准备工作

安装 jdk

参考了 最详细jdk安装以及配置环境（保姆级教程）_jdk环境配置-CSDN博客

安装 maven 并为 maven 设置阿里云镜像仓库

参考了 全站最全Maven下载安装配置教学（2024更新...全版本）建议收藏...赠送IDEA配置Maven教程-CSDN博客

用 maven 编译 Helloworld 版本的 java 程序

用 cmd 在 `DClab` 目录下键入

```
mvn archetype:generate -DgroupId=hoshiuz -DartifactId=helloworld -
DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false
```

即可在 `DClab` 下创建 `helloworld` 项目。
接着用 cmd 进入项目目录，使用 `mvn compile` 来编译项目。
编译完成后，在 cmd 中键入 `java -cp target/classes com.example.App` 来运行程序，可以在 cmd 中看到结果 `Hello World!` 。

题目

1. 基于 Socket 编写支持多客户端并发访问的 Key-Value 存储服务器，客户端可以存储（PUT key value）和检索（GET key）数据。数据存储在内存中（或使用简单的文件持久化）。
   - 服务器支持的指令格式：
     - `PUT key value` ——存储键值对
     - `GET key` ——获取 key 对应的 value
     - `DELETE key` ——删除 key 及其对应的 value
     - `EXIT` ——关闭客户端连接
2. 用 JMeter 测试你的服务器程序的并发性能（平均响应时间和 TPS 指标）。

JMeter 下载安装

参考了 【超详细】Jmeter安装配置详细教程_jmeter安装教程以及jdk环境配置-CSDN博客

设计思路

可以考虑直接在服务端开一个静态的 `HashMap`，由于要实现多客户端并发，而 `HashMap` 是线程不安全的，所以应当使用 `ConcurrentHashMap`。

然后剩余操作就和老师讲的示例差不多，服务端两个 `socket`，一个用来监听，另一个用来通信；客户端只需要用 `socket` 传指令即可。

考虑服务端线程的编写，对于从客户端收到的一行字符串，如何将其识别为指令呢？由于给出的指令类别很少，只有四种，所以可以考虑将字符串 `split(" ")` 后，根据拆分出的第一个单词来分类讨论处理。

但是这样处理会有一个问题：如果指令和操作数之间隔了很多空格的话，识别出的键与值可能会变成空格，所以处理方法为：把 `split(" ")` 改成 `split("\\s+")`，用正则表达式来根据连续的空格拆分

这样剩下的就是一些简单的分类讨论和用 `socket` 通信了。

代码

KeyValueServer 类：

```
// KeyValueServer 类
package lab1;

import java.io.*;
import java.net.*;
import java.util.Map;
import java.util.concurrent.ConcurrentHashMap;

@SuppressWarnings("InfiniteLoopStatement")
public class KeyValueServer {
    static Map<String, String> map = new ConcurrentHashMap<>();
    public static void main(String[] args) throws Exception {
        System.out.println("在端口1234监听。");

        try (ServerSocket listenSocket = new ServerSocket(1234)) {
            while (true) {
                try {
                    Socket socket = listenSocket.accept();
                    ServerThread serverThread = new ServerThread(socket);
                    serverThread.start();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
```

```
    }
}
```

