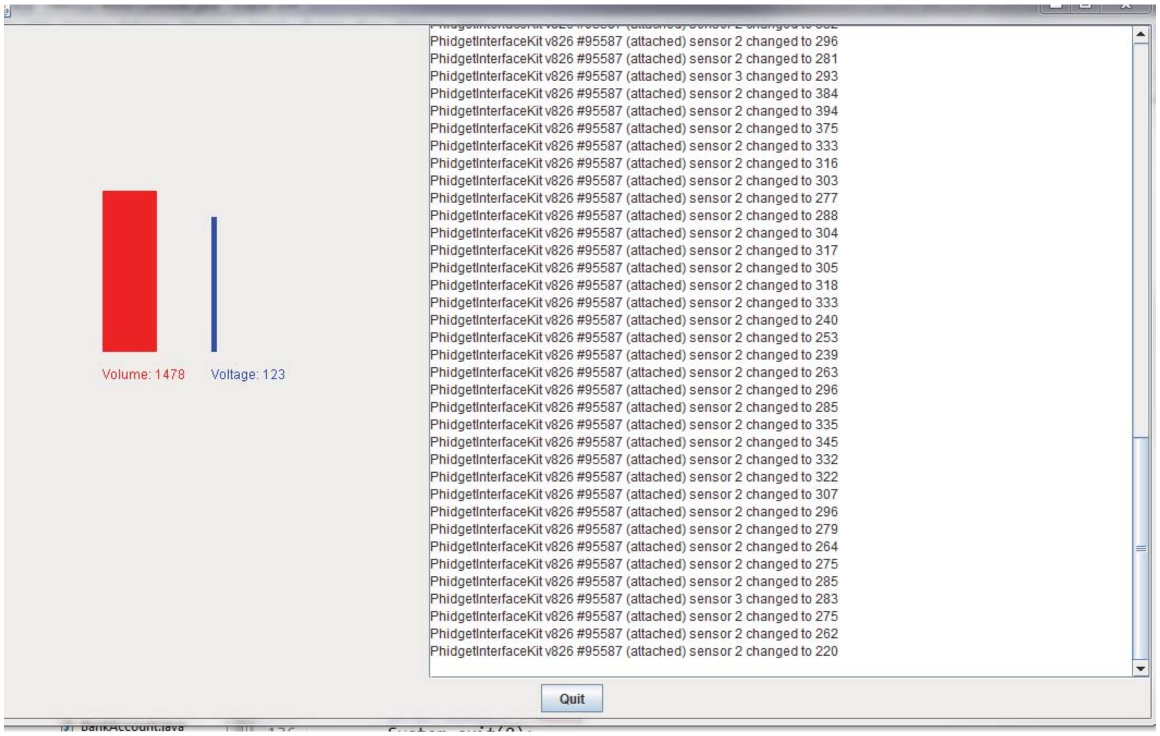
Introduction to Computers and Engineering Problem Solving
Spring 2012
Problem Set 9: Dam management: threads and sensors
Due: 12 noon, Friday, May 4, 2012

1. Problem Statement

You are the operator of a dam that provides water for domestic, industrial and agricultural use, and also generates power:

