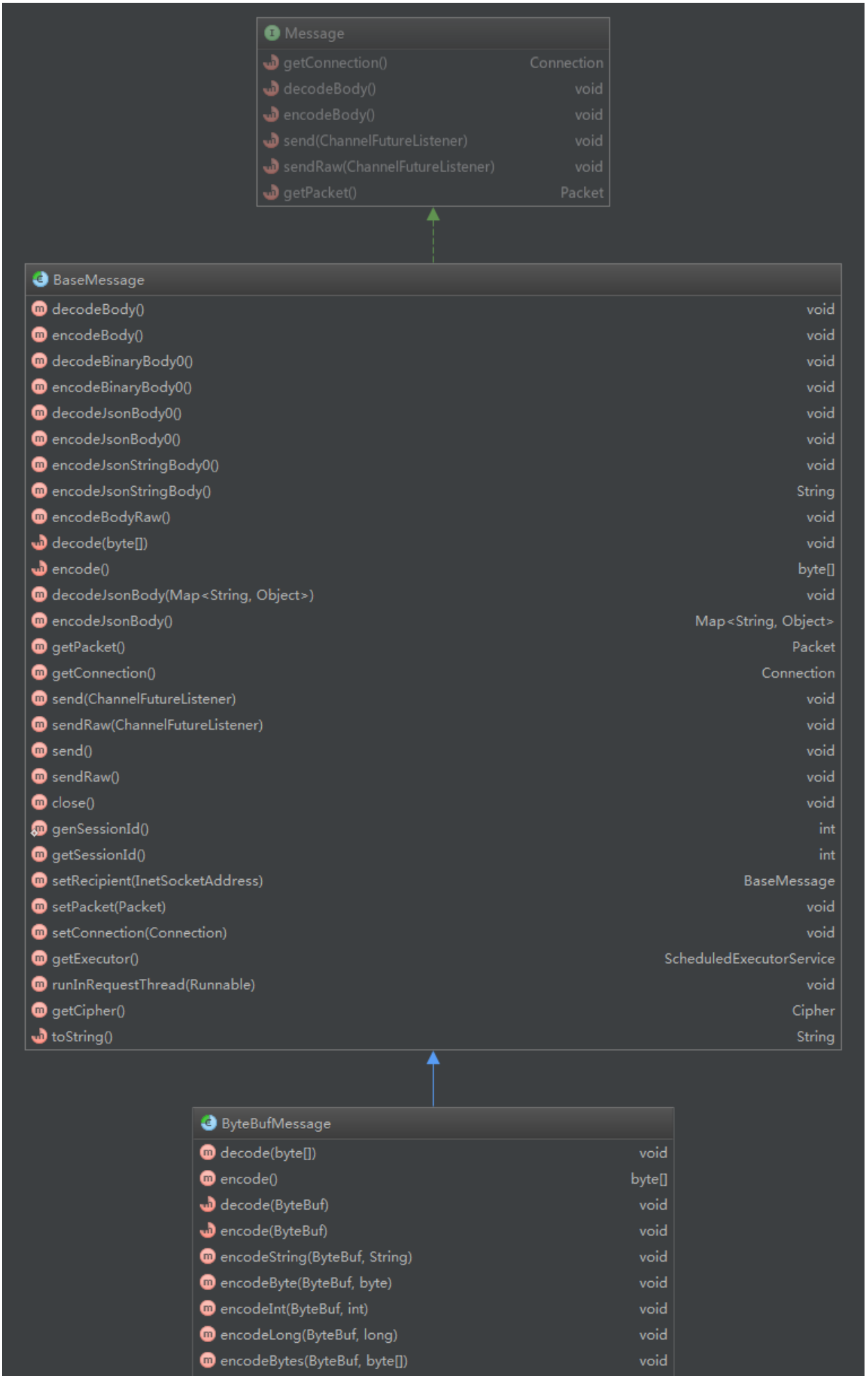
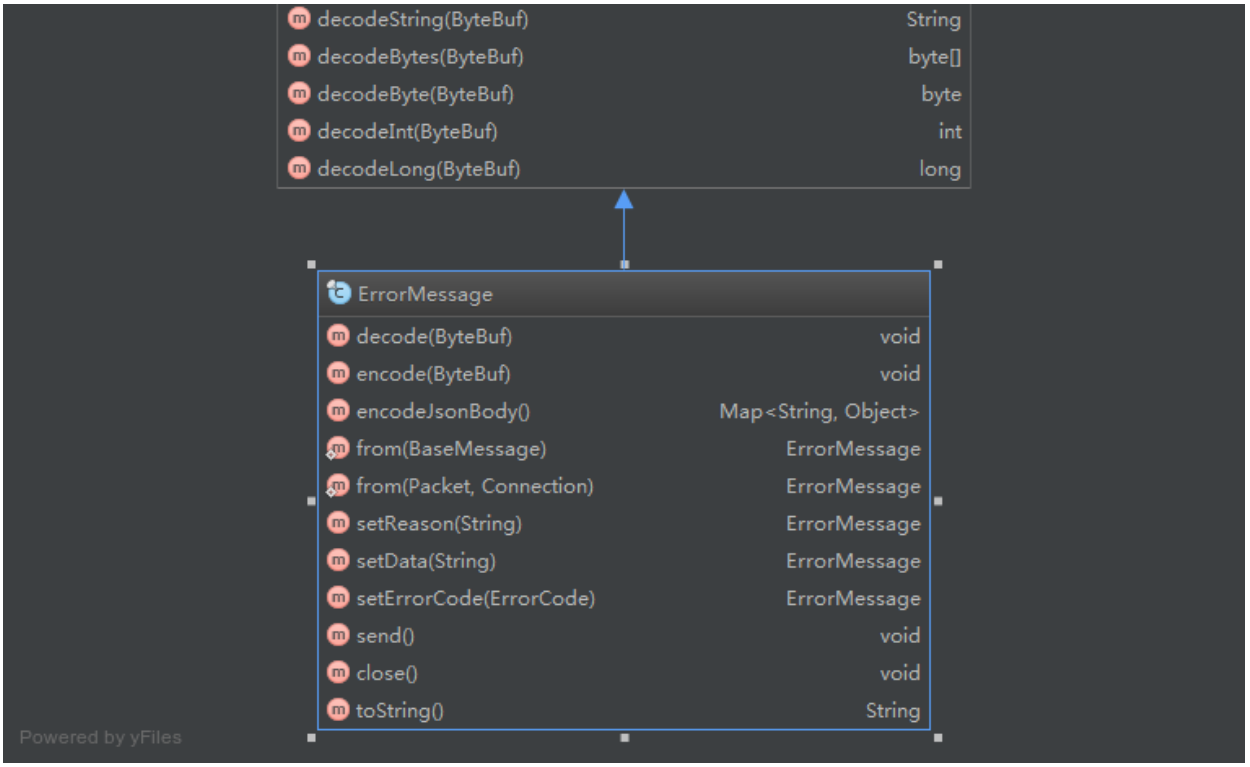
1、接收错误消息

```
//ConnClientChannelHandler.java

@Override
public void channelRead(ChannelHandlerContext ctx, Object msg) throws Exc
eption {
 connection.updateLastReadTime();
 if (msg instanceof Packet) {
 Packet packet = (Packet) msg;
 Command command = Command.toCMD(packet.cmd);
 if (command == Command.HANDSHAKE) {
  ...
  } else if (command == Command.FAST_CONNECT) {
  ...
  } else if (command == Command.KICK) {
  ...
  } else if (command == Command.ERROR) {
  ErrorMessage message = new ErrorMessage(packet, connection);
  message.decodeBody();
  //TODO 根据消息类型，做业务处理
  //if (message.cmd == Command.BIND.cmd) {
  // //TODO 根据错误码code，做业务处理
  // LOGGER.info(">>> bind user failure.");
  //}
  LOGGER.error("receive an error packet=" + message);
  } else if (command == Command.PUSH) {
  ...
