COMS/SE 319: Software Construction and User Interface
Spring 2019

LAB Activity 1 – Threads, Data Corruption & Deadlocks, and SERVER-CLIENT

Task 1: Play with Threads

Learning Objectives:

Students will:

- know how to create and run multiple processes and threads on Eclipse.
- know how to find out what is the value of local variables and arguments on a specific stack frame of a specific thread.

Resource:

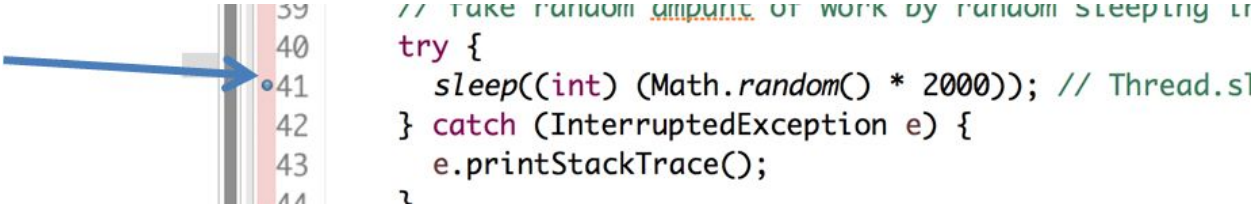
All the links shown in the snapshot below have a wealth of information. Please read first.

https://docs.oracle.com/javase/tutorial/essential/concurrency/procthread.html

- **Step 1:**
  - Download ThreadExample1.java **(inside Lab1_All Sample Codes.zip inside Canvas Home page**).
  - Read the code and see if you understand it.

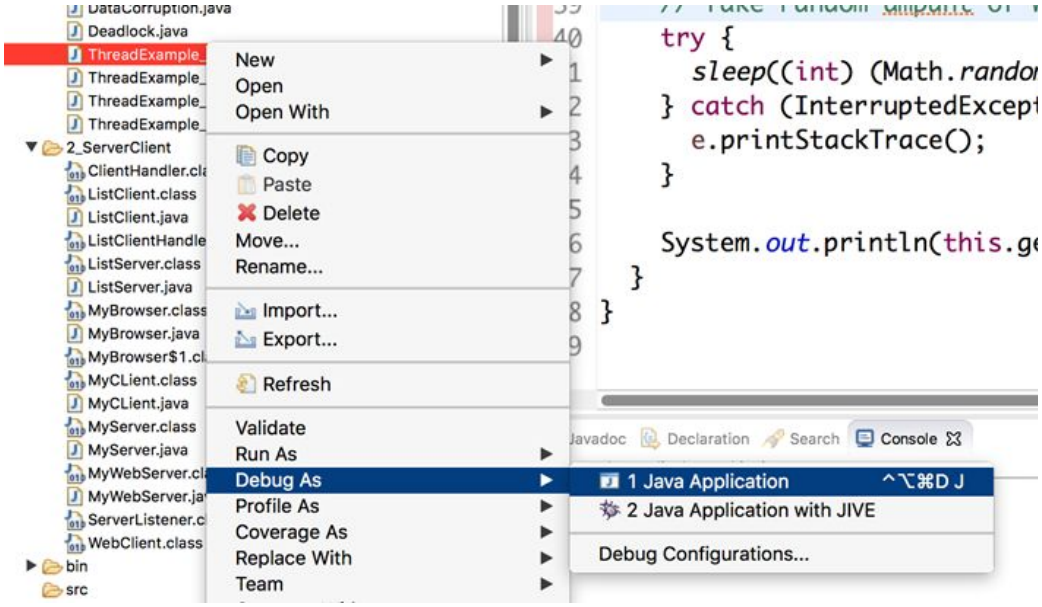
- Run it several times. Are the results the same? Why or why not?

