# A Weird Thread Exercise

**This exercise is not being collected or graded.** You probably won't have time to finish this during class, so please continue to work on it at home. It's a good exercise for practicing Java synchronization.

1. First, you will create a "MutableInteger" class. It will simply wrap an int value with two methods: getValue() and setValue(). Since we will only use ONE MutableInteger for the whole project, use the Singleton design pattern! That means make a private constructor, and create a static final instance of the class for everyone to use. **IMPORTANT: initialize the value to -1 to start off.**

2. Now write a class called LastDigitThread that extends Thread, and has a main() method inside. The main method will create and start 10 threads (instances of the class). Each thread will be associated with a digit (0 through 9), so there should be a constructor that accepts an int.

**OVERVIEW**

main() will continually try to store random positive integers in the MutableInteger object. The threads will all try to read a value from this object in order to "process" it (as described below). Each value stored by main will be read and processed by just ONE thread. When the thread reads the value, it will store -1 in the MutableInteger object to let main know that it is time to store another random number there.

**DETAILS**

main() will go into a loop, waiting for the value of MutableInteger to become -1. When it does, main will store a POSITIVE random integer in the MutableInteger object.

Each thread will go into a loop, waiting for the value of MutableInteger to become something OTHER THAN -1. When it does, the thread will read the value and set MutableInteger to -1, and then "process" the value (as described below).

**The "processing" done by the Threads:**

After the thread has obtained the "value", it will proceed as follows. Remember that each thread is associated with a "digit" (0-9). The thread will (very stupidly) count how many numbers less than the value end with the digit. Don't be clever and solve this some efficient way, just write a loop that cycles from 0 up to the value and asks if (i%10 == digit) -- we want it to take some time.

**HINTS:**

- You should synchronize the transactions on the MutableInteger object.
- You should use wait and notifyAll to make everything work together smoothly without any "busy waits".
- Don't hold a lock during the slow computation!

## [Sample Output on back page…]

MAKE YOUR OUTPUT LOOK SIMILAR TO THIS EXAMPLE:

```
Storing 1074095
Storing 1791997
Storing 30561612
Storing 15216320
107410 numbers less than 1074095 end with 0
Storing 80563723
Storing 43286275
Storing 52292079
Storing 70394486
179200 numbers less than 1791997 end with 1
Storing 9445617
Storing 32555795
Storing 15451058
Storing 23579693
Storing 35114408
944561 numbers less than 9445617 end with 7
Storing 56000115
1521632 numbers less than 15216320 end with 0
Storing 71642166
1545105 numbers less than 15451058 end with 9
Storing 79274520
2357970 numbers less than 23579693 end with 1
Storing 30080515
3056161 numbers less than 30561612 end with 2
Storing 28667008
3255579 numbers less than 32555795 end with 8
Storing 94755537
4328628 numbers less than 43286275 end with 4
Storing 34703070
3511441 numbers less than 35114408 end with 7
Storing 56586784
5229208 numbers less than 52292079 end with 5
Storing 13751426
3008052 numbers less than 30080515 end with 2
Storing 72404328
...
```