- The dam has random inflows (rainstorms, snow melt, streams) and random demands for water from its users. You will handle those in two separate threads that generate additions and subtractions from the water reservoir's volume.
- The operator has two controls:
  - A rotation sensor controls the gate at the spillway. If the water levels are dropping, the spillway can be closed to keep the level (volume) near the desired point. If water levels are increasing, the spillway can be opened.
  - A slider controls the speed of the power turbines. The voltage generated from the turbine-generator sets varies based on the water volume and on the slider setting.
- The system status is displayed on a Swing GUI, which has three components:
  - A JTextArea and JScrollPane, which you should base on the JTextViewer from lecture 29, that displays the sensor events
  - A pair of rectangles that show the current volume and voltage, on a JPanel
  - A quit button



```
PhidgetInterfaceKit v826 #95587 (attached) sensor 2 changed to 296
PhidgetInterfaceKit v826 #95587 (attached) sensor 2 changed to 281
PhidgetInterfaceKit v826 #95587 (attached) sensor 3 changed to 293
PhidgetInterfaceKit v826 #95587 (attached) sensor 2 changed to 384
PhidgetInterfaceKit v826 #95587 (attached) sensor 2 changed to 394
PhidgetInterfaceKit v826 #95587 (attached) sensor 2 changed to 375
PhidgetInterfaceKit v826 #95587 (attached) sensor 2 changed to 333
PhidgetInterfaceKit v826 #95587 (attached) sensor 2 changed to 316
PhidgetInterfaceKit v826 #95587 (attached) sensor 2 changed to 303
PhidgetInterfaceKit v826 #95587 (attached) sensor 2 changed to 277
PhidgetInterfaceKit v826 #95587 (attached) sensor 2 changed to 288
PhidgetInterfaceKit v826 #95587 (attached) sensor 2 changed to 304
PhidgetInterfaceKit v826 #95587 (attached) sensor 2 changed to 317
PhidgetInterfaceKit v826 #95587 (attached) sensor 2 changed to 305
PhidgetInterfaceKit v826 #95587 (attached) sensor 2 changed to 318
PhidgetInterfaceKit v826 #95587 (attached) sensor 2 changed to 333
PhidgetInterfaceKit v826 #95587 (attached) sensor 2 changed to 240
PhidgetInterfaceKit v826 #95587 (attached) sensor 2 changed to 253
PhidgetInterfaceKit v826 #95587 (attached) sensor 2 changed to 239
PhidgetInterfaceKit v826 #95587 (attached) sensor 2 changed to 263
PhidgetInterfaceKit v826 #95587 (attached) sensor 2 changed to 296
PhidgetInterfaceKit v826 #95587 (attached) sensor 2 changed to 285
PhidgetInterfaceKit v826 #95587 (attached) sensor 2 changed to 335
PhidgetInterfaceKit v826 #95587 (attached) sensor 2 changed to 345
PhidgetInterfaceKit v826 #95587 (attached) sensor 2 changed to 332
PhidgetInterfaceKit v826 #95587 (attached) sensor 2 changed to 322
PhidgetInterfaceKit v826 #95587 (attached) sensor 2 changed to 307
PhidgetInterfaceKit v826 #95587 (attached) sensor 2 changed to 296
PhidgetInterfaceKit v826 #95587 (attached) sensor 2 changed to 279
PhidgetInterfaceKit v826 #95587 (attached) sensor 2 changed to 264
PhidgetInterfaceKit v826 #95587 (attached) sensor 2 changed to 275
PhidgetInterfaceKit v826 #95587 (attached) sensor 2 changed to 285
PhidgetInterfaceKit v826 #95587 (attached) sensor 3 changed to 283
PhidgetInterfaceKit v826 #95587 (attached) sensor 2 changed to 275
PhidgetInterfaceKit v826 #95587 (attached) sensor 2 changed to 262
PhidgetInterfaceKit v826 #95587 (attached) sensor 2 changed to 220
```

The figure shows an example of the GUI. The operator is to keep the water volume at 1200 and the voltage at 110, as much as possible. (Both of these values are in 1000s: the reservoir holds 1,200,000 acre-feet and the powerhouse generates at 110,000 volts.) LEDs must be lit when the rotation or slider sensors are near their maximum values, as warnings (see below for details).

2. Program

For the water flows in and out of the reservoir, use the BankAccount and ATM classes from lecture 29 as approximate models:

- Write a Reservoir class, analogous to BankAccount, that holds the volume of water in the reservoir.
  - It needs data members, a constructor, get and set methods, as appropriate. Package access is appropriate. Use ints for volume and voltage.
    - Ints are 4 bytes, for which read and write operations are atomic in Java; doubles would require additional synchronization for thread safety. Minimize the use of the synchronized keyword.
  - It should have a method to add and withdraw water (change volume).
    - This method must also update the voltage, since voltage is mostly determined by volume; use the voltage equation below.
    - This method will be called by the threads that add and withdraw water.
  - It should have a method for updating the volume and voltage given sensor readings of slider and rotational sensor using the equations below.
    - This method will be called by the controller when sensor events occur.
  - The suggested volume change function for the spillway is

    ```
    volume += (rotation - 500)/50;
    ```

    where rotation is the value of the rotation sensor, although you can use any scaling you wish.
  - The suggested voltage change function (after volume is updated) is:

    ```
    voltage = MIN_VOLTAGE + (MAX_VOLTAGE-
        MIN_VOLTAGE)*volume/MAX_VOLUME;
    voltage += (slider - 500)/200;
    ```

    though you can use any scaling you wish.
  - While updating voltage and volume, the following constraints are imposed (define constants for these):
    - The maximum water volume is 2000, and the minimum is zero

      The minimum voltage is 95 volts, except that it's zero if the reservoir volume is less than MAXIMUM_VOLUME/4: there is not enough water to run the turbines in this case. The maximum voltage is 125 volts.
  - Be very careful about synchronization. Recall that the ATM and Bank classes had a synchronization problem.

