COMS/SE 319: Software Construction and User Interface
Fall 2018

LAB Activity 1 – Threads, Data Corruption & Deadlocks

Task 1: Play with Threads

Learning Objectives:

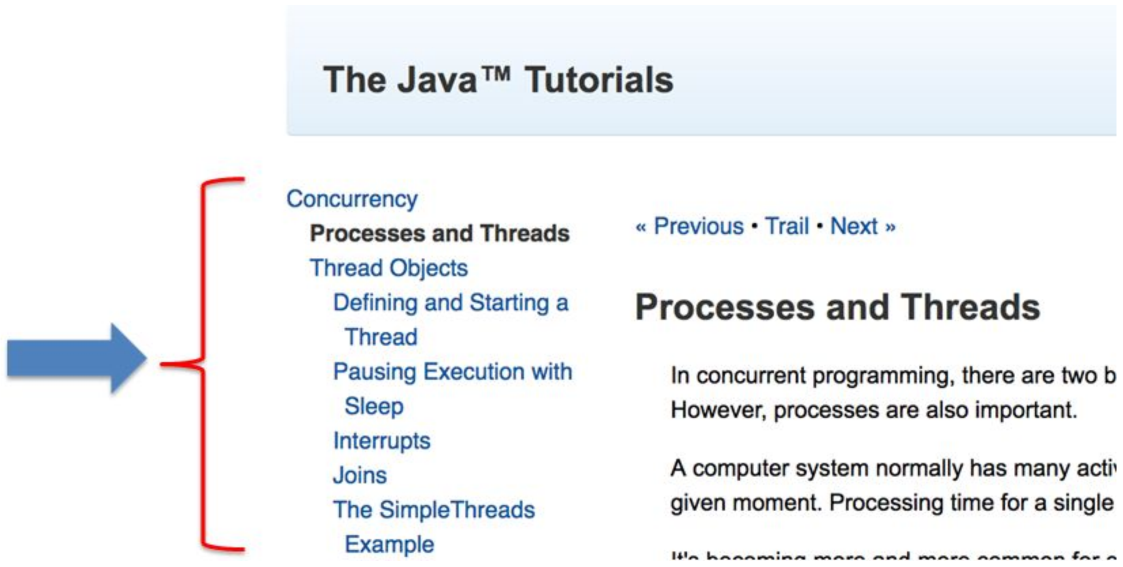
Students will:

- know how to create and run multiple processes and threads on Eclipse.
- know how to find out what is 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



The Java™ Tutorials

Concurrency
Processes and Threads
Thread Objects
Defining and Starting a Thread
Pausing Execution with Sleep
Interrupts
Joins
The SimpleThreads Example

« Previous • Trail • Next »

Processes and Threads

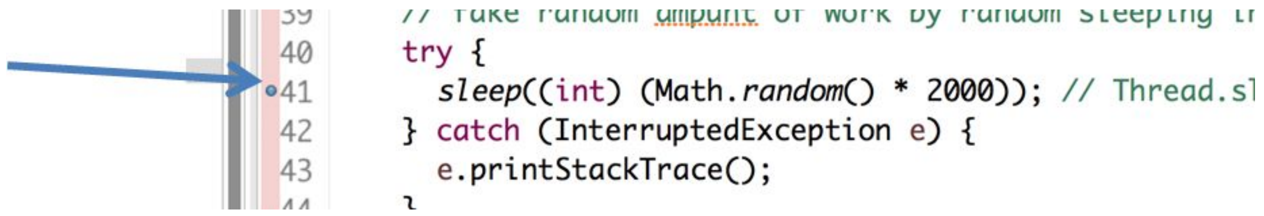
In concurrent programming, there are two b
However, processes are also important.

A computer system normally has many acti
given moment. Processing time for a single

- **Step 1:**
  - Download ThreadExample1.java (inside Sample Codes_Activity 01.zip).
  - Read the code and see if you understand it.
  - 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 number of threads were you able to create?

