# 分布式计算第二次作业

**姓名：张泽群 学号：19049100002 课程号：CS205105**

西安电子科技大学 XIDIAN UNIVERSITY 计算机科学与技术学院 School of Computer Science and Technology

**第二次作业！**

- 准备：

安装jdk，安装maven，为maven设置阿里或腾讯镜像仓库，用maven编译Helloworld版的java程序

- 题目：

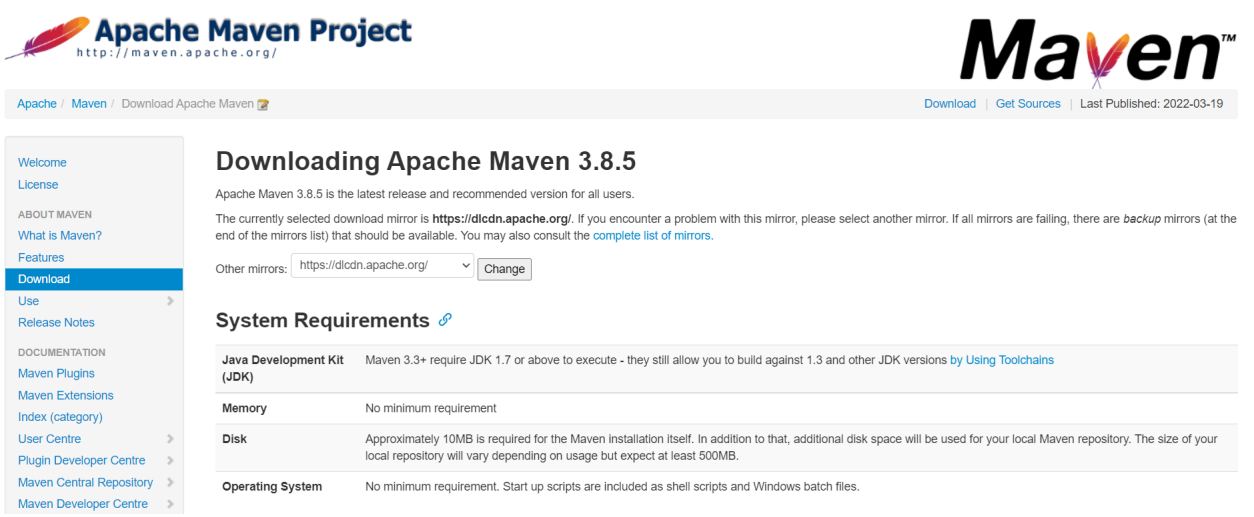
将基于TCP协议的Client-Server通信程序示例的服务器端程序改造成线程池版。

- 提交要求：

1. 3月28日前将源程序发送至：xddistcomcourse@163.com
2. 邮件标题风格：第2次作业+学号+姓名
3. 源程序打包文件命名方式：第2次作业+学号+姓名.zip