ServerThread 类

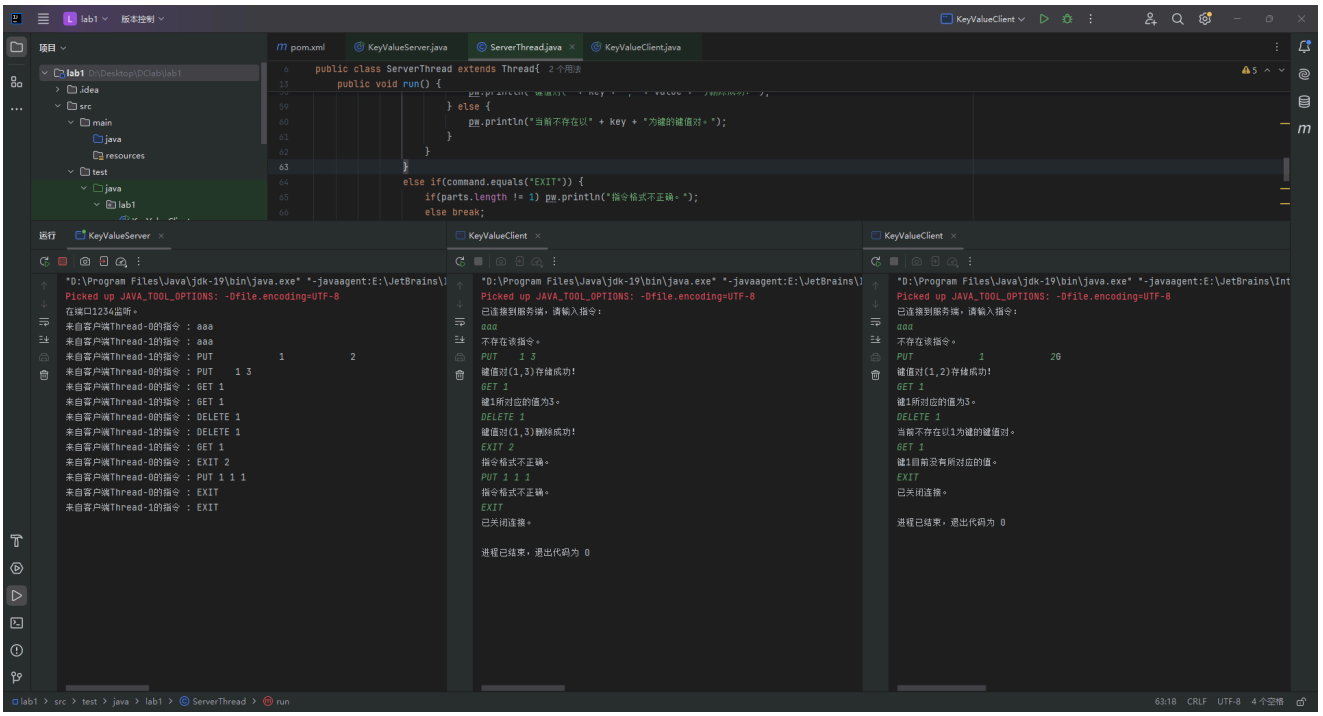
```
// ServerThread 类
package lab1;

import java.io.*;
import java.net.Socket;

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, true);
            String line = null;
            while((line=br.readLine())!=null) {
                System.out.println("来自客户端" + this.getName() + "的指令 : "
+ line);
                String[] parts = line.split("\\s+");
                String command = parts[0];
                if(command.equals("PUT")) {
                    if(parts.length != 3) pw.println("指令格式不正确。");
                    else {
                        String key = parts[1];
                        String value = parts[2];
                        KeyValueServer.map.put(key, value);
                        pw.println("键值对(" + key + "," + value + ")存储成
功! ");
                    }
                }
                else if(command.equals("GET")) {
                    if(parts.length != 2) pw.println("指令格式不正确。");
```

```
                    else {
                        String key = parts[1];
                        String value = KeyValueServer.map.get(key);
                        if (value != null) {
                            pw.println("键" + key + "所对应的值为" + value +
"。");
                        } else {
                            pw.println("键" + key + "目前没有所对应的值。");
                        }
                    }
                }
                else if(command.equals("DELETE")) {
                    if(parts.length != 2) pw.println("指令格式不正确。");
                    else {
                        String key = parts[1];
                        if (KeyValueServer.map.containsKey(key)) {
                            String value = KeyValueServer.map.get(key);
                            KeyValueServer.map.remove(key);
                            pw.println("键值对(" + key + "," + value + ")删除
成功! ");
                        } else {
                            pw.println("当前不存在以" + key + "为键的键值对。");
                        }
                    }
                }
                else if(command.equals("EXIT")) {
                    if(parts.length != 1) pw.println("指令格式不正确。");
                    else break;
                }
                else {
                    pw.println("不存在该指令。");
                }
            }
        } catch(IOException e){
            e.printStackTrace();
        } finally {
            try {
                if(pw!=null) pw.close();
                if(os!=null) os.close();
                if(br!=null) br.close();
                if(is!=null) is.close();
                if(isr!=null) isr.close();
                if(socket!=null) socket.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}
```

KeyValueClient 类

```java
// KeyValueClient 类
package lab1;

import java.io.*;
import java.net.*;
import java.util.Scanner;

public class KeyValueClient {
    public static void main(String[] args) throws Exception {
        BufferedReader stdIn = new BufferedReader(new
InputStreamReader(System.in));

        try(Socket socket = new Socket("127.0.0.1", 1234)) {
            InputStream inStream = socket.getInputStream();
            OutputStream outStream = socket.getOutputStream();
            BufferedReader in = new BufferedReader(new
InputStreamReader(inStream));
            PrintWriter out = new PrintWriter(outStream, true);
            Scanner sc = new Scanner(System.in);

            System.out.println("已连接到服务端，请输入指令：");
            while(true) {
                String command = sc.nextLine();
                out.println(command);
                if(command.equals("EXIT")) {
                    System.out.println("已关闭连接。");
                    break;
                }
                String response = in.readLine();
                if(response!=null) {
                    System.out.println(response);
                }
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

运行截图



指标测试结果

Label	# 样本	平均值 ↑	中位数	90% 百分位	95% 百分位	99% 百分位	最小值	最大值	异常 %	吞吐量	接收 KB/sec	发送 KB/sec
TCP取样器	220552	5	5	14	19	25	0	69	0.02%	7050.7/sec	1745.25	0.00
总体	220552	5	5	14	19	25	0	69	0.02%	7050.7/sec	1745.25	0.00

如图，平均响应时间为 5 毫秒，TPS为7050.7。