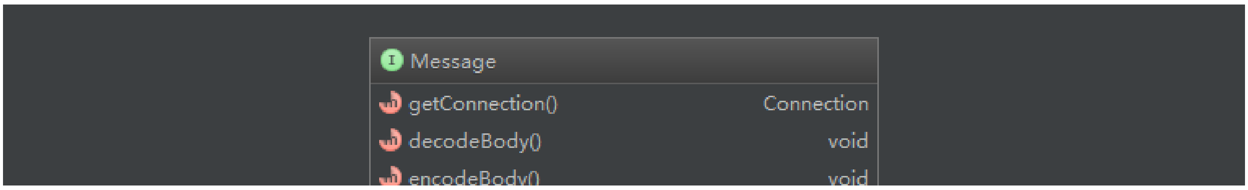
1、接收踢人消息

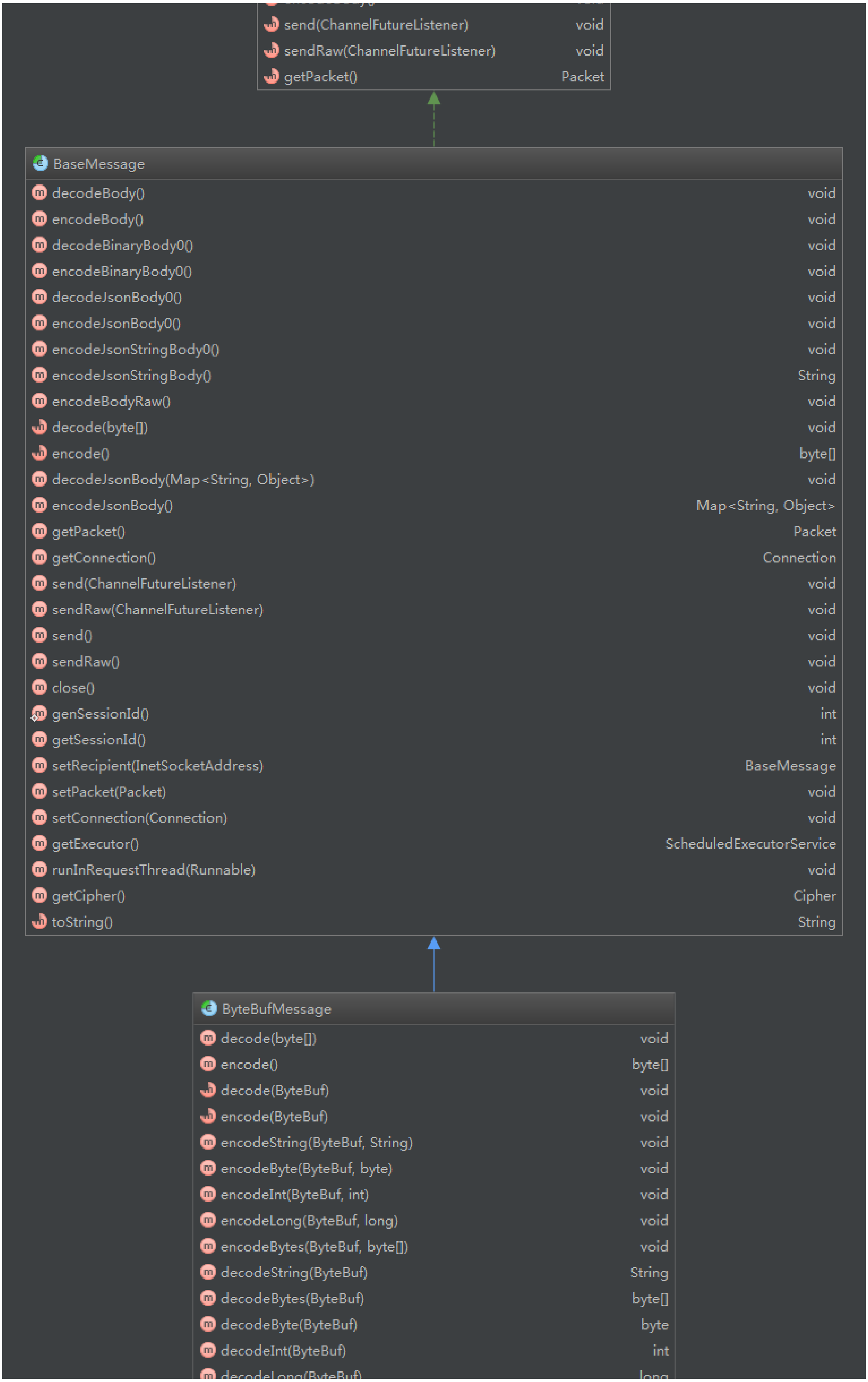
```
//ConnClientChannelHandler.java

@Override
public void channelRead(ChannelHandlerContext ctx, Object msg) throws Exception {
 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) {
  KickUserMessage message = new KickUserMessage(packet, connection);
  //TODO 这里可以做些业务处理(类似QQ/微信多端登录互踢), 如: 用户同意下线(发送unbind消息), 不同意下线则忽略不管它;
  LOGGER.error("receive kick user msg userId={}, deviceId={}, message={},", clientConfig.getUserId(), clientConfig.getDeviceId(), message);
  ctx.close();
  } else if (command == Command.ERROR) {
  ...
  } 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());
 }