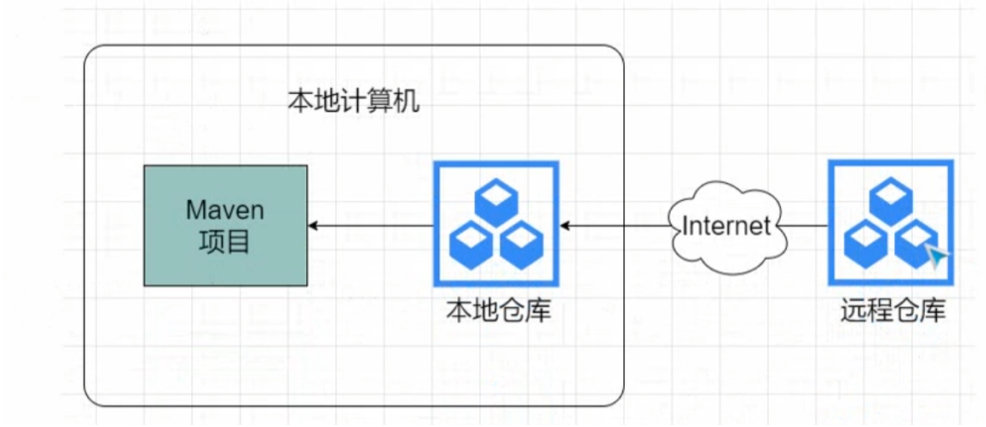
## 1. 准备

### 1.1 下载和安装




#### 安装完成标识

## 1.2 Maven的配置




### 配置本地仓库

settings.xml - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
<settings xmlns="http://maven.apache.org/SETTINGS/1.2.0"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.2.0
https://maven.apache.org/xsd/settings-1.2.0.xsd">
  <!-- localRepository
   | The path to the local repository maven will use to store artifacts.
   |
   | Default: ${user.home}/.m2/repository
  <localRepository>/path/to/local/repo</localRepository>
  -->
<localRepository>D:\Maven\Mavenlib</localRepository>

  <!-- interactiveMode
   | This will determine whether maven prompts you when it needs input. If set to false,
   | maven will use a sensible default value, perhaps based on some other setting, for
   | the parameter in question.
   |
   | Default: true
  <interactiveMode>true</interactiveMode>
  -->

  <!-- offline
   | Determines whether maven should attempt to connect to the network when executing a build.
```

第 55 行，第 1 列 100% Windows (CRLF) UTF-8

### 配置阿里云远程仓库

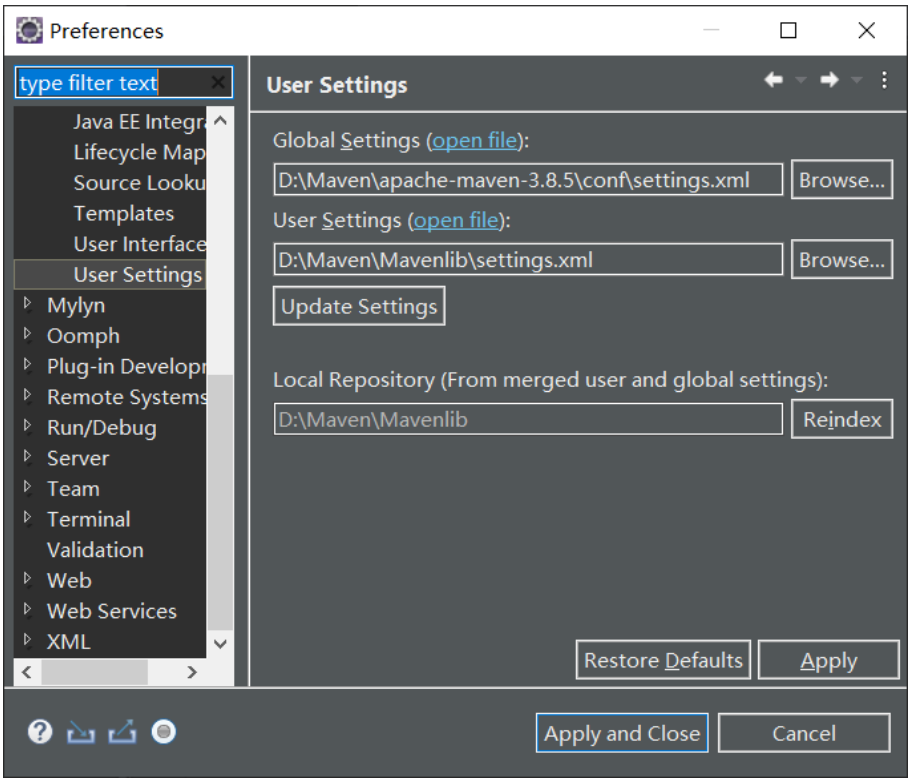
settings.xml - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
   | It works like this: a POM may declare a repository to use in resolving certain artifacts.
   | However, this repository may have problems with heavy traffic at times, so people have
mirrored
   | it to several places.
   |
   | That repository definition will have a unique id, so we can create a mirror reference for that
   | repository, to be used as an alternate download site. The mirror site will be the preferred
   | server for that repository.
   |-->

  <mirrors>
   <mirror>
        <id>nexus-aliyun</id>
        <mirrorOf>central</mirrorOf>
        <name>Nexus aliyun</name>
        <url>http://maven.aliyun.com/nexus/content/groups/public/</url>
    </mirror>
  </mirrors>

  <!-- profiles
   | This is a list of profiles which can be activated in a variety of ways, and which can modify
   | the build process. Profiles provided in the settings.xml are intended to provide local machine-
   | specific paths and repository locations which allow the build to work in the local environment.
```

