



COMP416: Computer Networks

Network Programming Project

Alpha416: Alpha Vantage-API Based Application Layer Protocol

Gülnisa Yıldırım,76401

Instructor: Öznur Özkasap

November,2024

ReadMe

First, follow the path **NPP1/AlphaServer/src/main/java/server** and run the AlphaServer.java file in the server package. Thus, the server will be available to listen on port 8080.

Then follow the path **NPP1/AlphaClient/src/main/java/client** and run the AlphaClient.java file in the client package. Thus, the first client will connect to the server and will be able to send commands. When you run it again, the second client will also connect to the server and thus you will be in a multithread server.

You can run the **"help"** command to get information about what kind of commands can be entered on the client side and get information.

Project Overview

In order to provide a multithreaded server, after the socket creation on the server side, clients are accepted with the accept() method. A **thread** (ClientHandler thread) is opened for each client. In this way, clients can receive service from the server independently of each other.

```
try(ServerSocket serverSocket = new ServerSocket(port_number)){
    System.out.println("Alpha is running on port: " + port_number);

    ExecutorService pool = Executors.newCachedThreadPool();

    while(true) {
        Socket clientSocket = serverSocket.accept();
        System.out.println("A new Client connected.");
        clientSocket.setSoTimeout(timeout);
        pool.execute(new ClientHandler(clientSocket));
    }
}
```

main of server

```
try(BufferedReader in =new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));
    PrintWriter out = new PrintWriter(clientSocket.getOutputStream(), autoFlush: true)){

    String request;
    while((request = in.readLine()) != null) {
        if(request.trim().equalsIgnoreCase( anotherString: "Alpha416 quit")) {
            out.println("Alpha416 ALPHA_208 Success\nDisconnected from Alpha416 Multithreaded server");
            break;
        }
        System.out.println("Client's request:"+request);
        String response= request_parser(request);
        out.println(response);
    }
}
```

run method of ClientHandler

On the server side, requests coming from the client are parsed in the method called **request_parser**. If methods such as “EXC” and “GAS” are entered, they are sent to the methods that handle the operations of that method, i.e. *handleGasPriceRequest*, *handleExchangeRateRequest*. Conditions such as whether the command starts with

“Alpha416” and its length are checked, and since there are some mandatory parameters for the EXC and GAS methods, these are checked before sending the handle methods.

```
private String request_parser(String request) { 1usage

    String[] pieces = request.split(regex: " ");

    //I added this command to see available commands
    if (pieces[0].equalsIgnoreCase( anotherString: "help")) {
        return "Use EXC after 'Alpha416' to get exchange rate between two currencies,\n" +
            "given by mandatory parameters -from and -to\n    optional parameters: \n" +
            "    -from_name : requests full name of from_currency name.\n" +
            "    -to_name : requests full name of to_currency name.\n" +
            "    -refresh : requests the last refreshed date fro the exchange rate.\n\n" +
            "Use GAS after 'Alpha416' to get gas price data for a specific date.\n    optional parameters: \n" +
            "    -change <date> : compares prices on two dates to indicate an increase or decrease." +
            "    -average <date> : calculates the average price between two specified dates.\n\n" +
            "Use -statistics after 'Alpha416' to see how much request that you sent to API.\n\n" +
            "Use QUIT after 'Alpha416' to disconnect. \nend_of_response";
    }

    if (pieces.length < 2 || !pieces[0].equals("Alpha416")) {
        return "Alpha416 ALPHA_400 Invalid Request\n'Error' :Missing parameters.\nend_of_response";
    }
}

String method_type = pieces[1];
if (method_type.equalsIgnoreCase( anotherString: "QUIT")) {
    return "Alpha416 ALPHA_200 Success\nDisconnected from Alpha416 Multithreaded Server.\nend_of_response";
}
if (method_type.equalsIgnoreCase( anotherString: "GAS")) {
    boolean dateCheck= pieces.length>2 && pieces[2].equalsIgnoreCase( anotherString: "-date");
    if(!dateCheck) {
        return "Alpha416 ALPHA_400 Invalid Request\n'Error' :Missing parameter for GAS method.\nend_of_response";
    }return handleGasPriceRequest(pieces);
}
if (method_type.equalsIgnoreCase( anotherString: "EXC")) {
    boolean fromCheck = pieces.length >2 &&pieces[2].equalsIgnoreCase( anotherString: "-from");
    boolean toCheck = pieces.length >4 &&pieces[4].equalsIgnoreCase( anotherString: "-to");

    if (!fromCheck || !toCheck) {
        StringBuilder error = new StringBuilder();

        error.append("Alpha416 ALPHA_400 Invalid Request\n'Error':Missing parameters for EXC method.");
        if (!fromCheck) {
            error.append("Missing -from parameter");
        }
        if (!toCheck) {
            error.append("Missing -to parameter");
        }
        return error.toString().trim()+ "\nend_of_response";
    }
    return handleExchangeRateRequests(pieces);
}
//Added to see total requests to API, it is incremented in the fetchData
if (method_type.equalsIgnoreCase( anotherString: "-statistics")) {
    return generateStringStats();
}
}
```

