

# CPI221 – Networking – Number Guessing

50 points

## Topics:

- 4 Pillars of OOP
- Multithreading
- Networking
- Client & Server Application
- Input & Output Streams

## Description

This will be a simple number guessing game. The server will maintain the a “number guessing game” that clients will connect to. The server will pick a number and client will send guesses in. When a client guesses, the server will give them feedback.

## Use the following Guidelines:

- Give identifiers semantic meaning and make them easy to read (examples numStudents, grossPay, etc).
  - Keep identifiers to a reasonably short length.
  - Suggestions: Use upper case for constants. Use title case (first letter is upper case) for classes. Use lower case with uppercase word separators for all other identifiers (variables, methods, objects).
  - Strive for self-commenting code
- Use tabs or spaces to indent code within blocks (code surrounded by braces). This includes classes, methods, and code associated with ifs, switches, and loops. Be consistent with the number of spaces or tabs that you use to indent.
- Use white space to make your program more readable.

## Important Note:

All submitted assignments must begin with the descriptive comment block. To avoid losing trivial points, make sure this comment header is included in every assignment you submit and that it is updated accordingly from assignment to assignment.

```
/*
Name: <your name>
Date: <turn in date>
Description: <short description of code in file/project>
Usage: <how to use your program, including syntax for launching the program (command line arguments)>
*/
```

## Programming Assignment:

### Instructions:

Review the lecture that walked through getting the chat client and server together. The code for this project has not been published, use the lecture as a tutorial and create your own version of the application and confirm that it works.

For this assignment you will create two pieces of software, a server and client.

#### Server:

The server maintains communication with the clients and maintains a number guessing game. When a client successfully guesses the number, the server will announce to all clients that there was a winner, reveal the hidden number, then start a new round.

When a client guesses, the server should respond with one of three messages: "Too Low", "Too High" or "You got it!"

To make it easier, clients should be able to join an already in progress round.

#### Client:

The client is very simple. Before connecting the client should get a name from the user. Upon making the connection to the server, the client should send the user's name to the server. The server will take the name and use the socket/socket resources to build appropriate thread jobs to communicate with that client.

Clients connecting mid-game just join the game already in progress, this should not interrupt the current round.

Then the client should show feedback from the server and take a guess from the client. This should continue until the client enters -1 at which point it should quit.

## Specifications:

### Server

The server should be multi-threaded. It holds a hidden number which the clients are attempting to guess. This hidden number should be an integer between 1 and 1,000,000.

Jobs:

1. Listen for new client connections
  - a. Receive the client's name
  - b. Build a client handler thread
  - c. Start the thread
2. Client handler thread
  - a. Listens for client input
  - b. When receiving client input, compare against hidden number
    - i. Send one of three messages back to the client that guessed:
      1. Your number: <client guess> was Too Low
      2. Your number: <client guess> was Too High
      3. Your number: <client guess> was Correct!!
  - c. Send feedback
    - i. If a client "wins"
      1. Re-randomize the hidden number
      2. Use a `sendToAll()` method to let all clients a victory happened and there is a new number

The Server should have support methods that the Client Handler threads can use such as:

`sendToAll(String)` – This method sends a string to all clients to give mass feedback (i.e. – a new round has started)

`redoNumber()` – This method re-randomizes the hidden number for a new round

`getNumber()` – get the hidden number

## Client

The client should also be multithreaded.

Jobs:

1. Listen to the server for feedback and output it to the user
2. Take input from the user and send it to the server

The client is pretty straight forward.

- Connect to the server
- Prompt the user for their name
- Send the name to the server
- Set up Server Listener thread and run it
  - When it receives a message just output it to the console, nothing fancy
- Set up Client Input thread and run it
  - When the user gives input, send it to the server, prompt the user for their next guess.

### *Extra Credit Opportunities:*

+5 – Make it with a GUI client. Send updates to all users showing what the other users are guessing.

### **Notes and tips:**

- Think about the jobs you need to do for the server and the client
- You can pass a reference to the main server or client object to the constructor of a Runnable
- You can “re-use” the same object reference... built a printwriter? You can pass that object reference to a Runnable and you can store it locally in your server or something like that
- Having a client disconnect from the server can cause all sorts of problems. You can actually use exception handling to figure out a client disconnected and then remove their resources on the server end.

