* Swing is not multithreaded
  + after being set up, the only thread that can safely interact with Swing components is the Swing event thread
    - the Swing event thread is responsible for handling every UI event that occurs
  + this is problematic when a thread that is doing calculations or functions needs to directly change the UI components, but only the Swing thread has access
    - this means you need to force the Swing threads to do the process
    - do this by using SwingUitilities.invokeLater()
      * Static helper method for properly asking Swing event thread to do something
      * Takes a Runnable object(something that implements Runnable) as parameter, and it runs this event when it has time to do so
      * does not create a new thread
        + schedules execution with existing Swing event thread
    - Ex:

class WidgetUpdater implements Runnable {

private Timer widget tw;

private double delta;

public WidgetUpdater{Timer tw, double delta){

this.tw = tw;

this.delta = delta;

}

public void run(){

tw.updateTimer(delta);

}

}

* then call SwingUtilities.invokeLater(new WidgetUpdater(tw, delta));

**Nested Classes**

* Inner Class
  + a class within a class
  + if public, then instances can be made oustide of the outer class
  + if private, then instances can only be made within the outer class
    - this is a good way to create helper classes that you don't want to expose outside of the class
* Local Class
  + class declared within an instance method of the outer class
  + instances of local classes have access to the encapsulated fields of the outer class instance
    - don't have to pass these fields through
  + later in the method, you can create instances of the local class
* Anonymous Class
  + define and create an object instance that implements an interface without having to formally define a class
  + special syntax evaluates as an expression
    - can only **extend ONE interface**
    - resulting instance can not have a constructor or any encapsulated state!
    - does have access to any local variables declared as final as well as any encapsulated state of the current object when used
  + ex:

SwingUtilities.invokeLater(newRunnable() {

public void run() {

tw.updateElapsed(delta);

}

});