Assignment 3
Sockets

Version: February 7, 2024

Prerequisites

1. Check out the helper on Slack
2. Lecture videos/or in person
3. Running and understanding the examples listed in Slack helper
4. Setup of a second device (should already be done)
5. Understanding about TCP, protocols

Learning outcomes of this assignment are:

1. Understanding how to work with sockets
2. Understanding how to use TCP sockets
3. Understanding how to create a custom protocol

In your Assignment3 directory: copy both starter codes into the directory. Leading to Assignment3/Assign3-1 and Assignment3/Assign3-2. Please make sure you have this structure, this will be important for us for grading.
The points in the assignment add up to more than 100 points, so there is some room for error and some EC build in. I will cap points at 105 though.

1 Simple Server/Client (30 points)

This is a simple getting started exercise which should not take you too long if you understand the concepts. If you are struggling there is also a code walk-through video on Canvas.
You are given a simple server/client. You do not have to do user input checking, e.g. if the user chooses a valid menu option. We will not put wrong data in the Client when using your program. The client should not do much error handling. The server should do error handling though and check if the request that is received is correct. For example, look at the given "add" request. The client accepts any string the user inputs and sends it to the server. The server will realize if wrong data is sent and will return an error response. The server should not crash if it receives wrong data. See the given Unit Tests to check how you can test wrong requests.
In the README you will find requests/responses described for a simple JSON custom protocol. 3 simple "services" are already implemented with basic error handling and do not need to be modified - so you do not need to add more error handling. Your task is to implement the remaining 2 "services" within the protocol. You need to implement the protocol exactly as given.

Do not change the protocol!!! This is important! This way any person's client should be able to communicate with any other person's server.
Host your server on AWS. Remember to use an open port. When finished, post your server's public IP with the exact Gradle command on Slack in the "servers" channel. Others should be able to test your server and provide feedback. You will get points for commenting on other peoples servers and as soon as you have 2 comments on your server from peers you can take your server off-line.
Commenting on others servers is fine until 2 days after the due date.

10 points for each request that is implemented well and can be called from the Client. 3 points for commenting on other peoples server. 3 points for hosting your server and others being able to access it. 6 points for providing good Unit tests for your two new services. Yes this adds up to more than 30 points so there is a bit of EC included.

2 The Game (70 points)

You are required to implement a client/server game and implement both of these yourself. The client and server will need to communicate with each other through a TCP connection and a JSON custom protocol you design yourself. The starter code already establishes a client/server communication and you can start from there. Watch the code walk-through video if you are struggling and the implementation video if you need more help.
The game: We want to implement a simple game that lets the user guess cities or animals based on an image. The server will have images and hints for each of them and send over an image to the client for them to guess. The client will not know the images, hints, current points in the system or answers. This will always be supplied from the server to the client in messages.
When a game is over the server will ask if the user wants to continue. The client can either continue or decide to quit. This will then end the connection and the server can accept a new client.

Preliminary things

I strongly advise you to work on Git and GitHub, to version control and also to practice with the given examples first. If you work on GitHub make sure your repository is private. When you submit you can either upload your code into the Assignment 3 folder on GitHub through the web browser or just push it if using Git.

What you definitely need (14 points)

1. Structure: you will have to create one program. Use the given starter code as a guideline. I strongly advise you to look at
   a) JavaSimpleSock
   b) JavaSimpleSock2
   c) SimpleCustomProtocol
2. One Gradle file

3. Server should run through $gradle\ runServer\ -Pport=port$ and the client through $gradle\ runClient\ -Pport=port\ -Phost=hostIP$.

4. Provide a README.md which includes:
   a) (1 points) A description of your project and a detailed description of what it does.
   b) (1 points) Include a checklist of the requirements marking if you think you fulfill this requirement or not.
   c) (1 points) An explanation of how we can run the program (hopefully exactly as described in this document).
   d) (5 points) A description of your protocol similar to what you usually see when a protocol is described (also see the first task). You should describe each request and all possible responses (including errors).
   e) (2 points) Include a link to a short screen capture (4-7min) showing you running your game (we also use this to see what works on your end).
   f) (2 points) Explain how you designed your program to be robust (see later under constraints).
   g) (2 points) Explain what you would need to change if you used UDP as protocol

The video you create is important to help us see how to run your programs. If it is NOT included, we reserve the right to deduct from the next section if we cannot easily find the functionality. We will not hunt for things.
We will not debug your code or try to make it run; it either runs through Gradle or it does not. If it does not you will not receive points for the implementation at all. So please start early and reach out to us if you are struggling so that we can help you submit a running program. So do not just try to create as much code as possible but to start slowly and make sure everything runs. I advise a very simple step by step approach. If you are struggling watch the video that I made about starting to implement this assignment, you see how slowly I would advise you to proceed at the beginning.

Requirements (56 points)

Your game will have a single player acting as the client playing the game on the server, which provides our guessing game.
A couple of things before going over the constraints:
You can skip requirements and continue working on the other requirements. Yes, you might lose points but 80% working is better than being stuck at requirement 2. So check if you can continue or simplify things in case you cannot get something to work.
You should work with the UI given in the starter code (the SI session will walk you through the UI). You can also make a more fancy UI if like, but we tried to keep it very simple. If you cannot figure out the UI, you can just work with the command line but you will loose 10 points from the overall if you do not work with the UI (this is not listed below). When working with the command line the image should still be displayed in a frame (see the AdvancedCustomProtocol for how this could be done). Sending an image is always mandatory!