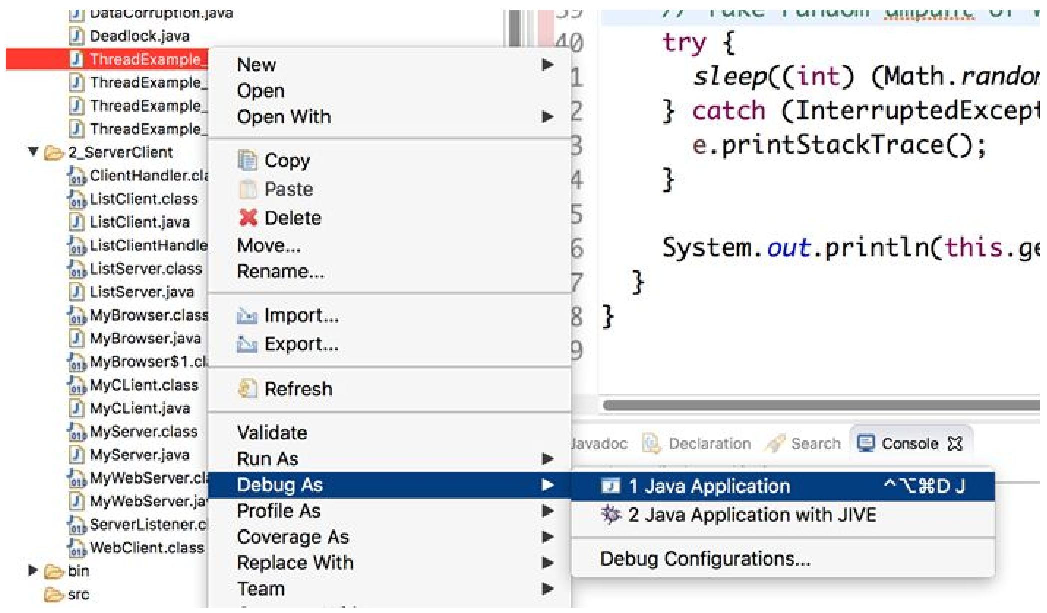
- **Step 2a:**
    - Put a breakpoint in line 41 (i.e. sleep() statement) by double clicking on the border next to the code. A dot will appear showing breakpoint. Also, put a breakpoint on the System.out.println("King is dead") line



```
39      // Take random amount of work by random sleeping th
40      try {
41        sleep((int) (Math.random() * 2000)); // Thread.sl
42      } catch (InterruptedException e) {
43        e.printStackTrace();
44      }
```

Note: Double clicking again removes the breakpoint.

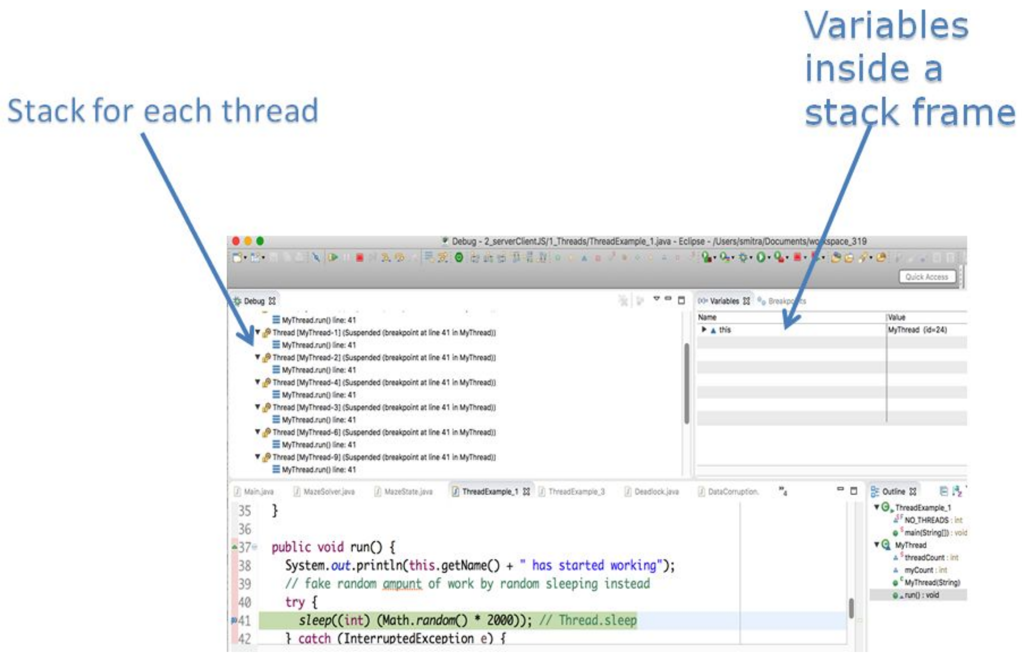
- **Step 2b:**
    - Now run the code in debug mode.



```
      // Take random amount of w
      try {
        sleep((int) (Math.rando
      } catch (InterruptedExcept
        e.printStackTrace();
      }

      System.out.println(this.ge
    }
}
```

