

1. 为网络版坦克做好准备，学习真正生产环境中的TCPServer的写法

a. 学习使用JDK Loggin API

i.

<https://www.cnblogs.com/liaojie970/p/5582147.html>

b. 复习BIO-NIO-AIO-Netty的编程模型

i. 网络程序的烦点 异常处理 正确关闭 ->线程的正常结束 ->线程池的正常结束

c. 写一个NettyClient

i. EventLoopGroup 网络IO事件处理线程组

ii. Netty中的任何方法都是异步模型

iii. 首先使用ChannelInitializer.initChannel() 添加channel的handler

iv. 在channelHandler中处理业务逻辑，学习使用ChannelInboundHandlerAdapter

d. 写一个NettyServer

i. nettyclient传递数据给nettyserver

ii. ByteBuf的使用

iii. 保存多个客户端

iv. 接收到一个客户端的数据后传递给多个客户端

e. 写一个图形化的Client，顺带完成聊天

f. 写一个图形化的NettyServer

g. 将NettyServer能够优雅的关闭

h. 将项目转化为Maven项目

i. 项目右击 - Configure - Convert to Maven Project

ii. 在Maven中添加依赖: io.netty:netty-all:4.1.9.Final

iii. 如果出现java.lang.Object无法解析问题, 重置buildpath

2. check out master

3. 去掉NPC Non-Player Character

a. Main.java 敌人不再初始化

4. 为了在网络端区分坦克, 在Tank中加入UUID属性

5. 创建net package

6. 创建net.Server类

a. 为了调试程序, 将Server做成窗口类

```
public class Server extends Frame {
```

```
    Button btnStart = new Button("start");
```

```
    TextArea taLeft = new TextArea();
```

```
    TextArea taRight = new TextArea();
```

```
    public Server() {
```

```
        this.setSize(1600, 600);
```

```
        this.setLocation(300, 30);
```

```
        this.add(btnStart, BorderLayout.NORTH);
```

```
        Panel p = new Panel(new GridLayout(1, 2));
```

```
        p.add(taLeft);
```

```
        p.add(taRight);
```

```
        this.add(p);
```

```
        this.addWindowListener(new WindowAdapter() {
```

```
            @Override
```

```
            public void windowClosing(WindowEvent e) {
```

```
                System.exit(0);
```

```
            }
```

```
        });
```

```
        this.setVisible(true);
```

```
    }
```

```
    public static void main(String[] args) {
```

```
        new Server();
```

```
    }
```

```
}
```

7. 添加辅助方法: addLeft addRight

```

public void addLeft(String txt) {
    taLeft.setText(taLeft.getText() + txt +
        System.getProperty("line.separator"));
}

```

8. 加入对于btnStart的Action处理

```

btnStart.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        button.setEnabled(false);
        button.setLabel("started");
        try {
            Server.this.start();
        } catch (Exception e1) {
            e1.printStackTrace();
        }
    }
});

```

9. 完成start方法

- a. 创建bossGroup和workerGroup
- b. 创建ServerBootstrap
- c. 加入ChannelInitializer的处理
- d. 每一个Channel addLast(new ServerHandler)

10. 完成ServerHandler, 重写方法

- a. channelRead()
- b. exceptionCaught()

11. Client端编程

- a. 连接并发送一个字符串
- b. 服务器端接收并返回
- c. 客户端接收

12. 正确的结束程序

13. 自定义消息 使用Netty Codec API