



# CS 341

## Software Design

### Minilab 8

### Mutli-threading

### November 29, 2007

This program simulates a water tank to which water can be added and removed. In the initial version of the program, clicking the Fill button adds a little water, while clicking the Empty button removes a little water. As the water level changes, the display is updated to reflect the current water level. The user could repeatedly click these buttons to fill or empty the tank, but that would be boring!

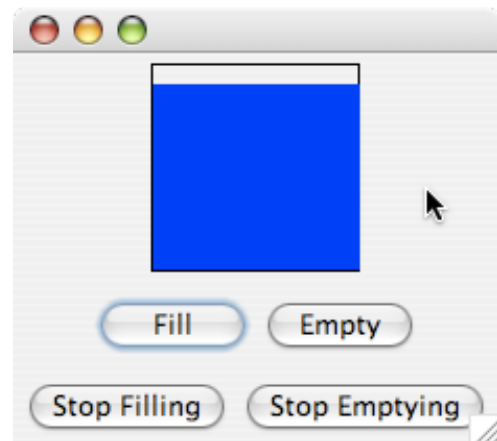
To gain experience with threads, you will change this behavior of the program such that clicking the Fill button will start a thread that slowly fills the tank. Clicking the Empty button will start a thread that slowly empties the tank. The Stop buttons should stop the respective threads. You should be careful that there is at most one Fill thread and at most one Empty thread running at a time. You should also automatically stop the filling thread when the tank is full.

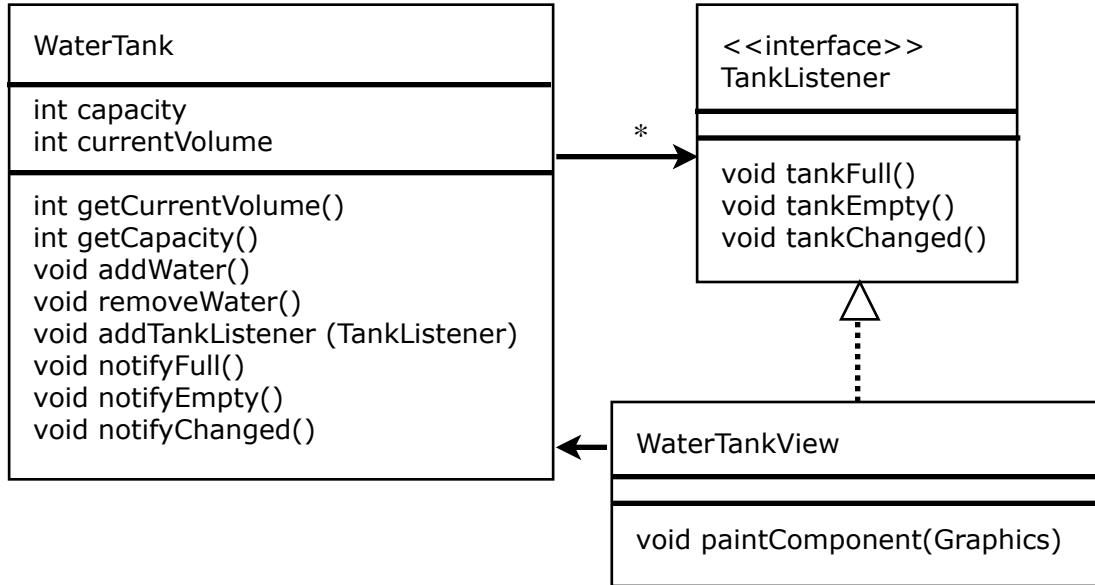
### The WaterTank program

The WaterTank program uses an MVC architecture. WaterTank is the model class. A water tank has a maximum capacity and a current volume of water. When the water level changes, the tank becomes full or the tank becomes empty, the water tank notifies all of its listeners.

The WaterTankView class implements the display of a water tank as a rectangle with the water level within it represented as a solid blue rectangle as shown to the right. The WaterTankView listens to the water tank so that it can redraw the blue rectangle as the water level changes.

The GUI is implemented in the main method of the WaterTank class.





## Methods in the Thread class

`Thread ()`

Creates a new thread. When this thread is started, it will call its `run()` method.

`public void start()`

Starts the thread.

`public static void sleep (int millis) throws InterruptedException`

Causes the currently-executing thread to give up the CPU for at least the number of milliseconds specified. Throws `InterruptedException` if `interrupt()` is called on this thread while it is sleeping.

`public boolean isAlive()`

Returns true if the thread has been started and its run method has not yet completed.

`public void run()`

This is the method that you define to specify the behavior of the thread while it is running. Do not call this method. It is called by `Thread.start()`.

## Working with threads

**Step 0:** Download the project and run it as it currently is. Click the Fill and Empty buttons repeatedly.

**Step 1:** Define the `Emptier` class. It should extend `Thread`. Its `run` method should slowly remove water from the tank. So, it should loop forever, calling `removeWater` from the water tank. The loop should also contain a call to `sleep`. To make it a little more interesting when both the filling and emptying threads are running, write your `Emptier` so that it sleeps a

random amount of time. To compute a random amount of time in the range from 0 to 200 milliseconds, do:

```
(int) (Math.random() * 200)
```

Put the try statement (required by your sleep call) around the entire while loop so that the run method exits when the thread is interrupted.

**Step 2:** Change the action listener associated with the empty button to create and start an emptying thread. Be sure to create only one Emptier. Test it. Save the emptying thread in an instance variable in the WaterTank class so that you will be able to interrupt it in the next step.

**Step 3:** Implement the action listener for the Stop Emptying button to interrupt the emptying thread. Test it.

**Step 4:** If you now click empty, then stop emptying, then empty again, you will likely discover that it doesn't start emptying again. This is because you have an Emptier object but the thread it is running in has stopped. Use Thread's `isAlive()` method to determine if the emptier thread has stopped. If so, create a new Emptier and start it. (Don't try to restart the existing thread; that won't work.)

**Step 5:** Now, do the same thing for the Filler thread. Initially don't worry about the tank overflowing. Define the Filler class. Change the action listener to create a filler if one is not already running. Test it. Implement the action listener to stop the filler thread. After testing the Fill button alone, try clicking both the fill and empty buttons. The water should move up and down as the two threads compete to control the water level.

**Step 6:** To stop the filling thread when the tank is full, make the Filler listen to the WaterTank. When it is notified that the tank is full, interrupt the filler.