第 148 行，第 1 列 100% Windows (CRLF) UTF-8

在eclipse进行配置




### 1.3 Maven项目结构

一个使用Maven管理的普通的Java项目，它的目录结构默认如下：

```
a-maven-project
├── pom.xml
├── src
│   ├── main
│   │   ├── java
│   │   └── resources
│   └── test
│       ├── java
│       └── resources
└── target
```

项目的根目录`a-maven-project`是项目名，它有一个项目描述文件`pom.xml`，存放Java源码的目录是`src/main/java`，存放资源文件的目录是`src/main/resources`，存放测试源码的目录是`src/test/java`，存放测试资源的目录是`src/test/resources`，最后，所有编译、打包生成的文件都放在`target`目录里。这些就是一个Maven项目的标准目录结构。

所有的目录结构都是约定好的标准结构，我们千万不要随意修改目录结构。使用标准结构不需要做任何配置，Maven就可以正常使用。

### 1.4 用maven编译Helloworld版的java程序

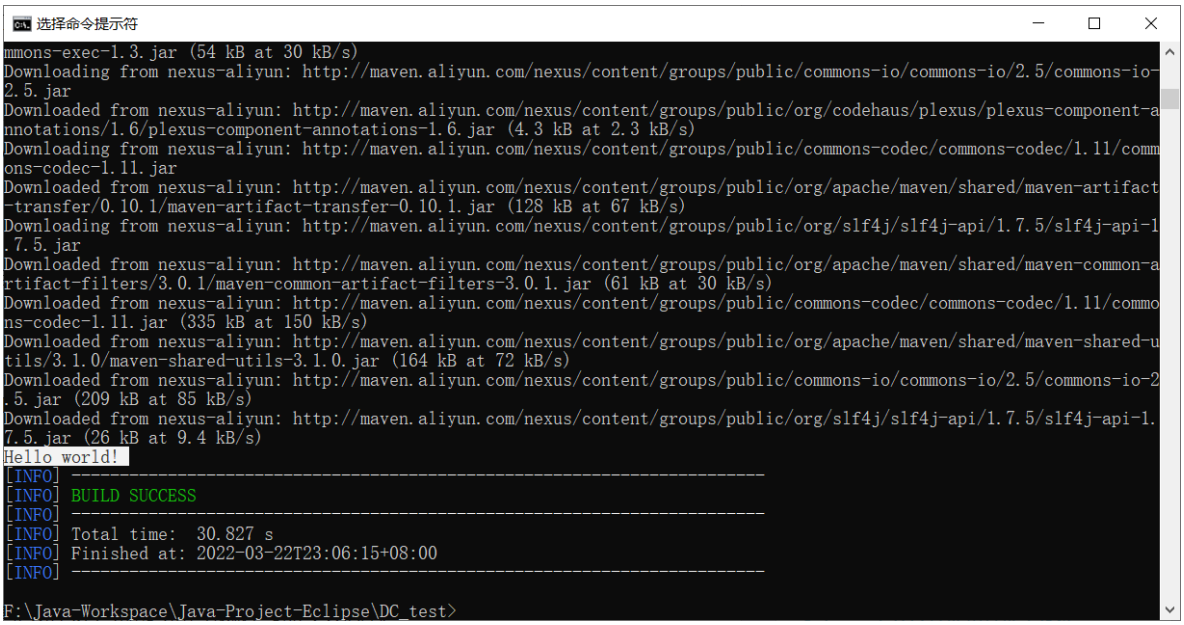
运行下面语句编译:

```
mvn compile
```

运行项目:

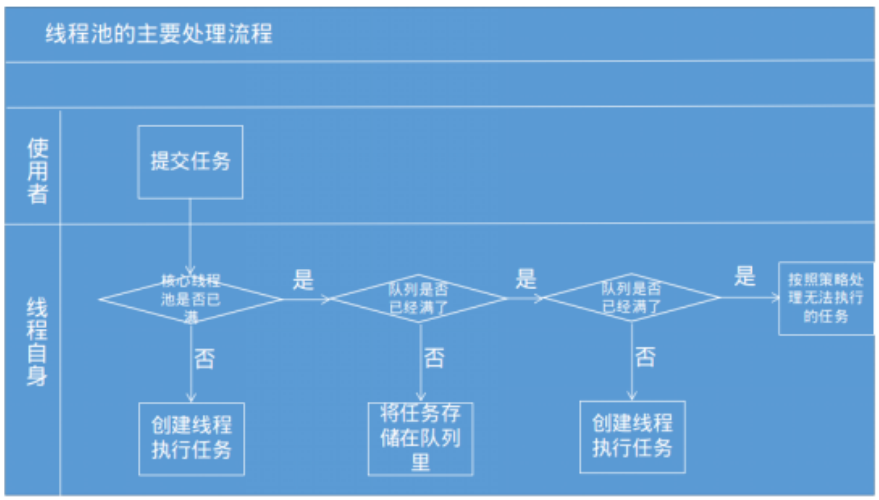
```
mvn exec:java -Dexec.mainClass="hello.HelloWorld"
```

结果如下:




## 2. 线程池版服务器端程序

### 2.1 线程池主要流程



①首先会判断核心线程池里是否有线程可执行，有空闲线程则创建一个线程来执行任务。

②当核心线程池里已经没有线程可执行的时候，此时将任务丢到任务队列中去。

③如果任务队列（有界）也已经满了的话，但运行的线程数小于最大线程池的数量的时候，此时将会新建一个线程用于执行任务，但如果运行的线程数已经达到最大线程池的数量的时候，此时将无法创建线程执行任务。

### 2.2 ThreadPoolExecutor

```
public ThreadPoolExecutor(int corePoolSize,
 int maximumPoolSize,
 long keepAliveTime,
 TimeUnit unit,
 BlockingQueue<Runnable> workQueue,
 ThreadFactory threadFactory,
 RejectedExecutionHandler handler)
```