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



Debug - 2_serverClientJS/1_Threads/ThreadExample_1.java - Eclipse - /Users/smitra/Documents/workspace_319

```
MyThread.run() line: 41
Thread [MyThread-1] (Suspended (breakpoint at line 41 in MyThread))
    MyThread.run() line: 41
Thread [MyThread-2] (Suspended (breakpoint at line 41 in MyThread))
    MyThread.run() line: 41
Thread [MyThread-4] (Suspended (breakpoint at line 41 in MyThread))
    MyThread.run() line: 41
Thread [MyThread-3] (Suspended (breakpoint at line 41 in MyThread))
    MyThread.run() line: 41
Thread [MyThread-6] (Suspended (breakpoint at line 41 in MyThread))
    MyThread.run() line: 41
Thread [MyThread-9] (Suspended (breakpoint at line 41 in MyThread))
    MyThread.run() line: 41
```

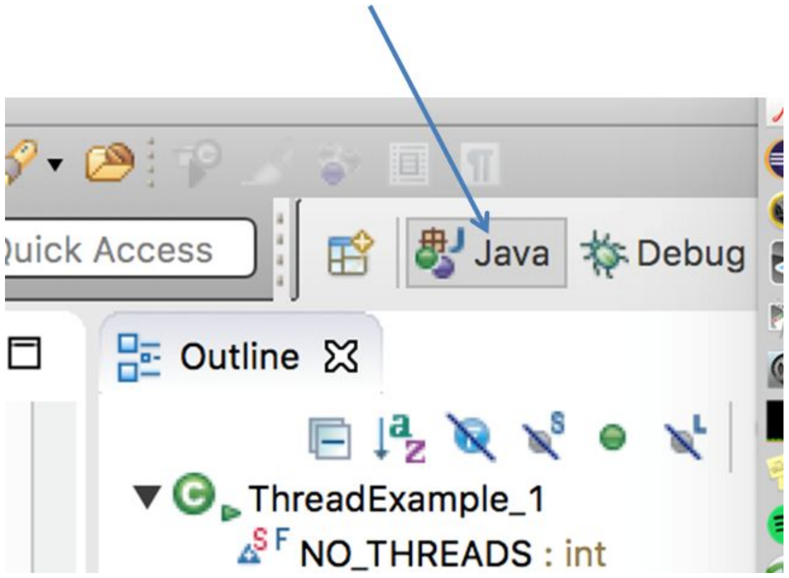
Name	Value
this	MyThread (id=24)

Main.java | MazeSolver.java | MazeState.java | ThreadExample_1 | ThreadExample_3 | Deadlock.java | DataCorruption.

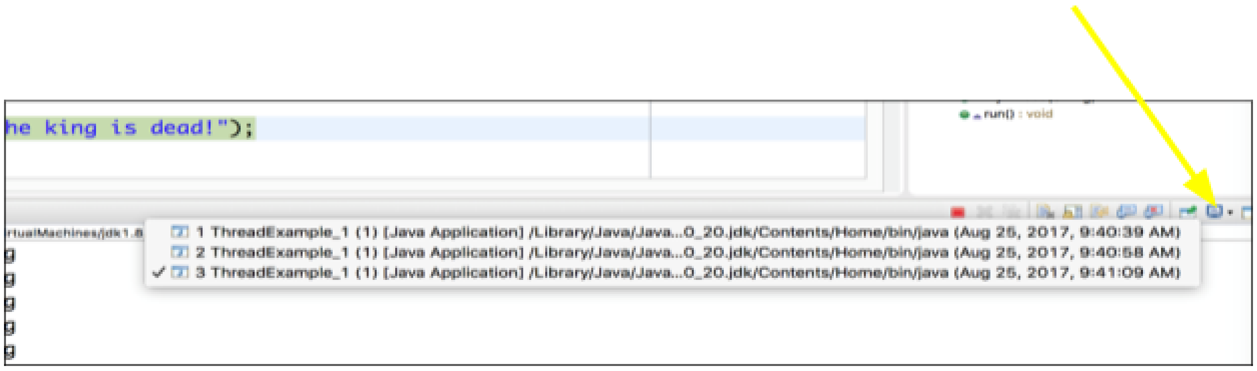
```
35  }
36
37  public void run() {
38    System.out.println(this.getName() + " has started working");
39    // fake random ampunt of work by random sleeping instead
40    try {
41      sleep((int) (Math.random() * 2000)); // Thread.sleep
42    } catch (InterruptedException e) {
```

```
ThreadExample_1
    NO_THREADS : int
    main(String[]) : void
MyThread
    threadCount : int
    myCount : int
    MyThread(String)
    run() : void
```

