  **while** (**true**) {  keys = **this**.selector.select();  **if** (keys > 0) {  // 获得通道管理器事件注册的集合  Iterator<SelectionKey> it = **this**.selector.selectedKeys().iterator();  **while** (it.hasNext()) {  SelectionKey key = it.next();  // 测试此通道是否完成套接字的连接  **if** (key.isConnectable()) {  // 获得与服务器相连的通道  SocketChannel channel = (SocketChannel) key.channel();  // 如果正在连接 就连接完成  **if** (channel.isConnectionPending()) {  channel.finishConnect();  System.***out***.println("完成连接!");  }  channel.register(**this**.selector, SelectionKey.***OP\_WRITE***);  }  // 在通道上面进行写操作  **else** **if** (key.isWritable()) {  SocketChannel channel = (SocketChannel) key.channel();  outBuff.clear();  // outBuff.flip();  System.***out***.println("客户端正在写数据...");  channel.write(outBuff.*wrap*("我是clientA".getBytes()));  channel.register(**this**.selector, SelectionKey.***OP\_READ***);  System.***out***.println("客户端写数据完成。");  }  // 在通道上面进行读取  **else** **if** (key.isReadable()) {  SocketChannel channel = (SocketChannel) key.channel();  inBuff.clear();  System.***out***.println("Client start read data.");  channel.read(inBuff);  System.***out***.println("-->" + **new** String(inBuff.array()));  System.***out***.println("client finish read data.");  }  }  } **else** {  // System.out.println("没有找到感兴趣的事件。");  }  }  }  /\*\*  \* 启动  \*/  **public** **void** start() {  **try** {  initClient();  listen();  } **catch** (IOException e) {  e.printStackTrace();  }  }    **public** **static** **void** main(String[] args) {  **new** SelectorClientTest().start();  }  } |

运行服务器端程序



运行客户端程序：



此时服务端控制台打印：