#### （1）corePoolSize 核心线程数量

即使没有任务执行，核心线程也会一直存活

线程数小于核心线程时，即使有空闲线程，线程池也会创建新线程执行任务

设置 allowCoreThreadTimeout=true 时，核心线程会超时关闭

#### （2）maximumPoolSize 最大线程数

当所有核心线程都在执行任务，且任务队列已满时，线程池会创建新线程执行任务。

当线程数 = maxPoolSize, 且任务队列已满时，

此时添加任务时会触发 RejectedExecutionHandler 进行处理

#### （3）keepAliveTime 线程空闲时间 TimeUnit 表示keepAliveTime的单位

如果线程数>corePoolSize，且有线程空闲时间达到 keepAliveTime 时，线程会销毁，直到线程数量=corePoolSize

如果设置 allowCoreThreadTimeout=true 时，核心线程执行完任务也会销毁直到数量=0

### （4）workQueue 用于缓存任务的阻塞队列

### （5）threadFactory 指定创建线程的工厂

### （6） handler

表示当workQueue已满，且池中的线程数达到maximumPoolSize时，线程池拒绝添加新任务时采取的策略

execute 函数

```
executor.execute(myTask);
```

① 当工作线程数小于核心线程数时，会创建核心线程数；
② 如果工作线程数大于等于核心线程数时，会尝试将任务添加进队列；

ThreadPoolTest.java 运行结果

```
The number of threads in the ThreadPool:1
Task 0 is running!
The number of tasks in the Queue:0
The number of tasks completed:0
The number of threads in the ThreadPool:2
The number of tasks in the Queue:0
The number of tasks completed:0
Task 1 is running!
The number of threads in the ThreadPool:3
The number of tasks in the Queue:0
The number of tasks completed:0
Task 2 is running!
The number of threads in the ThreadPool:4
The number of tasks in the Queue:0
The number of tasks completed:0
Task 3 is running!
The number of threads in the ThreadPool:5
The number of tasks in the Queue:0
The number of tasks completed:0
Task 4 is running!
The number of threads in the ThreadPool:5
The number of tasks in the Queue:1
The number of tasks completed:0
The number of threads in the ThreadPool:5
The number of tasks in the Queue:2
The number of tasks completed:0
The number of threads in the ThreadPool:5
The number of tasks in the Queue:3
The number of tasks completed:0
The number of threads in the ThreadPool:5
The number of tasks in the Queue:4
The number of tasks completed:0
The number of threads in the ThreadPool:5
The number of tasks in the Queue:5
The number of tasks completed:0
The number of threads in the ThreadPool:6
The number of tasks in the Queue:5
The number of tasks completed:0
Task 10 is running!
The number of threads in the ThreadPool:7
The number of tasks in the Queue:5
The number of tasks completed:0
Task 11 is running!
The number of threads in the ThreadPool:8
The number of tasks in the Queue:5
The number of tasks completed:0
Task 12 is running!
The number of threads in the ThreadPool:9
The number of tasks in the Queue:5
The number of tasks completed:0
```

```
Task 13 is running!
The number of threads in the ThreadPool:10
The number of tasks in the Queue:5
The number of tasks completed:0
Task 14 is running!
Task 1 has been done.
Task 13 has been done.
Task 0 has been done.
Task 5 is running!
Task 6 is running!
Task 4 has been done.
Task 11 has been done.
Task 10 has been done.
Task 12 has been done.
Task 9 is running!
Task 8 is running!
Task 7 is running!
Task 14 has been done.
Task 2 has been done.
Task 3 has been done.
Task 7 has been done.
Task 8 has been done.
Task 6 has been done.
Task 5 has been done.
Task 9 has been done.
```

可见 0、1、2、3、4 号任务进入核心线程，5，6，7，8，9号任务进入任务队列，10，11，12，13，14创建为新的线程并满足小于最大线程数量。

### 2.3 ThreadPoolEchoServer 线程池服务器端程序

ServerThread.java

```java
import java.net.*;
import java.io.*;

public class ServerThread extends Thread {

    Socket socket = null;

    public ServerThread(Socket socket) {
        this.socket = socket;
    }

    public void run(){
        InputStream is = null;
        InputStreamReader isr = null;
        BufferedReader br = null;
        OutputStream os = null;
        PrintWriter pw = null;
        try {
            is = socket.getInputStream();
            isr = new InputStreamReader(is);
            br = new BufferedReader(isr);
            os = socket.getOutputStream();
            pw = new PrintWriter(os);

            String info = null;
            InetAddress address= socket.getInetAddress();

            System.out.println(address+"has connect.");

            while((info=br.readLine())!=null) {
                System.out.println("Message from client "+address+" : "+info);
                System.out.println(" ");
                pw.println(info);
                pw.flush();
            }
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            try {
                if(pw!=null) pw.close();
                if(os!=null) os.close();
                if(br!=null) br.close();
                if(isr!=null) isr.close();
                if(is!=null) is.close();
                if(socket!=null) socket.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
```

```
    }

}
```

### ThreadPoolEchoServer.java

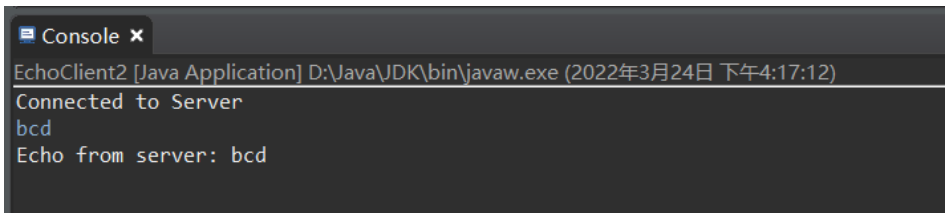
```
import java.net.*;
import java.io.*;
import java.net.ServerSocket;
import java.util.concurrent.*;

public class ThreadPoolEchoServer {

    public static void main(String[] args) throws Exception{
        // TODO Auto-generated method stub

        ServerSocket listenSocket = new ServerSocket(8189);
        Socket socket = null;

        ThreadPoolExecutor executor = new
ThreadPoolExecutor(5,10,200,TimeUnit.MILLISECONDS,
                new ArrayBlockingQueue<Runnable>(5));

        int count = 0;
        System.out.println("Server listening at 8189");

        while(true) {
            socket = listenSocket.accept();
            count++;
            System.out.println("The total number of client is: "+count+" .");
            ServerThread serverThread = new ServerThread(socket);
            executor.execute(serverThread);
            System.out.println("The number of threads in the
ThreadPool:"+executor.getPoolSize());
            System.out.println("The number of tasks in the
Queue:"+executor.getQueue().size());
            System.out.println("The number of tasks
completed:"+executor.getCompletedTaskCount());
        }
    }
}
```