- **Step 2a:**
  - Put a breakpoint in line 41 (i.e. sleep() statement) by double-clicking on the border next to the code. A dot will appear showing breakpoint. Also, put a breakpoint on the System.out.println("King is dead") line

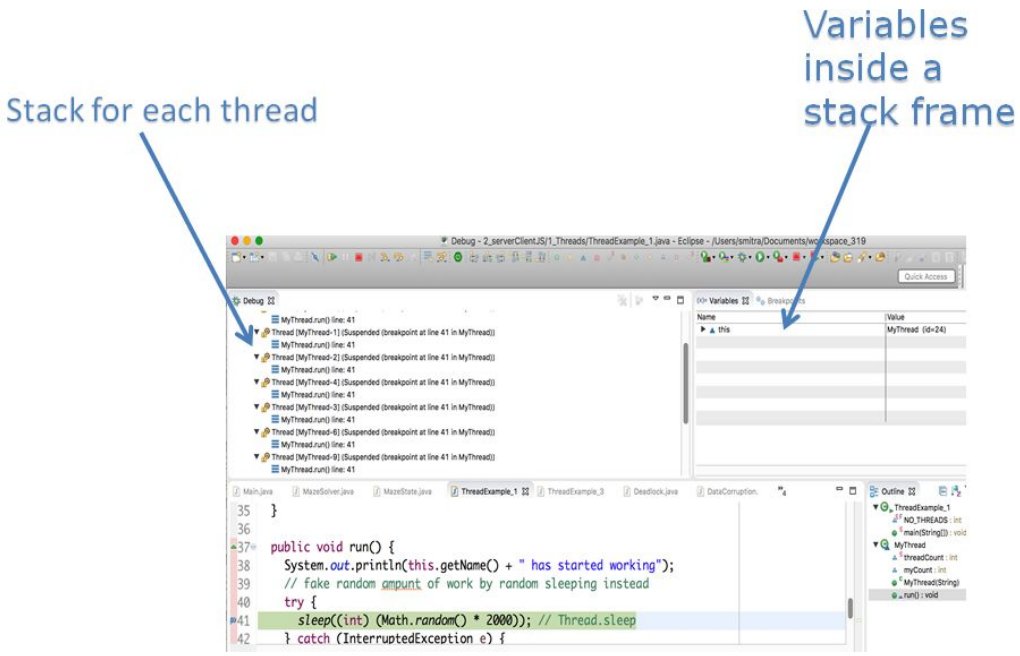


Note: Double clicking again removes the breakpoint.

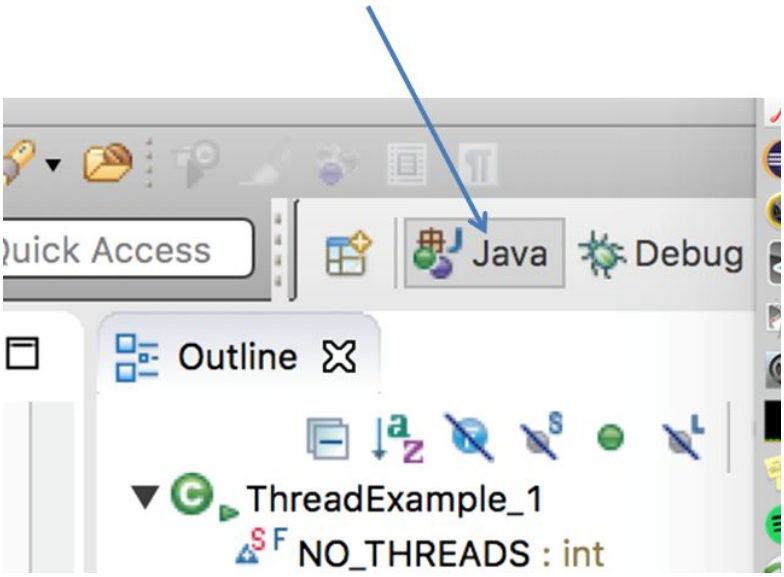
- **Step 2b:**
  - Now run the code in debug mode.