If I say "type X" then X would either be in the command line or in the input field of the UI.
The main part here is to create a Client/Server application. Ensure neither side crashes and is robust even if you might not be able to implement all the functionality. The server should not crash even if the client sends over wrong data. You should also make sure you have a good protocol.
Important: I do not want you to use the Network Util classes from the Advanced Custom Protocol. I want you to send over data on your own with your own setup. You do not have to do it fancy you can stick to doing it the way it was already started in the starter code.
The given code shows some example messages being send, this is not the start of the game, this is just to show you how you can send and receive messages.
Requirements:

1. (4 points) The images need to be send from the server to the client. The client does not have access to the images, so you need to figure out how to actually send the image over the network and not just the path to the image! (skip this if you do not know how)

2. (1 points) When the clients starts up, it should connect to the server. The server will reply by asking for the name of the player. The client should send their name and the server should receive it and greet the client by name. This is already roughly done but you can change the protocol.

3. Evaluations needs to happen on the server side; the client will not know the images, hints, their corresponding answers, the points, or the leader board. The client will always send a request to the server for these and will get a response with the requested data. No real points for this since if this is not done then you do not really do a client/server application. So this will lead to deductions on the parts where these things are used.

4. (3 points) After the name was received the player is in the game and gets a menu allowing them to choose a category to guess, I used cities and animals (you can also come up with other categories if you like), also includes the option of typing "done" or "exit" (explained later).

5. Guessing game (after choosing the category - animal or city)
    - (5 points) When a category is chosen the client sends the request to the server, the server chooses a random image from the category and sends it to the client. The message should also include a "task", e.g. "Guess the city". Display the image on the client with the task.
    - (3 points) At this point the player can either type a guess (e.g. "goat"), type "help" or "done".
    - (7 points) Guess: The client enters a guess and the server must check the guess and respond accordingly. If the answer is correct the player gets 100 points added to their current game. If incorrect 20 points are taken away (points can be negative). After a successful guess they get back to the main menu.
    - (5 points) Help: A help request will be send to the sever. Each image has 3 hints. If all three hints were shown, then no more hints will be provided to the

player. Each time the user asks for a hint they lose 20 points. If the user has used up all their hints and asks for more, then points do not get deducted but a message needs to be displayed informing the player that there are not more hints available.

- (4 points) Done: When typing done, the user quits the current game and gets the points they have currently accumulated. That can even be negative. Have a good message informing the player that the current game is over and inform them how many points they received. The player is then asked to type their name if they want to continue with a new game (basically back to the start). Points are written to the leader board, to be explained later.
- (4 points) To make it more interesting, the server has a 1 in 10 probability of not showing a new image to guess, make the player lose all their points in that particular game, therefore causing the current round to end. I did 0 points if the player had more than 0 points, and points*2 if the points were already negative. This is so the player does not just continue forever, causing them to decide a good time to cash out with all of their current points.
- (4 points) All in button, if the player thinks they know the answer, then they can press this button. If they got the answer right it will give them 200 points, if they get it wrong they will lose 200. Then it goes back to the main menu (same as with the normal guess).

6. (5 points) The leader board will show all players that have played since the server started with their name and points. The server will maintain the leader board and send it to the client when requested. You can assume that the same name is the same player. If a player receives more points than there were originally on the leader board before for this person, overwrite the points. If the person was not on the board yet, add the person even with negative points. Add on extra credit: 3 more points if the leader board is persistent even when the server is restarted (we did not cover this in class yet though).

7. (3 points) The current points should always be displayed (see the points field).

8. (3 points) In the main menu when the player types exit, the client should exit gracefully. Exit should also work when the player is asked for their name but NOT while in game, since that would mean if they are in the negative they could just exit.

9. (3 points) In the main menu when the player types done, the current game will end. This should work the same way as the "done" while they are in guessing mode.

10. (4 points) Your protocol must be robust. If a command that is not understood is sent to the server or arguments are sent that the server does not know, then the protocol should define how these are indicated to the client (imagine someone else wants to implement a client for your server). Your protocol must have headers and optionally payloads. This means, in the context of one logical message, the receiver of the message has to be able to read a header, understand the metadata it provides, and use it to assist with processing a payload (if one is even present). This protocol needs to be described in detail in the README.md.

11. (4 points) Your programs must be robust. If errors occur on either the client or server or if there is a network problem, you have to consider how these should be

handled in the most recoverable and informative way possible. Implement good general error handling and output. Your client/server should not crash even when invalid inputs are provided by the user. If we can crash things in normal play we will remove points. We will not be crazy mean, but we will test a couple of things.

Important

- We will NOT debug or try to fix your code.
- If we cannot find a certain requirement you will not receive points for it. It is in your best interest to have 80% working correctly rather than having everything implemented but not working at all.
- Remember the video where you can show us your functionality!
- If your code does not compile/run then you will not receive points for the coding part of the assignment.
- If your code does not run through Gradle we will not run it and thus not grade it.

Coding

You are in your junior year so good coding practices and appropriate error handling are expected along with a solution that compiles and runs.

Submission

On Canvas add the link to your Assignment 3 folder on GitHub. Yes, the overall points in this assignment add up to more than 100 points; everything over 100 points will be extra credit but will be cut at 105.
As always also submit the zip file of your Assignment3 directory on Canvas.