* Exceptions
  + Building exception objects can be done by extending a class with Exception, to create your own
  + The method signature can contain Throws (exceptionName) if the exception isn’t checked
  + Checked exceptions use a try-catch block
    - Try {

//Code

} catch(Exception e) {

//handle try/catch

}

* Swings and GUIs
  + Lightweight component
    - Anything that isn’t a heavyweight component, buttons, panels, etc
  + Heavyweight component
    - Interacts with the operating system to coordinate mouse position, button clicks, a lot more bulk in handling.
  + MVC pattern
    - Model-View-Controller Pattern
    - Used to separate application’s concerns
      * Model
        + Represents an object carrying data
        + Has logic to update controller if data changes
      * View
        + View represents the visualization of the data that model contains
      * Controller
        + Controller acts on both model and view. It controls the data flow into model object and updates the view whenever data changes. It keeps view and model separate.
  + Concurrency
    - Different thread states
      * New
        + When a new thread is created, it is in the new state. The thread has not yet started to run when thread is in this state. When a thread lies in the new state, it’s code is yet to be run and hasn’t started to execute
      * Runnable
        + A thread that is ready to run is moved to runnable state. In this state, a thread might actually be running or it might be ready run at any instant of time. It is the responsibility of the thread scheduler to give the thread, time to run.  
          A multi-threaded program allocates a fixed amount of time to each individual thread. Each and every thread runs for a short while and then pauses and relinquishes the CPU to another thread, so that other threads can get a chance to run. When this happens, all such threads that are ready to run, waiting for the CPU and the currently running thread lies in runnable state.
      * Blocked/Waiting
        + For example, when a thread is waiting for I/O to complete, it lies in the blocked state. It’s the responsibility of the thread scheduler to reactivate and schedule a blocked/waiting thread. A thread in this state cannot continue its execution any further until it is moved to runnable state. Any thread in these states does not consume any CPU cycle.
        + A thread is in the blocked state when it tries to access a protected section of code that is currently locked by some other thread. When the protected section is unlocked, the schedule picks one of the thread which is blocked for that section and moves it to the runnable state. Whereas, a thread is in the waiting state when it waits for another thread on a condition. When this condition is fulfilled, the scheduler is notified and the waiting thread is moved to runnable state.
        + If a currently running thread is moved to blocked/waiting state, another thread in the runnable state is scheduled by the thread scheduler to run. It is the responsibility of thread scheduler to determine which thread to run.
      * Timed Waiting
        + A thread lies in timed waiting state when it calls a method with a time out parameter
        + For example, when a thread calls sleep or a conditional wait, it is moved to a timed waiting state.
      * Terminated
        + Thread is terminated
      * Operating system alternates threads between ready and running
    - What can go wrong with concurrency
      * Multithreaded programs can have issues modifying the same object at the same time.
      * Using tags like synchronized can prevent threads from executing the same functions at once.
    - Know how threads are produced (and run) in java. What do we use to model a thread?
      * A runnable is used to model a thread
      * The ExacutorService class can be used to produce and run threads.