MVC reconsidered: Boggle and OOGA

- What are use-cases and scenarios for boggle (client POV)
  - As visible timer ticks, what does user do?

  - How does user interact with the game?
    - What are different possibilities? GUI, Text?
    - What should happen as user "enters" words?

  - What happens when game is over?
    - Solitaire version
    - Multi-player (networked) version

  - Other relevant scenarios, use-cases? (starting, re-starting)...

From scenarios to View to Model

- Interacting with the game, what does view look like?
  - Sketch a simple GUI/view, solitaire and networked?

  - What events does generate/process, how are these handled and who does the handling
    - More use-cases, but lower-level: click here, select menu
    - What happens, e.g., when user enters a word in textfield?

From View to Model (via controller)

- What are responsibilities of Boggle Model?
  - Find given word, what does find mean?
  - Find all words, from what word source?
  - Find missed words, from what list?

- How do GUI and model interact?
  - What generates a list of every word on the board?
  - What shows the user words she/he missed?
  - What shows the user "bad" words (what does bad mean?)
  - What lights up words selected by the user from a list?

From controller to threads

- Threads are lightweight processes (what’s a process?)
  - Threads are part of a single program, share state of the program (memory, resources, etc.)
  - Several threads can run “at the same time”
    - What does this mean?
  - Every Swing/AWT program has at least two threads
    - AWT/event thread
    - Main program thread

- Coordinating threads is complicated
  - Deadlock, starvation/fairness
  - Monitors for lock/single thread access
Concurrent Programming

- Typically must have method for ensuring atomic access to objects
  - If different threads can read and write the same object then there is potential for problems
    - ProducerConsumer.java example
    - Consider getting x and y coordinates of a moving object
  - If an object is read-only, there are no issues in concurrent programming
    - String is immutable in Java, other classes can have instance variables be defined as final, cannot change (like const)
- In Java, the keyword synchronized is the locking mechanism used to ensure atomicity
  - Uses per-object monitor (C.A.R. Hoare), processes wait to get the monitor, it’s re-entrant

Using synchronized methods

- Methods can be synchronized, an object can be the argument of a synchronized block, a class cannot be synchronized
  - Every object has a lock, entering a synchronized method of the object, or using the object in a synchronized block, blocks other threads from using synchronized methods of the object (since the object is locked)
  - If a synchronized method calls another synchronized method on the same object, the lock is maintained (even recursively)
  - Another thread can execute any unsynchronized method of an object O, even if O’s lock is held
  - A thread blocks if it tries to execute a synchronized method of an object O if O’s lock is held by a different thread

Thread classes in Java

- Classes can extend java.lang.Thread or implement java.lang.Runnable, (note: Thread implements Runnable)
  - A thread’s run method is executed when the thread is started
  - Typically the run method is “infinite”
    - Executes until some final/done state is reached
    - The run method must call sleep(..) or yield(); if not the thread is selfish and once running may never stop
  - A runnable object is run by constructing a Thread object from the runnable and starting the thread
- Threads have priorities and groups
  - Higher priority threads execute first
  - Thread groups can be a useful organizational tool

Example: Timer, TimerUser

- What’s the thread in the TimerUser class? The Timer class?
  - How do threads communicate?
  - How do threads stop?
- How does Timer work
  - What is a daemon thread?
  - How do clients find out when they’re ticked?
    - Is there a better approach? Drawbacks?
    - What about signature of Timer.pause()?
- Why is an interface needed to support callbacks?
  - What would we do in C++?