- **Step 2c: Switch to debug perspective:**

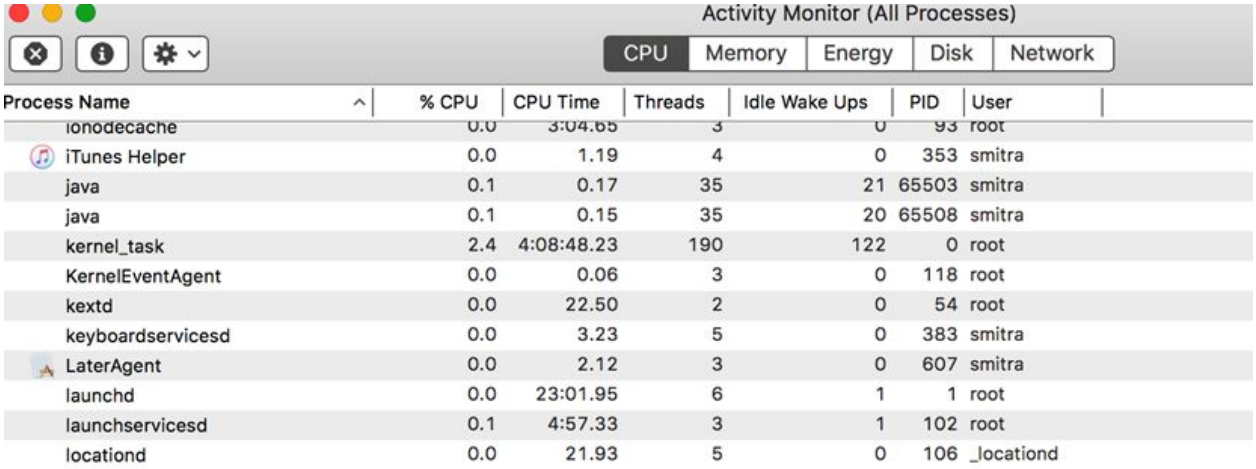


- **Step 2d:**
  - Click on stacks of different threads.
  - They all have only the "run" method on their stack. What is the variable in the stack frame?
  - Find out what is on the stack for the MAIN THREAD. What are the variables on the stack frame?
- **Step 2e: Switch perspectives:**
  - On the top right part of the eclipse window, click on the Java icon to switch to Java perspective.



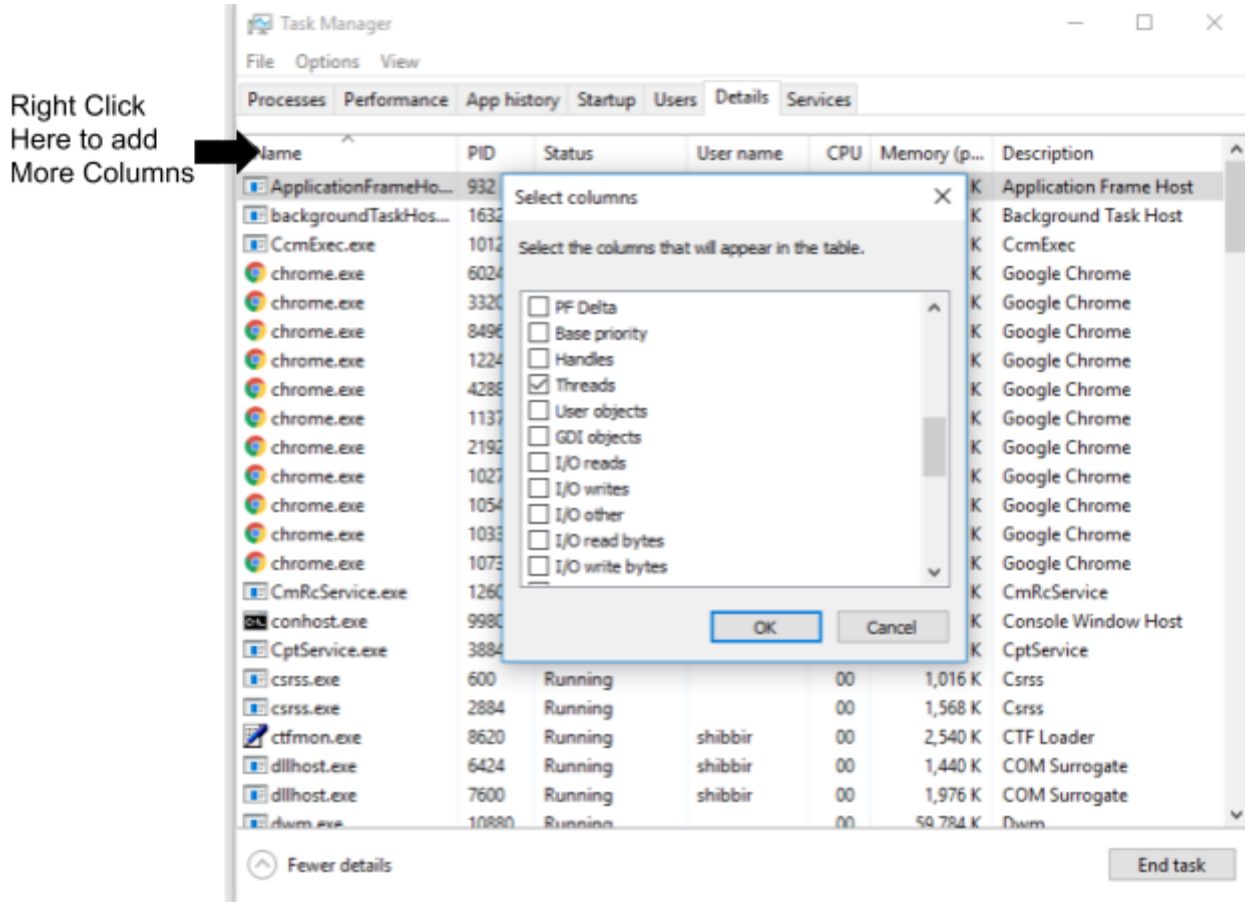
- **Step 3:**
  - Run another process by repeating Step 2b.
  - Run the THIRD process by repeating Step 2b.
  - You can switch between the processes by clicking on **display-selected-console** icon (indicated by yellow arrow). There are three consoles. You can choose any 1. Clicking on the red button will KILL that process and all its threads