In the **handleGasPriceRequest** method, the necessary operations are performed according to the parameters and it is made ready to retrieve data. Then, in the **handleGasPriceRequest** method, I call the **fetchData** method with the url I built and fetch the

API data and get the needed values from this data. I prepare the response to be sent to the client with these values. Same operations also done in **handleExchangeRateRequest**.

handleExchangeRateRequest

```
private String handleExchangeRateRequests(String[] pieces) { //usage

    boolean from_name = false;

    boolean to_name = false;

    boolean refresh = false;

    String fromCurr = null;

    String toCurr = null;

    for (int i = 2; i < pieces.length; i++) {

        if(pieces[i].equalsIgnoreCase( anotherString: "-from_name")) {
            from_name = true;
        }
        if(pieces[i].equalsIgnoreCase( anotherString: "-to_name")) {
            to_name = true;
        }
        if(pieces[i].equalsIgnoreCase( anotherString: "-refresh")) {
            refresh = true;
        }
        if(pieces[i].equalsIgnoreCase( anotherString: "-from")) {
            fromCurr = (i+1 < pieces.length) ? pieces[i+1] : null;
            i++;
        }
        if(pieces[i].equalsIgnoreCase( anotherString: "-to")) {
            toCurr = (i+1 < pieces.length) ? pieces[i+1] : null;
            i++;
        }
    }
}
```

```
//Dividing JSON file into pieces to get values and getting values
JSONObject rate_data = json_f.getJSONObject( key: "Realtime Currency Exchange Rate");

String ex_Rate = rate_data.getString( key: "5. Exchange Rate");

String fromCurrName = rate_data.optString( key: "2. From_Currency Name", defaultvalue: "N/A");

String toCurrName = rate_data.optString( key: "4. To_Currency Name", defaultvalue: "N/A");

String last_refreshed = rate_data.getString( key: "6. Last Refreshed");

StringBuilder response = new StringBuilder("Alpha416 ALPHA_200 Success\nExchange rate: "+ex_Rate);

//if user used -from_name parameter we also add info of -from currency's name
if(from_name){
    response.append(", From: ").append(fromCurrName);
    from_name = false;
}

//if user used -to_name parameter we also add info of -from currency's name
if(to_name){
    response.append(", To: ").append(toCurrName);
    to_name = false;
}

//if user -refresh we also add metainfo of last refreshed date of info
if(refresh){
    response.append(", Refresh: ").append(last_refreshed);
    refresh = false;
}

response.append("\nend_of_response");

return response.toString();
```

```

if(fromCurr == null || toCurr == null) {
    return "Alpha416 ALPHA_408 Invalid Request\n'Error':Currency parameters are missing, give currency after -from and -to parameters.\n";
}

try {
    //url build for exchange
    String url = url_api + "?function=CURRENCY_EXCHANGE_RATE&from_currency=" + fromCurr + "&to_currency=" + toCurr + "&apikey=" + API_key;

    String returning_result = fetchData(url);
    String mockData = ""
    {
        "Realtime Currency Exchange Rate": {
            "1. From_Currency Code": "USD",
            "2. From_Currency Name": "United States Dollar",
            "3. To_Currency Code": "EUR",
            "4. To_Currency Name": "Euro",
            "5. Exchange Rate": "0.9325",
            "6. Last Refreshed": "2024-11-09 11:07:01",
            "7. Time Zone": "UTC",
            "8. Bid Price": "0.9323",
            "9. Ask Price": "0.9327"
        }
    }
    ""

    JSONObject json_f = new JSONObject(returning_result);

    if(json_f.has("Error message:")){
        return "Alpha416 Alpha_404 Not Found\n'Error':Currency data not found.\nend_of_response";
    }
}