- **<u>Step 2d:</u>**
    - 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?
- **<u>Step 2e: Switch perspectives:</u>**
    - On the top right part of the eclipse window, click on the Java icon to switch to Java perspective.



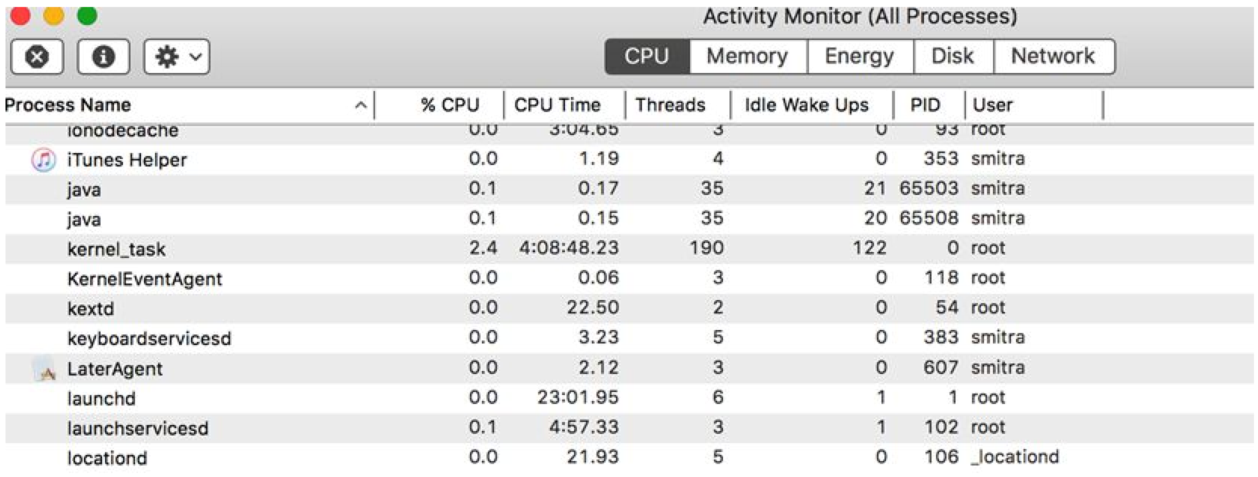
- **Step 3:**
    - Run another process by repeating Step 2b.
    - Run a THIRD process by repeating Step 2b.
    - You can switch between the processes by clicking on **display-selected-console** icon (indicated by yellow arrow). There are three consoles. You can choose any one. Clicking on the red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



```
he king is dead!");
```