- **Step 4:**
  - Using the Operating Systems Task Manager, Identify the three processes. An example for MacOS is shown below.
  - How many threads do these processes have?



Process Name	% CPU	CPU Time	Threads	Idle Wake Ups	PID	User
iTunes Helper	0.0	1.19	4	0	353	smitra
java	0.1	0.17	35	21	65503	smitra
java	0.1	0.15	35	20	65508	smitra
kernel_task	2.4	4:08:48.23	190	122	0	root
KernelEventAgent	0.0	0.06	3	0	118	root
kextd	0.0	22.50	2	0	54	root
keyboardservicesd	0.0	3.23	5	0	383	smitra
LaterAgent	0.0	2.12	3	0	607	smitra
launchd	0.0	23:01.95	6	1	1	root
launchservicesd	0.1	4:57.33	3	1	102	root
locationd	0.0	21.93	5	0	106	_locationd

In Windows,



Task 2: Learn about Data Corruption

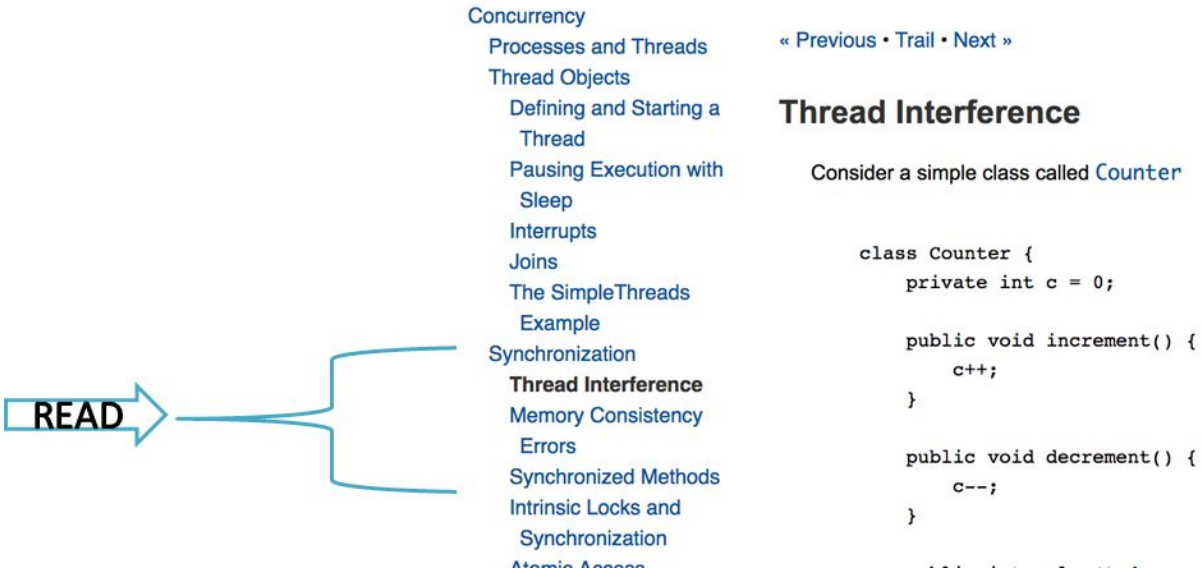
Learning Objectives:

Students will:

- learn about issues to watch out for when running concurrent code.
- in particular, learn about data corruption!

Resource:

All the links shown in the snapshot below have a wealth of information. Please read first.



- **Step 1:**
  - Download DataCorruption.java (inside Lab1_All Sample Codes.zip).
  - Read the code and see if you understand it.
  - Run it several times. Are the results the same? Why or why not?

Note that debugging concurrent code is really hard because it is hard to understand what is going on! Sometimes the code will work ok and sometimes it will not work ok.

- **Step 2:**
  - Explain why the results were not the same in Step1.
  - Hint: Reading the Oracle tutorials may help.
- **Step 3:**
  - Fix the problem so that the answer is ALWAYS correct.
  - Hint1: Reading the Oracle tutorials may help.
  - Hint2: Use synchronized ???
- **Step 4:**
  - Take a look at https://docs.oracle.com/javase/tutorial/essential/concurrency/collections.html

Task 3: Learn about DEADLOCKS

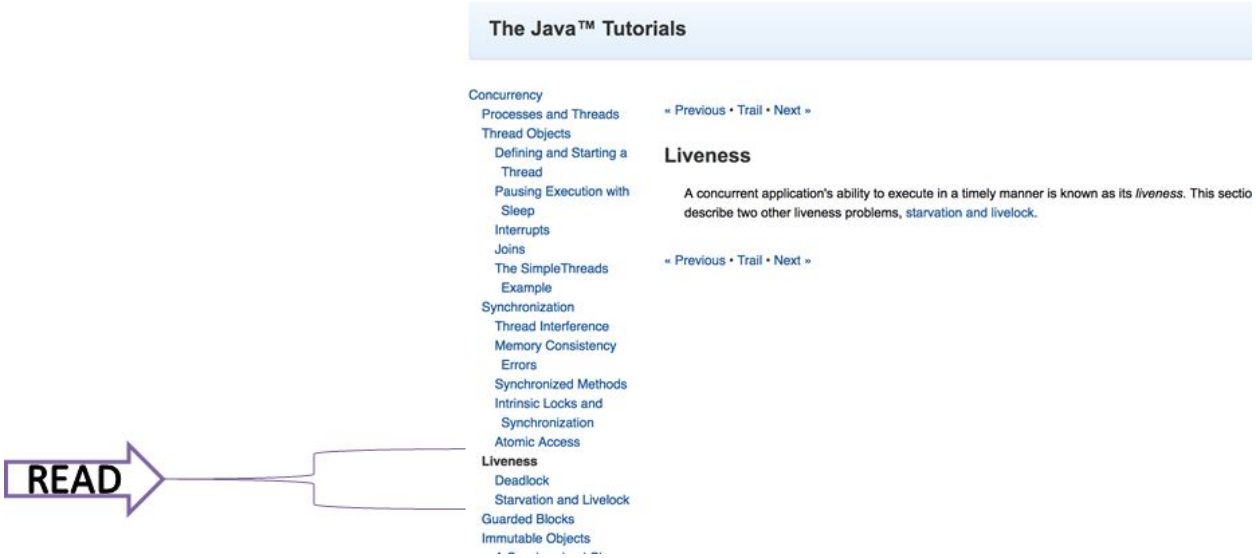
Learning Objectives:

Students will:

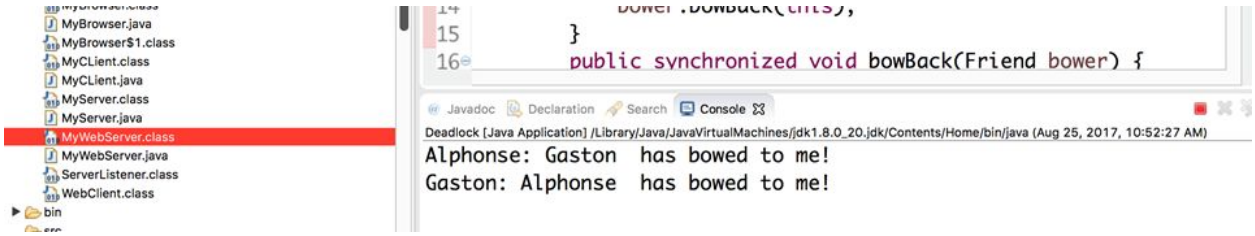
- learn about issues to watch out for when running concurrent code.
- in particular, learn about deadlocks!

Resource:

All the links shown in the snapshot below have a wealth of information. Please read first.(https://docs.oracle.com/javase/tutorial/essential/concurrency/procthread.html)



- **Step 1:**
  - Download Deadlock.java (inside Lab1_All Sample Codes.zip).
  - Read the code and see if you understand it.
  - Run it several times. Does the program stop?



Note: You can always force a program to stop by clicking on the red square button on the console

- **Step 2:**
    - Comment out the print statement in the bow() method.
    - Run it several times. Does the program stop?

- **Step 3:**
    - If you want to know more about concurrency issues do read the other links in the Java tutorial.

TRY IT YOURSELF

Questions: Task 1:

- Run it several times. Are the results the same? Why or why not?
- Increase the value of NO_THREADS by powers of 10 until no more threads can be created. How many maximum numbers of threads were you able to create?
- Find out what are the variables in the stack frame of a thread.

Questions: Task 2:

- Run it several times. Are the results the same? Why or why not?

SERVER-CLIENT

Task 1: Play with Simple Server/Client

Learning Objectives:

Students will:

- learn how to write server-side code
- learn how to write client-side code
- learn how to run server and client codes
- know about socket and port# and IP
- know about typical errors

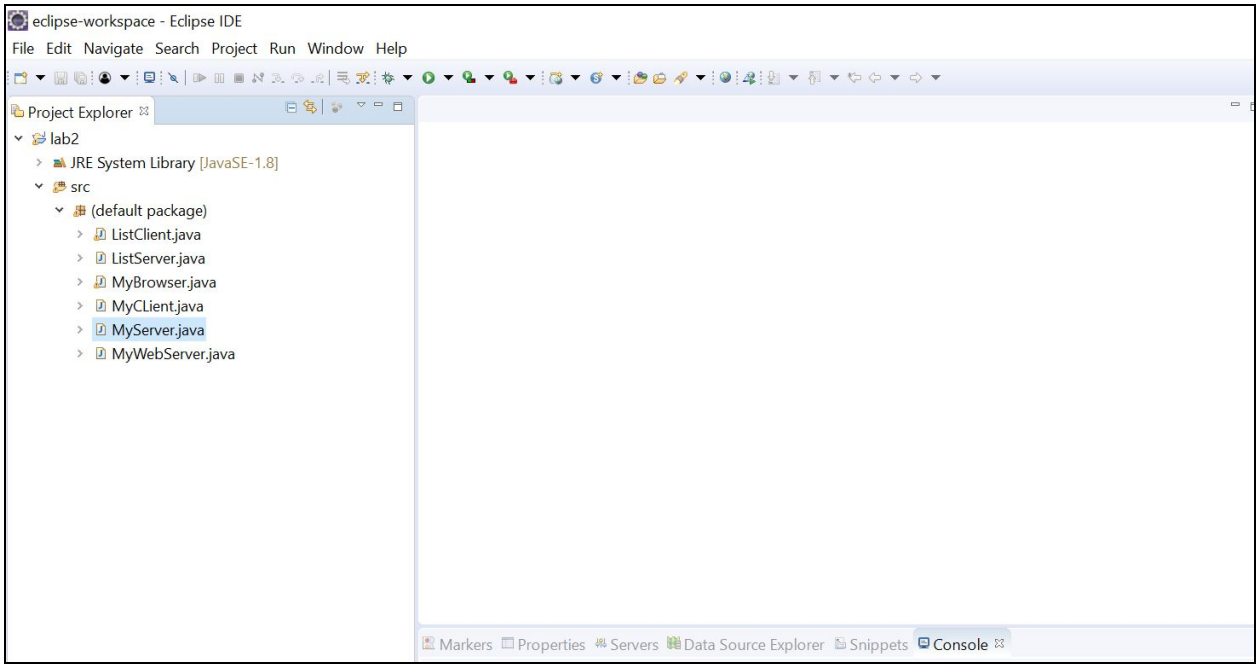
Resource:

All the links shown in the snapshot below have a wealth of information. Please read first.

https://docs.oracle.com/javase/tutorial/networking/sockets/index.html

- **<u>Step 1:</u>**

  Download and unzip **Lab1_All Sample Codes.zip** from Canvas Home Page**,** Create a java project in Eclipse (File>New>Project>Java Project>Provide Name and click Finish) and then expand the newly created project and select "src" and paste all source codes in that like below.



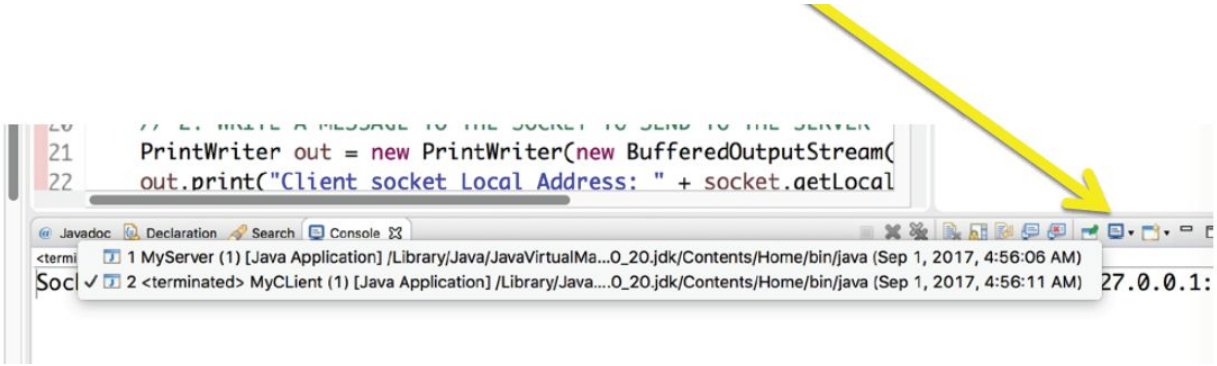
  - READ MyServer.java

- **What is the port number for the service provided by this server? [Please answer it in Canvas Quiz]**
- Does the server send any data to the client via the socket?
- Why is the server in a "forever" loop?

- **Step 2:**
  - Run MyServer.java (do not kill it)
  - Run MyServer.java again!
  - What message do you get on console? Do you understand what is happening here?

- **Step 3:**
  - Read MyClient.java
  - Does the Client read any data from the server?
  - Does the Client send any data to the server?
  - What is the complete address of the server program that the client program connects to? (You need an IP address AND a port#).

- **Step 4::**
  - Run MyClient.java
  - Make sure to switch and take a look at both client and server consoles.



  - What is the CLIENT port number that the server connects to?
- **Step 5:**
  - Run MyClient.java again
  - Run MyClient.java again!
  - Run MyClient.java again!
- **Step 6:**
  - On Comment out line 24 (i.e. out.flush()) of MyClient.java code.

- Run MyServer.java again!
- Run MyClient.java again.
- What message do you get on the server console?
- Note: you may be getting an error anyway. Add out.close() to get rid of error after line 24.
- Explain the reason for the error.