```



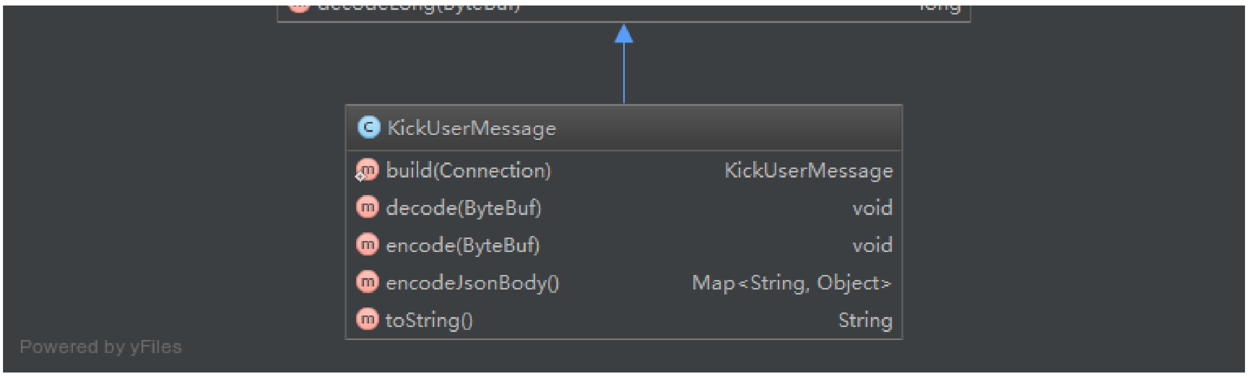
```
Message
getConnection()     Connection
decodeBody()        void
encodeBody()        void
```



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
decodeString(ByteBuf)	String
decodeBytes(ByteBuf)	byte[]
decodeByte(ByteBuf)	byte
decodeInt(ByteBuf)	int
decodeLong(ByteBuf)	long



KickUserMessage.java

```java
public class KickUserMessage extends ByteBufMessage {
 public String deviceId;
 public String userId;
 public KickUserMessage(Packet message, Connection connection) {
 super(message, connection);
 }
 public static KickUserMessage build(Connection connection) {
 if (connection.getSessionContext().isSecurity()) {
 return new KickUserMessage(new Packet(KICK), connection);
 } else {
 return new KickUserMessage(new JsonPacket(KICK), connection);
 }
 }
 @Override
 public void decode(ByteBuf body) {
 deviceId = decodeString(body);
 userId = decodeString(body);
 }
 @Override
 public void encode(ByteBuf body) {
 encodeString(body, deviceId);
 encodeString(body, userId);
 }
 @Override
 protected Map<String, Object> encodeJsonBody() {
 Map<String, Object> body = new HashMap<>(2);
 body.put("deviceId", deviceId);
 body.put("userId", userId);
 return body;
 }
 @Override
 public String toString() {
 return "KickUserMessage{" +
 "deviceId='" + deviceId + '\'' +
 ", userId='" + userId + '\'' +
 '}';
 }
}
```