```

handleGasPriceRequest

```

private String handleGasPriceRequest(String[] pieces) { 1 usage

    boolean average_calculation = false;

    String date = null;

    String date_compare = null;

    for(int i = 2; i < pieces.length; i++) {

        if(pieces[i].equals("-average") && i+1 < pieces.length) {

            date_compare = pieces[i+1];
            average_calculation = true;
            i++;
        }
        else if(pieces[i].equals("-date") && i+1 < pieces.length) {
            date = pieces[i+1];
            i++;
        }
        else if(pieces[i].equals("-change") && i+1 < pieces.length) {
            date_compare = pieces[i+1];
            i++;
        }
    }

    if(date == null) {
        return "Alpha416 ALPHA_400 Invalid Request\n'Error':Missing date parameter.\nend_of_response";
    }

    try {
        //url build for gas price
        String url = url_api + "?function=NATURAL_GAS&interval=monthly&apikey=" + API_key;
        String returning_result = fetchData(url);
    }
}

```

```

// get as JSON file
JSONObject json_f = new JSONObject(returning_result);
JSONArray array_data = json_f.getJSONArray( key: "data");

Double price = null;
Double compare_price = null;

for (int i = 0; i < array_data.length(); i++) {
    //divide into pieces to get values and getting values
    JSONObject obj_data = array_data.getJSONObject(i);
    String date_data = obj_data.getString( key: "date");
    double value_in_date = obj_data.getDouble( key: "value");

    if (date_data.equals(date)) {
        price = value_in_date;
    }
    if (date_compare != null && date_data.equals(date_compare)) {
        compare_price = value_in_date;
    }
}

if (price == null) {
    return "Alpha416 ALPHA_404 Not Found\n'Error':Data is not available.\nend_of_response";
}

StringBuilder response = new StringBuilder("Alpha416 ALPHA_200 Success\nGas price on " + date + ":" + price);

```

```

if (date_compare != null) {
    if(compare_price == null) {
        response.append("Alpha416 ALPHA_404 Not Found\n'Error':Date data to compare is not found.\nend_of_response");
    }
    else {
        if (average_calculation) {
            double average = (price + compare_price) / 2;
            response.append(", Average price: ").append(average);
        } else {
            String change_in_price = (price > compare_price) ? "increased" : "decreased";
            response.append(", Price ").append(change_in_price).append(" compared to ").append(date_compare);
        }
    }
}

response.append("\nend_of_response");