Task 2: Play with Server/Client

Learning Objectives:

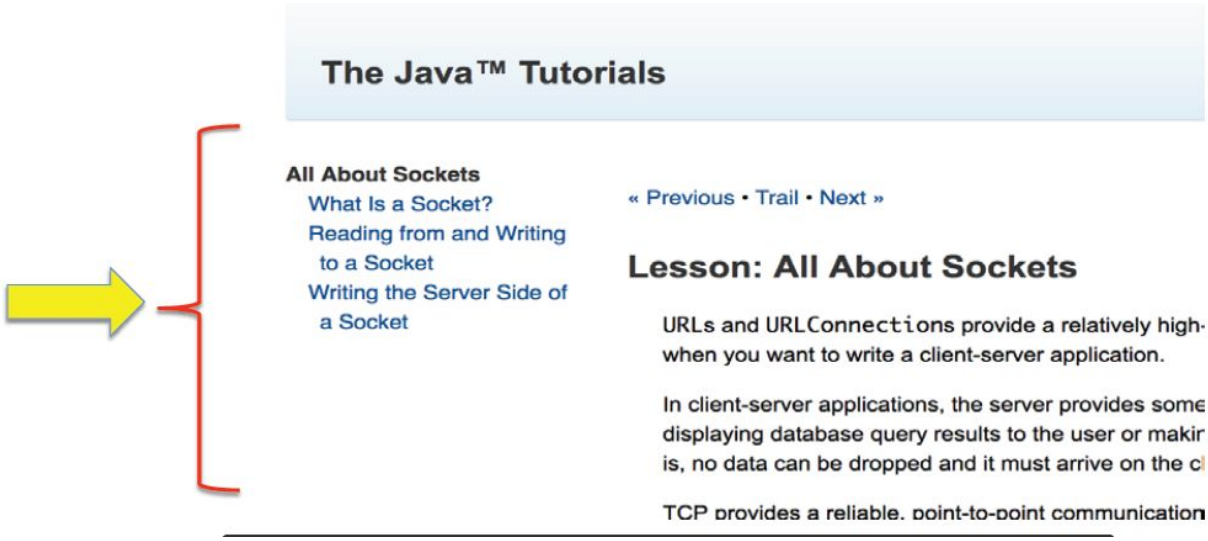
Students will:

- learn how servers and clients can "talk" to each other. i.e. both must agree on a "protocol"

Resource:

All the links shown in the snapshot below have a wealth of information. Please read first.

https://docs.oracle.com/javase/tutorial/networking/sockets/index.html



- **Step 1:**
  - READ ListServer.java
  - Does the server send any data to clients?

- **Does the server get any data from clients? [Please answer it in Canvas Quiz]**

- **Step 2:**
  - Read ListClient.java
  - Why does the client start a thread?
  - In which thread does the client write to the server?
  - When will the client exit?

- **Step 3:**
  - Run ListServer.java
  - Run ListClient.java one or more times.
  - Check the results of server and clients (by switching console window).
  - Describe the sequence of read/writes between server/client.

- **Protocol:**
  - The protocol is an understanding between server and client on how they will communicate.
  - In this example, the server expects three messages from the client and then sends a message to the client. Similarly, the client has expectations from the client too!

Task 3: PLAY WITH WEBSERVER/ WEBCLIENT

Learning Objectives:

Students will:

- learn a bit about the protocol followed by WEB-servers and WEB-clients. This protocol is known as the HTTP protocol.

- **Step 1:**
  - READ MyWebServer.java
  - Note: normal web servers use port#80. However, our WebServer uses port#4444
  - What is the sequence of messages the server is sending to a client on receiving a GET message?
- **Step 2:**
  - Read MyBrowser.java
  - What is the browser sending to the server?

- **Step 3:**
    - Run MyWebServer.java
    - Run MyBrowser.java
    - It will open up a display window. Also, on the console, you will need to enter the hostname and then when prompted, enter the port number (4444)
    - **Take a look at what is displayed on the MyBrowser window? [Please answer it in Canvas Quiz]**

- **Step 4:**
    - Run MyBrowser.java again
    - This time, use hostname www.google.com and port number 80
    - Describe what is displayed on the MyBrowser window?