- Write a model class that is analogous to ATM to handle the random additions to and withdrawals of water from the reservoir.

    - This class should extend Thread. This is easier than implementing Runnable for this homework.
    - It needs appropriate data members, constructor, and a run() method
    - The run() method should use the nextInt() method on a Random object, to generate random inflows of 0-100, and random outflows of 0 to 75, in an infinite loop. Use nextInt(100) and -nextInt(75).
    - The loop should sleep for a second between each addition and withdrawal. Use `Thread.sleep(1000);`

- Write a view class that displays two rectangles, and the volume and voltage values, as shown in the figure above. The view must ask the Reservoir object for the current volume and voltage.

- Write a controller class. Base it approximately on PressureController4 from lecture 29.
    - It will need references to the view, reservoir, and two model classes. The reservoir has initial volume of 1200 and initial voltage of 110.
    - The controller will start two threads (models). Both will generate additions and withdrawals from the reservoir. Think of these as streams of data coming from a set of remote sensors at different places on the dam and reservoir
    - The rotation sensor must be on port 2 and the slider sensor must be on port 4. (We will take off points if they are on the wrong ports)
    - The LED for the rotation sensor status must be on digital out port 1 and the slider on digital out port 2. (We will take off points if they are on the wrong ports)
    - It is not necessary to use a timer in this or any class in this homework. You can handle events as they occur. (However, the extra credit requires a timer)
    - Construct the Swing components needed. Create and add the text area and scroll pane to show the sensor events, based on the JEventViewer example. Add the view, panel and a quit button.
    - The openInterfaceKit() and closeInterfaceKit() methods are pretty standard. In the sensorChanged() method:
        - get the sensor values as usual,
        - update the JTextArea,
        - turn on the first LED only if the rotation sensor is over 950,
        - turn on the second LED only if the slider sensor is over 950
    - The JTextArea will need to be updated from the Phidgets sensor change event. You will need to use `SwingUtilities.invokeLater`
    - Make sure the LEDs are turned off at the start and end of the program; see the LED examples from lecture.

We don't provide sample output other than the GUI figure above, because everyone's output will be different.

3. Extra Credit

In addition to your solution above, you may implement additional functionality for this problem set and get up to 40 extra credit points. **You must first complete and submit the entire standard solution as outlined above.** Do not attempt to develop the extra credit solution until you have completed the normal assignment since it will be graded separately. When you submit your solution to the 1.00 Web site, first submit your original solution. Then upload your extra credit solution as a second .zip file. Both versions should contain all the files needed to compile and run your solution.

You can get extra credit on only one homework from problem sets 8 to 10. If you don't have time do to the extra credit this time, you still have the opportunity to do so in the future.

For extra credit, show graphs of values of the rotation and slider sensors over time, similar to PressureAvgController from lecture 26 (you must use a Timer to do this). Your graphs must show the last 50 sensor readings (sensor values are sampled at 100ms time intervals). Use awt.geom.Path2D to line-link adjacent value pairs while painting.

You must also compute averages of every consecutive 5 values from the last 50 sensor readings (So instead 50 sensor readings, this gives 10 averages). Again, use awt.geom.Path2D to line-link adjacent average value pairs while painting.

Turn In

1. Place a comment with your and your partner's full names, sections, TA names and assignment number at the beginning of all `.java` files in your solution. In this homework, you will have multiple `.java` files.
2. Place all of the files in your solution in a single zip file.
    a. Do not turn in electronic or printed copies of compiled byte code (`.class` files) or backup source code (`.java~` files)
    b. Do not turn in printed copies of your solution.
3. Submit this single zip file on the 1.00 Web site under the appropriate section and problem set number. For directions see **How To: Submit Homework** on the 1.00 Web site.
4. Your solution is due at noon. Your uploaded files should have a timestamp of no later than noon on the due date.
5. After you submit your solution, please recheck that you submitted your .java file. If you submitted your .class file, you will receive **zero credit.**

Penalties

- 30 points off if you turn in your problem set after Friday noon but before noon on the following Monday. You have one no-penalty late submission per term for a turn-in after Friday noon and before Monday noon.
- No credit if you turn in your problem set after noon on the following Monday.