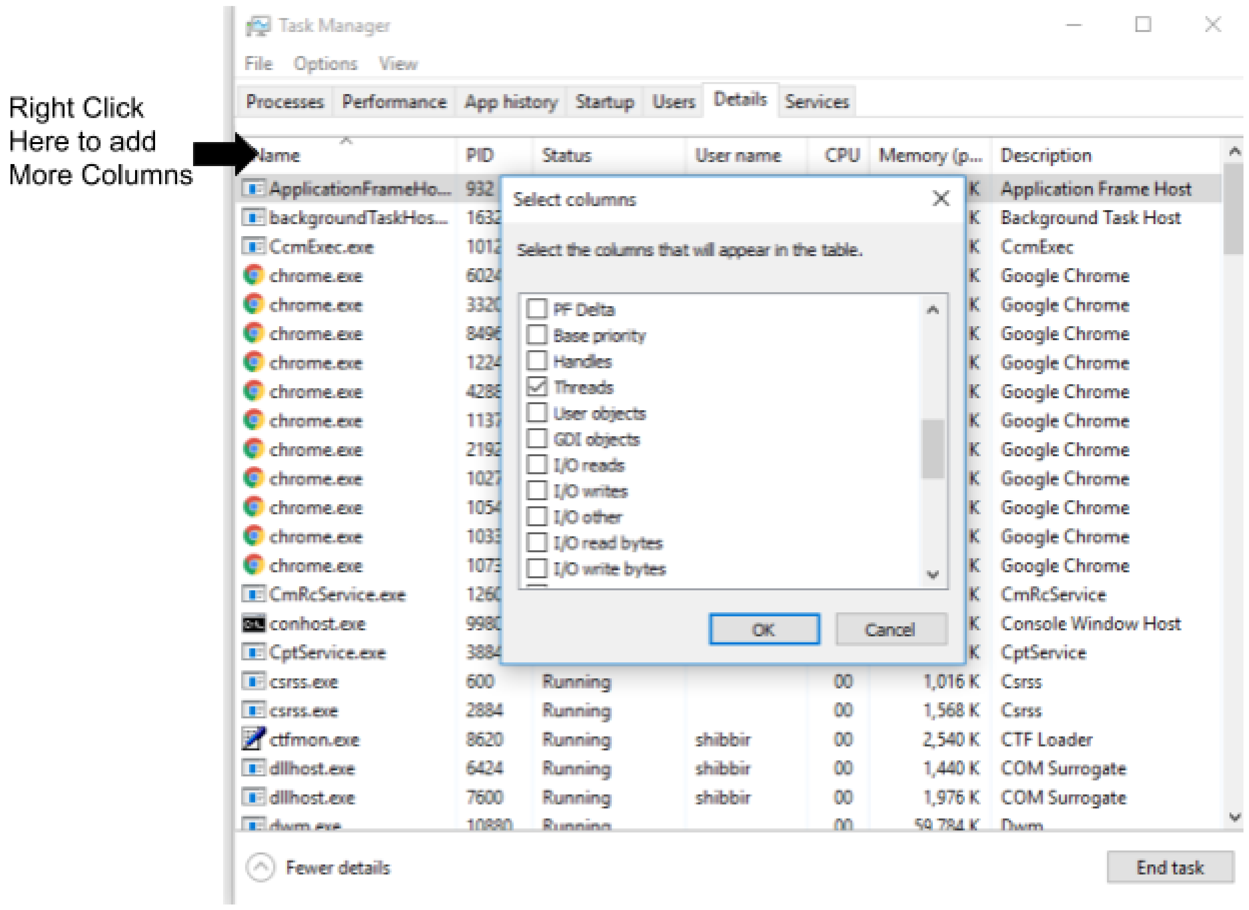
```
rtualMachines/jdk1.8
    1 ThreadExample_1 (1) [Java Application] /Library/Java/Java...0_20.jdk/Contents/Home/bin/java (Aug 25, 2017, 9:40:39 AM)
    2 ThreadExample_1 (1) [Java Application] /Library/Java/Java...0_20.jdk/Contents/Home/bin/java (Aug 25, 2017, 9:40:58 AM)
  ✓ 3 ThreadExample_1 (1) [Java Application] /Library/Java/Java...0_20.jdk/Contents/Home/bin/java (Aug 25, 2017, 9:41:09 AM)
g
g
g
g
g
```

- **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
ionodecache	0.0	3:04.65	3	0	93	root
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,


Right Click Here to add More Columns
Task Manager
File Options View
Processes Performance App history Startup Users Details Services
Fewer details
End task

Select columns

Select the columns that will appear in the table.

- [ ] PF Delta
- [ ] Base priority
- [ ] Handles
- [x] Threads
- [ ] User objects
- [ ] GDI objects
- [ ] I/O reads
- [ ] I/O writes
- [ ] I/O other
- [ ] I/O read bytes
- [ ] I/O write bytes

