# Homework for Java Networking

## Assignment:

Bryce Canyon Hiking Company has decided that they would now like to provide rate quotes as a socket service in addition to the desktop solution from the previous assignment. For this homework, you are to modify your last homework to connect to a socket server to get the quote information instead of using the jar file with utility classes provided in the last assignment. You will still need the library and API to build your UI as before, but all of the quote processing and error feedback will be provided by the server. So for this assignment you will be using your UI client along with a network client to connect to the business logic that computes the rates on the server. Your client should collect user input, package it to be sent on the network to the BHC server and receive the resulting response from the server (which can a message with a successful quote or an error message) and then display the response in your client. **At that point the client must remain running for client submission of subsequent quotes and not shut down after each quote.**

The BHC server is running on web6.jhuep.com at port 20025. It expects data in the form of

```
hike_id:begin_month:begin_day:begin_year:duration:numberHikers (e.g: 0:8:7:2024:3:2)
```

- The *hike_id* is an integer representation of the Hike that can be obtained from the Hike enumeration (it's based on the order in which hikes are defined in the enumeration).
- January is month 1 and December is 12
- Years are in four digits
- All values are separated by colons, ":"

The returned result will be the cost followed by a colon (:) followed by some text. If things go well, you'll get the cost and the text "Quoted Rate", if there is a problem, the cost will by -0.01 and the text will have some explanation (a comma separated list of errors). You will need to parse the return results and display them in your GUI. You are not responsible for the logic of the rate quote, as the server will handle it. All you are doing is designing the GUI client and then displaying the results from the server.

You **must** use a Java application (executable jar) solution for this application that is uploaded to the Canvas server and not the class web server (web3.jhuep.com). You can rely on the server to catch and report errors from your inputs, though you must recognize when errors are returned and display them on the client.

## Test Cases

Here is a small yet specific set of test cases that you must handle correctly in your code. Depending upon your choice of input I will also be testing other forms of errant input as well, like missing fields or bad data (characters for numbers, etc). These cases are provided so you can reasonably test your code before submission. I WILL be testing other cases not provided here to make sure your values do honor constraints enforced by the API and possible errors that the server handles.

| Hike | Duration | Start Of Tour | Number of Hikers | Expected Result |
|---|---|---|---|---|
| Navajo Loop | 4 days | July 1, 2025 | 2 | 320.0 |
| Navajo Loop | 9 days | July 1, 2025 | 1 | 9 is not a valid duration for selected hike<br>The ending date was not defined |
| Navajo Loop | 4 days | January 1, 2025 | 2 | The starting date is out of season<br>The ending date is out of season |
| Navajo Loop | 4 days | July 1, 2023 | 2 | Begin date for hike has already past |
| Navajo Loop | 4 days | Last Valid Date of Season | 1 | The ending date is out of season |
| Navajo Loop | 4 days | July 1, 2100 | 1 | Year falls outside valid range |
| Navajo Loop | 4 days | July 55, 2025 | 1 | Date and month combination is not valid |
| Navajo Loop | 4 days | July 1, 2025 | -3 | Number of hikers [-3] cannot be less than 1 |
| Navajo Loop | 4 days | July 1, 2025 | 500 | Number of hikers [500] is over the limit of # |
| *The following test cases will be dependent upon UI limitations* | | | | |
| Bad text in some or all fields (aka, alpha chars instead of numbers) | | | | Error message provided by you |
| Data missing in fields | | | | Error message provided by you |

## Submission:

A zip file posted to Canvas assignments using the class naming convention from the Syllabus. Your file must include:

1. The executable jar file
2. The entire IDE project (cleaned to not include compiled and built files) using the same name as your zip file above.
3. If you did not use an IDE (Eclipse, IntelliJ, etc) then you MUST provide your source code and supporting libraries (BhcUtils.jar)

## Grading:

Use the following breakdown as a GUIDELINE ONLY for general weighting of different parts of the assignment. Specific grading may not completely fall into the breakdown below:

1. Reasonable code design/Coding Guidelines/General Grading Guidelines 25%
2. Application runs and passes all test cases laid out above 30%
3. Handling Error Messages and Quotes from the Server 20%
4. Design and Implementation of the network client 25%