**CSC Exam 2**

**Shahroz Imtiaz**

* **Recursion**
  + **Recursive definition -** something is defined in terms of smaller versions of itself.
    - **Consists/ Verifying of:**
      * **Base Case Question** - is the base case correct for the values
      * **Smaller-Caller Question** - Does the recursive statement progress us closer towards the base case
      * **General Case Question** - Assuming the previous one resulted correctly, is the general correct?
  + **Direct Recursion** - Is when the recursive call is on the same method (A🡪A)
  + **Indirect Recursion -** Is when a method is called and eventually on the method-call stack is another reference back to the same method (A🡪B🡪…🡪A)
  + Given each method call in java gets a new element pushed onto a stack; more and more data will be allocated. And a limit might be reached resulting in an **out of space in run-time stack.**
  + **Memoization -** Is when data is computed as needed, but is also saved for later use.
* **Queues** - is a data structure that is expandable in size.
  + **Modifying**:
    - To modify the queue, you can only access/remove the first element and add new elements to the end. Known as FIFO (first in first out)
      * **enqueue** - adds an element to the back of the queue
        + (cause a problem if bounded)
      * **dequeue** - takes an element off the front of the queue
        + (cause a problem if empty)
  + **Applications:** 
    - **Queues are useful** to organize ordering in order to be served/processed. Used extensively for systems/programs that involve buffers, and **when resources are shared** among consumers **(task scheduling).** 
      * **Examples:**
        + **CPU Core thread assignment/processing**
        + **I/O Buffer**
  + **Device-Status Table -** is a table containing an entry for each device: device’s type, address, and state (functioning/idle/busy).
    - A **wait queue** will maintain a list of waiting requests when the device is busy.
  + **Process –** is informally a program during execution (Active), however, a program (Passive) is more than just program code.
    - **Process State** – is the current activity of the Process.
      * **New -** The process is being created
      * **Running -** Instructions are being executed
      * **Waiting -** The process is waiting for some event to occur
      * **Ready -** The process is waiting to be assigned to a processor
      * **Terminated -** The process has finished execution
    - **Process Scheduler -** Selects an available process for program execution on the CPU.
      * As processes enter the system they will be added to a **Job Queue**, which consists of all processes on the system.
      * The processes that are ready and waiting to execute are kept on the **Ready Queue.**
* **Concurrency**
  + A **thread** of execution is the smallest sequence of programmed instructions that can be managed independently by a scheduler
    - Multiple threads can exist within the same process, executing **concurrently** (one starting before others finish) and share resources such as memory, while different processes do not share these resources.
      * All Java programs comprise at least a single thread of control (**Main thread**)
    - **Thread methods:**
      * **isAlive() -** returns whether or not the Thread is still alive (waiting, sleeping, or otherwise occupied).
      * **sleep(ms)** - pauses the calling thread for ms milliseconds.
        + This could cause an **InterruptedException** - A checked exception that is thrown when a thread is waiting, sleeping, or otherwise occupied, and the thread is interrupted.
    - **Java Thread States –** is the current activity of the thread object.
      * **New** - A thread is in this state when an object for the thread is created
      * **Runnable** - Calling the start method allocates memory for the new thread in the JVM and calls the run() method for the thread object. When the run method is invoked the state changes from new to runnable. NOTE: a running thread is runnable, there is no distinguishing state between able to run and running.
      * **Blocked** - The thread becomes blocked if it performs a blocking statement — like doing I/O or calling a Thread method of sleep()
      * **Dead** - The thread moves to the dead state when the run() method completes
    - **Thread Synchronization**
      * By **synchronizing** the method (adding the **synchronized** **modifier**), you ensure only one synchronized method is run at a time).
        + If you have multiple operations being performed you may not want to synchronize the entire method, but rather prohibit sections of code within a method. The synchronized(...){ ...} block, will grab a lock on the specified object.

It is possible to have two separate threads lock up, Thread A is on line 9 and Thread B is on line 15...neither is giving up their locks so they both sit and wait...forever...in a **Deadlock**.