客户端程序不变，所有程序见 **code** folder。

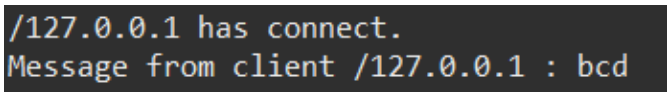
### 2.4 运行测试结果

Socket对应 ip地址 127.0.0.1 和 端口 8259

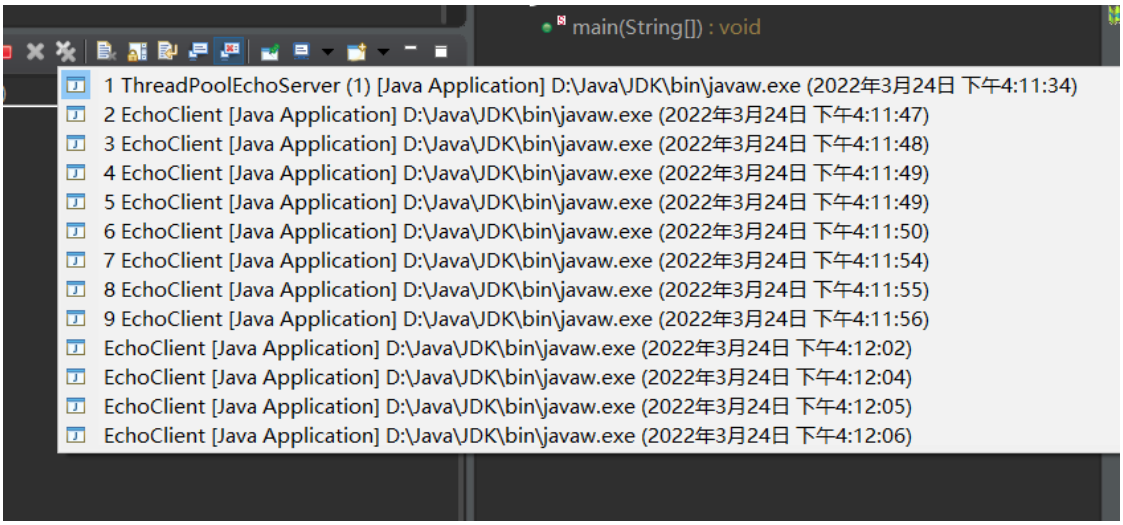
**EchoClient.java 运行结果**



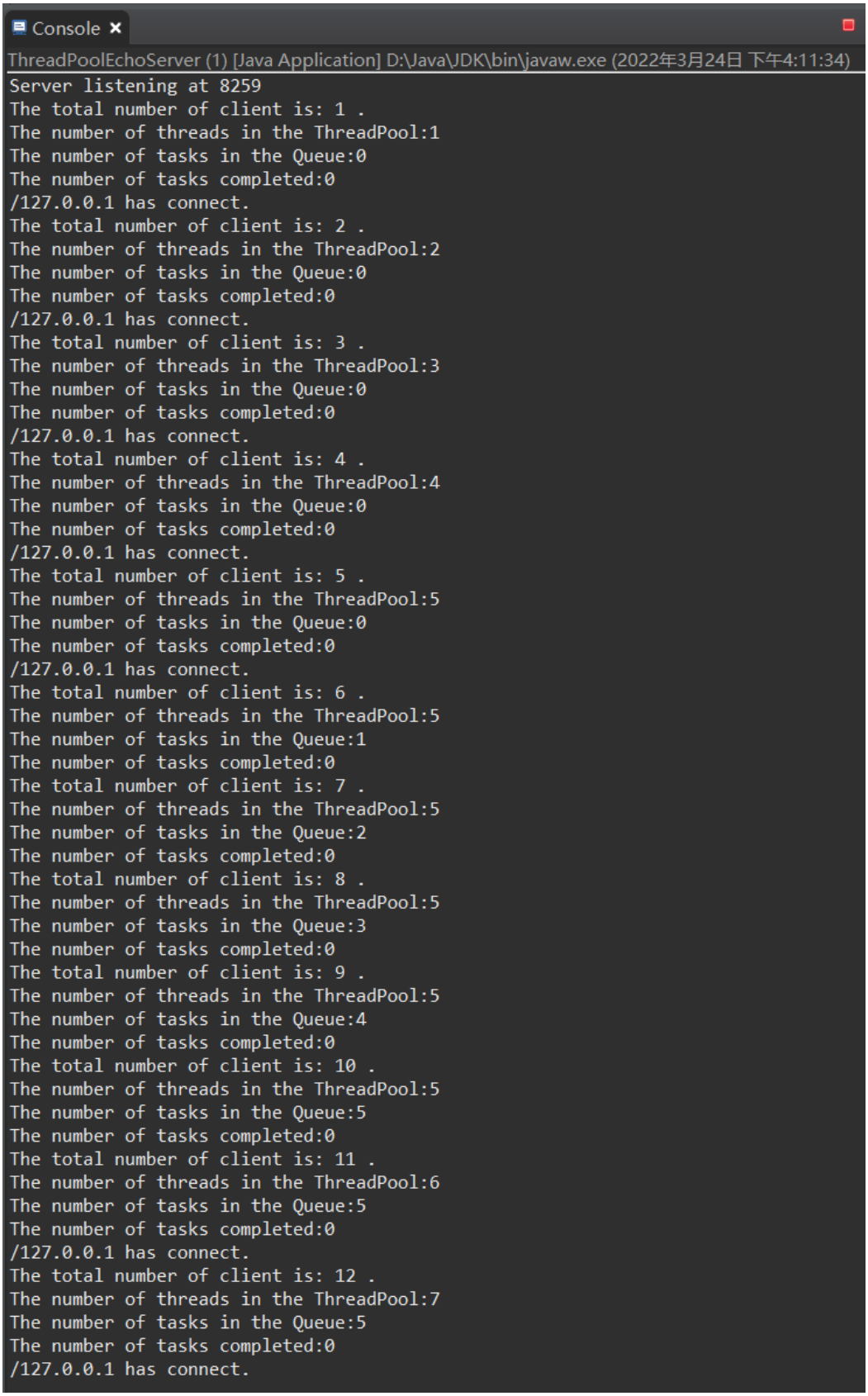

**对应服务器**




**ThreadPoolEchoServer.java 运行结果**




共有12个客户端进程。


Console
ThreadPoolEchoServer (1) [Java Application] D:\Java\JDK\bin\javaw.exe (2022年3月24日 下午4:11:34)
Server listening at 8259
The total number of client is: 1 .
The number of threads in the ThreadPool:1
The number of tasks in the Queue:0
The number of tasks completed:0
/127.0.0.1 has connect.
The total number of client is: 2 .
The number of threads in the ThreadPool:2
The number of tasks in the Queue:0
The number of tasks completed:0
/127.0.0.1 has connect.
The total number of client is: 3 .
The number of threads in the ThreadPool:3
The number of tasks in the Queue:0
The number of tasks completed:0
/127.0.0.1 has connect.
The total number of client is: 4 .
The number of threads in the ThreadPool:4
The number of tasks in the Queue:0
The number of tasks completed:0
/127.0.0.1 has connect.
The total number of client is: 5 .
The number of threads in the ThreadPool:5
The number of tasks in the Queue:0
The number of tasks completed:0
/127.0.0.1 has connect.
The total number of client is: 6 .
The number of threads in the ThreadPool:5
The number of tasks in the Queue:1
The number of tasks completed:0
The total number of client is: 7 .
The number of threads in the ThreadPool:5
The number of tasks in the Queue:2
The number of tasks completed:0
The total number of client is: 8 .
The number of threads in the ThreadPool:5
The number of tasks in the Queue:3
The number of tasks completed:0
The total number of client is: 9 .
The number of threads in the ThreadPool:5
The number of tasks in the Queue:4
The number of tasks completed:0
The total number of client is: 10 .
The number of threads in the ThreadPool:5
The number of tasks in the Queue:5
The number of tasks completed:0
The total number of client is: 11 .
The number of threads in the ThreadPool:6
The number of tasks in the Queue:5
The number of tasks completed:0
/127.0.0.1 has connect.
The total number of client is: 12 .
The number of threads in the ThreadPool:7
The number of tasks in the Queue:5
The number of tasks completed:0
/127.0.0.1 has connect.