OK Cancel

Name	PID	Status	User name	CPU	Memory (p...	Description
ApplicationFrameHo...	932				K	Application Frame Host
backgroundTaskHos...	1632				K	Background Task Host
CcmExec.exe	1012				K	CcmExec
chrome.exe	6024				K	Google Chrome
chrome.exe	3320				K	Google Chrome
chrome.exe	8496				K	Google Chrome
chrome.exe	1224				K	Google Chrome
chrome.exe	4288				K	Google Chrome
chrome.exe	1137				K	Google Chrome
chrome.exe	2192				K	Google Chrome
chrome.exe	1027				K	Google Chrome
chrome.exe	1054				K	Google Chrome
chrome.exe	1033				K	Google Chrome
chrome.exe	1073				K	Google Chrome
CmRcService.exe	1260				K	CmRcService
conhost.exe	9980				K	Console Window Host
CptService.exe	3884				K	CptService
csrss.exe	600	Running		00	1,016 K	Csrss
csrss.exe	2884	Running		00	1,568 K	Csrss
ctfmon.exe	8620	Running	shibbir	00	2,540 K	CTF Loader
dllhost.exe	6424	Running	shibbir	00	1,440 K	COM Surrogate
dllhost.exe	7600	Running	shibbir	00	1,976 K	COM Surrogate
dwm.exe	10880	Running		00	59,784 K	Dwm

Task 2: Learn about Data Corruption

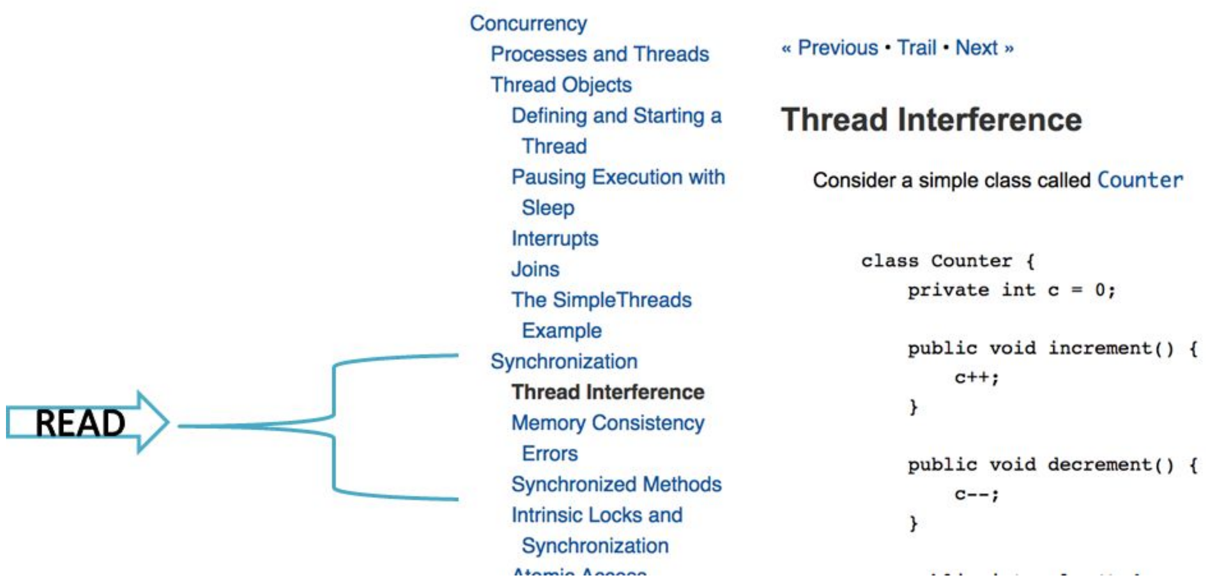
Learning Objectives:

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.



Concurrency
- Processes and Threads
- Thread Objects
  - Defining and Starting a Thread
  - Pausing Execution with Sleep
  - Interrupts
  - Joins
  - The SimpleThreads Example
- Synchronization
  - **Thread Interference**
  - Memory Consistency Errors
  - Synchronized Methods
  - Intrinsic Locks and Synchronization

« Previous • Trail • Next »

Thread Interference

Consider a simple class called `Counter`

```
class Counter {
    private int c = 0;

    public void increment() {
        c++;
    }

    public void decrement() {
        c--;
    }
```

- **Step 1:**
  - Download DataCorruption.java (inside Sample Codes_Activity 01.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