  } else if (command == Command.HEARTBEAT) {
  LOGGER.info("receive heartbeat pong...");
  } else if (command == Command.OK) {
  ...
  } else if (command == Command.HTTP_PROXY) {
  ...
  }
  }
  LOGGER.debug("receive package={}, chanel={}", msg, ctx.channel());
}


```


Message
getConnection() Connection
decodeBody() void
encodeBody() void
send(ChannelFutureListener) void
sendRaw(ChannelFutureListener) void
getPacket() Packet
BaseMessage
decodeBody() void
encodeBody() void
decodeBinaryBody0() void
encodeBinaryBody0() void
decodeJsonBody0() void
encodeJsonBody0() void
encodeJsonStringBody0() void
encodeJsonStringBody() String
encodeBodyRaw() void
decode(byte[]) void
encode() byte[]
decodeJsonBody(Map<String, Object>) void
encodeJsonBody() Map<String, Object>
getPacket() Packet
getConnection() Connection
send(ChannelFutureListener) void
sendRaw(ChannelFutureListener) void
send() void
sendRaw() void
close() void
genSessionId() int
getSessionId() int
setRecipient(InetSocketAddress) BaseMessage
setPacket(Packet) void
setConnection(Connection) void
getExecutor() ScheduledExecutorService
runInRequestThread(Runnable) void
getCipher() Cipher
toString() String
ByteBufMessage
decode(byte[]) void
encode() byte[]
decode(ByteBuf) void
encode(ByteBuf) void
encodeString(ByteBuf, String) void
encodeByte(ByteBuf, byte) void
encodeInt(ByteBuf, int) void
encodeLong(ByteBuf, long) void
encodeBytes(ByteBuf, byte[]) void




```java
public final class ErrorMessage extends ByteBufMessage {
 public byte cmd;
 public byte code;
 public String reason;
 public String data;

 public ErrorMessage(byte cmd, Packet message, Connection connection) {
 super(message, connection);
 this.cmd = cmd;
 }

 public ErrorMessage(Packet message, Connection connection) {
 super(message, connection);
 }

 @Override
 public void decode(ByteBuf body) {
 cmd = decodeByte(body);
 code = decodeByte(body);
 reason = decodeString(body);
 data = decodeString(body);
 }

 @Override
```

```
25  public void encode(ByteBuf body) {
26  encodeByte(body, cmd);
27  encodeByte(body, code);
28  encodeString(body, reason);
29  encodeString(body, data);
30  }
31
32  @Override
33  protected Map<String, Object> encodeJsonBody() {
34  Map<String, Object> body = new HashMap<>(4);
35  if (cmd > 0) body.put("cmd", cmd);
36  if (code > 0) body.put("code", code);
37  if (reason != null) body.put("reason", reason);
38  if (data != null) body.put("data", data);
39  return body;
40  }
41
42  public static ErrorMessage from(BaseMessage src) {
43  return new ErrorMessage(src.packet.cmd, src.packet.response(ERROR),
src.connection);
44  }
45
46  public static ErrorMessage from(Packet src, Connection connection) {
47  return new ErrorMessage(src.cmd, src.response(ERROR), connection);
48  }
49
50
51  public ErrorMessage setReason(String reason) {
52  this.reason = reason;
53  return this;
54  }
55
56  public ErrorMessage setData(String data) {
57  this.data = data;
58  return this;
59  }
60
61  public ErrorMessage setErrorCode(ErrorCode code) {
62  this.code = code.errorCode;
63  this.reason = code.errorMsg;
64  return this;
```

```
65  }
66
67  @Override
68  public void send() {
69  super.sendRaw();
70  }
71
72  @Override
73  public void close() {
74  sendRaw(ChannelFutureListener.CLOSE);
75  }
76
77  @Override
78  public String toString() {
79  return "ErrorMessage{" +
80  "reason='" + reason + '\'' +
81  ", code=" + code +
82  ", data=" + data +
83  ", packet=" + packet +
84  '}';
85  }
86 }
```