## Grading of Programming Assignment

The Grader will grade your program following these steps:

- (1) Compile the code. If it does not compile a U or F will be given in the Specifications section. This will probably also affect the Efficiency/Stability section.
- (2) The Grader will read your program and give points based on the points allocated to each component, the readability of your code (organization of the code and comments), logic, inclusion of the required functions, and correctness of the implementations of each function.

### Rubric:

	Levels of Achievement						
Criteria	A	B	C	D	E	U	F
Specifications 👍 Weight 50.00%	100 % The program works and meets all of the specifications.	85 % The program works and produces the correct results and displays them correctly. It also meets most of the other specifications.	75 % The program produces mostly correct results but does not display them correctly and/or missing some specifications	65 % The program produces partially correct results, display problems and/or missing specifications	35 % Program compiles and runs and attempts specifications, but several problems exist	20 % Code does not compile and run. Produces excessive incorrect results	0 % Code does not compile. Barely an attempt was made at specifications.
Code Quality 👍 Weight 20.00%	100 % Code is written clearly	85 % Code readability is less	75 % The code is readable only by someone who knows what it is supposed to be doing.	65 % Code is using single letter variables, poorly organized	35 % The code is poorly organized and very difficult to read.	20 % Code uses excessive single letter identifiers. Excessively poorly organized.	0 % Code is incomprehensible
Documentation 👍 Weight 15.00%	100 % Code is very well commented	85 % Commenting is simple but solid	75 % Commenting is severely lacking	65 % Bare minimum commenting	35 % Comments are poor	20 % Only the header comment exists identifying the student.	0 % Non existent
Efficiency 🤖 Weight 15.00%	100 % The code is extremely efficient without sacrificing readability and understanding.	85 % The code is fairly efficient without sacrificing readability and understanding.	75 % The code is brute force but concise.	65 % The code is brute force and unnecessarily long.	35 % The code is huge and appears to be patched together.	20 % The code has created very poor runtimes for much simpler faster algorithms.	0 % Code is incomprehensible

### What to Submit?

You are required to submit your solutions in a compressed format (.zip). Zip all files into a single zip file. Make sure your compressed file is labeled correctly - <lastname>\_<firstname>\_hw4.zip

The compressed file MUST contain the following:

- <lastname>\_<firstname>\_hw4.c / .cpp

If you did extra credit that involved File I/O, please include a sample file. No other files should be in the compressed folder.

If multiple submissions are made, the most recent submission will be graded, even if the assignment is submitted late.

### Where to Submit?

All submissions must be electronically submitted to the respected homework link in the course web page where you downloaded the assignment.

---

## Academic Integrity and Honor Code.

*You are encouraged to cooperate in study group on learning the course materials. However, you may not cooperate on preparing the individual assignments. Anything that you turn in must be your own work: You must write up your own solution with your own understanding. If you use an idea that is found in a book or from other sources, or that was developed by someone else or jointly with some group, make sure you acknowledge the source and/or the names of the persons in the write-up for each problem. When you help your peers, you should never show your work to them. All assignment questions must be asked in the course discussion board. Asking assignment questions or making your assignment available in the public websites before the assignment due will be considered cheating.*

*The instructor and the TA will **CAREFULLY** check any possible proliferation or plagiarism. We will use the document/program comparison tools like MOSS (Measure Of Software Similarity: <http://moss.stanford.edu/>) to check any assignment that you submitted for grading. The Ira A. Fulton Schools of Engineering expect all students to adhere to ASU's policy on Academic Dishonesty. These policies can be found in the Code of Student Conduct:*

*[http://www.asu.edu/studentaffairs/studentlife/judicial/academic\\_integrity.h  
tm](http://www.asu.edu/studentaffairs/studentlife/judicial/academic_integrity.htm)*

*ALL cases of cheating or plagiarism will be handed to the Dean's office. Penalties include a failing grade in the class, a note on your official transcript that shows you were punished for cheating, suspension, expulsion and revocation of already awarded degrees.*

---