```

In the `fetchData` method, I create an HTTP url connection and retrieve data from the url, convert them to string and return them.

```
private String fetchData(String url_s) throws IOException { 2 usages

    //as my IDE suggested it, I used URI because it said URL is deprecated
    URI uri = URI.create(url_s);
    URL urlObj = uri.toURL();

    //open connection
    HttpURLConnection conn = (HttpURLConnection) urlObj.openConnection();
    conn.setRequestMethod("GET");

    BufferedReader in = new BufferedReader(new InputStreamReader(conn.getInputStream()));

    String inputLine;

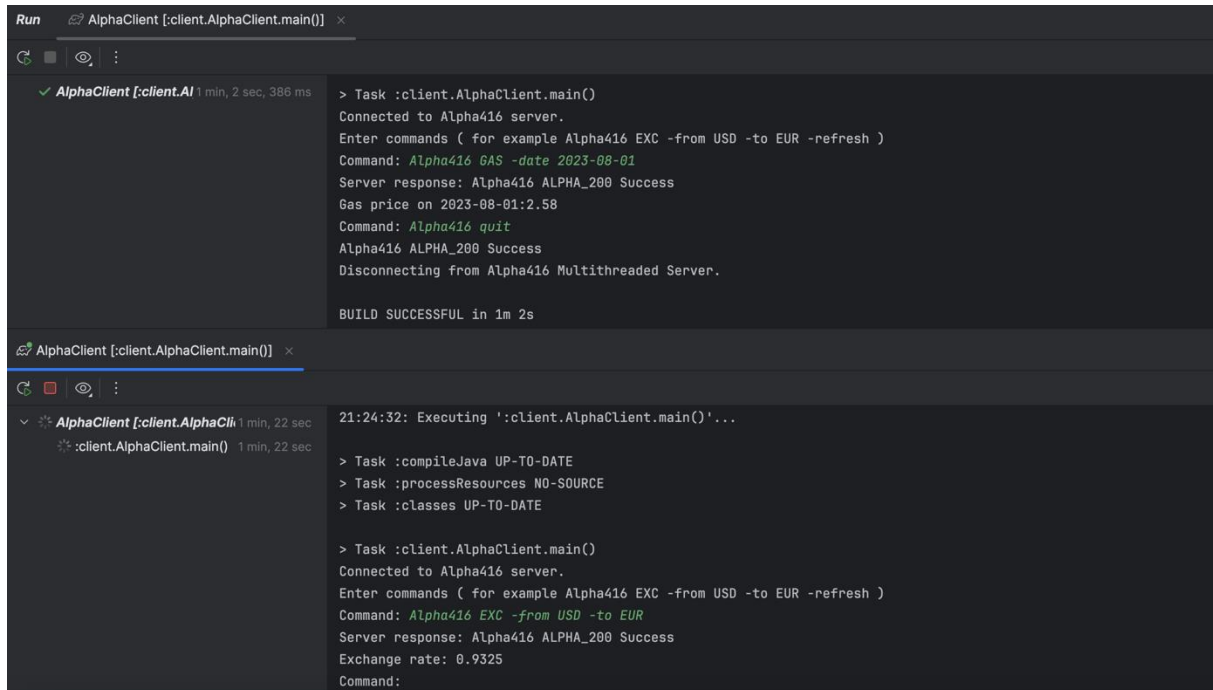
    StringBuilder fetchedData = new StringBuilder();

    while ((inputLine = in.readLine()) != null) {
        fetchedData.append(inputLine);
    }
    System.out.println(fetchedData);
    in.close();
    request_counter.incrementAndGet(); //increment the request counter for -statistics parameter
    return fetchedData.toString();
}
```

In this small **generateStats** function, I print the total number of requests for the "-statistics" command that I added and call this method in the place where I caught the -statistics parameter in **request_parser**. This *request_counter* value, which I initially initialized as an atomic integer, is incremented in **fetchData** for each request made. Thanks to this -statistics parameter, we will be able to see how many requests are made and since the API we are currently working on has a daily limit of 25 requests, we will be able to track this.

```
private String generateStats() { 1 usage
    return "Alpha416 ALPHA200 Success\n" +
        "Total requests: " + request_counter.get()+"\nend_of_response";
}
```

Since test trials will be conducted in the demo, I did not include the tests in the report, but let me add here the request image I made from two clients, one with the GAS method and one with the EXC method.



```
Run AlphaClient [:client.AlphaClient.main()] x
✓ AlphaClient [:client.AlphaClient.main()] 1 min, 2 sec, 386 ms
> Task :client.AlphaClient.main()
Connected to Alpha416 server.
Enter commands ( for example Alpha416 EXC -from USD -to EUR -refresh )
Command: Alpha416 GAS -date 2023-08-01
Server response: Alpha416 ALPHA_200 Success
Gas price on 2023-08-01:2.58
Command: Alpha416 quit
Alpha416 ALPHA_200 Success
Disconnecting from Alpha416 Multithreaded Server.
BUILD SUCCESSFUL in 1m 2s

AlphaClient [:client.AlphaClient.main()] x
21:24:32: Executing ':client.AlphaClient.main()'...
✓ AlphaClient [:client.AlphaClient.main()] 1 min, 22 sec
  ✨ :client.AlphaClient.main() 1 min, 22 sec
> Task :compileJava UP-TO-DATE
> Task :processResources NO-SOURCE
> Task :classes UP-TO-DATE

> Task :client.AlphaClient.main()
Connected to Alpha416 server.
Enter commands ( for example Alpha416 EXC -from USD -to EUR -refresh )
Command: Alpha416 EXC -from USD -to EUR
Server response: Alpha416 ALPHA_200 Success
Exchange rate: 0.9325
Command:
```