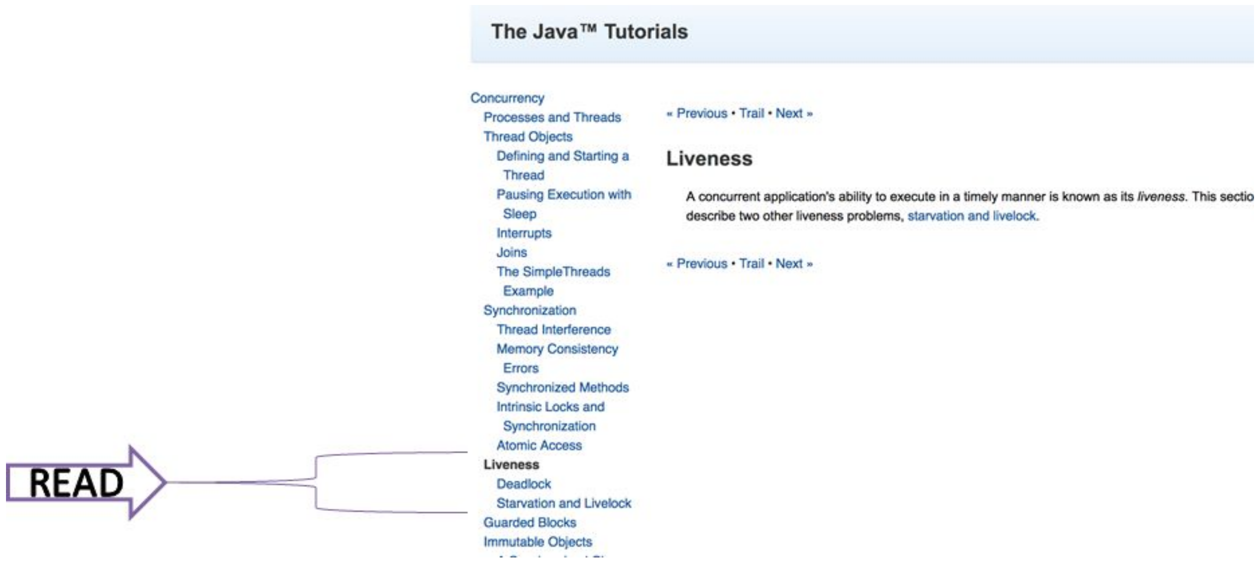
Task 3: Learn about DEADLOCKS

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)



The Java™ Tutorials

Concurrency
- Processes and Threads
- Thread Objects
  - Defining and Starting a Thread
  - Pausing Execution with Sleep
  - Interrupts
  - Joins
  - The SimpleThreads Example
- Synchronization
  - Thread Interference
  - Memory Consistency Errors
  - Synchronized Methods
  - Intrinsic Locks and Synchronization
  - Atomic Access
- Liveness
  - Deadlock
  - Starvation and Livelock
- Guarded Blocks
- Immutable Objects

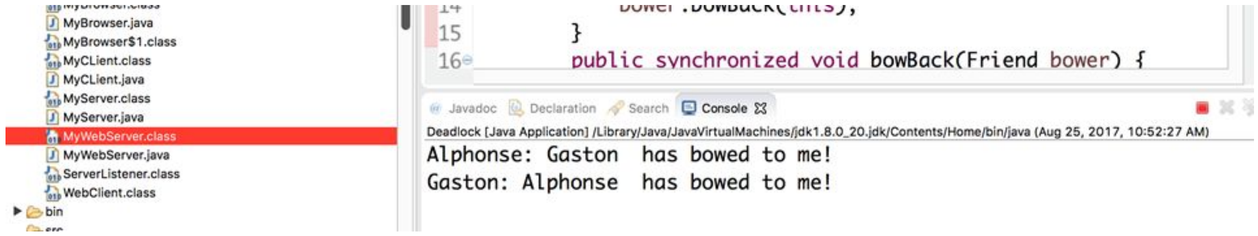
« Previous • Trail • Next »

Liveness

A concurrent application's ability to execute in a timely manner is known as its *liveness*. This sectio describe two other liveness problems, starvation and livelock.

« Previous • Trail • Next »

- **Step 1:**
  - Download Deadlock.java (inside Sample Codes_Activity 01.zip).
  - Read the code and see if you understand it.
  - Run it several times. Does the program stop?



MyBrowser.java
MyBrowser$1.class
MyCLient.class
MyCLient.java
MyServer.class
MyServer.java
MyWebServer.class
MyWebServer.java
ServerListener.class
WebClient.class
bin

```
        }
        public synchronized void bowBack(Friend bower) {
```

Deadlock [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_20.jdk/Contents/Home/bin/java (Aug 25, 2017, 10:52:27 AM)

```
Alphonse: Gaston  has bowed to me!
Gaston: Alphonse  has bowed to me!
```

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

- **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 number 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?