CS431 — Exercise 3

January 25, 2017

Due: Monday, January 30, 2017 before midnight (40 points)

In this exercise you will expand on the concepts from the previous exercise by implementing a blocking queue yourself instead of relying on one provided by the standard library.

Semaphore-based Blocking Queue

Create a class that matches the following header:

```
public final class BlockingQueue<T> {
    ...
}
```

In this class, you should use three semaphores as shown in the producer-consumer problem code: `mutex`, `empty`, and `full` to implement a blocking queue. You can either implement a queue yourself or use a non-blocking queue as an internal data structure for implementing your blocking queue. Your queue should have the following two public methods:

1. `public T dequeue()`
2. `public void enqueue(T t)`

Ideally, you can test your blocking queue by placing it directly in your code for exercise 2. For testing in this exercise, include the following main method in your class:

```
public static void main(String[] args) throws Exception {
    BlockingQueue<Integer> queue = new BlockingQueue<>(100);
    Runnable r = () -> { // replace lambda if you don’t have access to Java 8
        for (int i = 0; i < 200; i++) {
            try {
                int n = queue.dequeue();
                System.out.println(n + " removed");
                Thread.sleep(500);
            } catch (Exception e) {}
        }
    };
    Thread t = new Thread(r);
    t.start();
    for (int i = 0; i < 200; i++) {
        System.out.println("Adding " + i);
        queue.enqueue(i);
    }
}
```

With this test, you should notice that 100 items are immediately enqueued but then the producer must sleep because the queue is full. The consumer will dequeue values at a slow pace (every half-second) as the producer adds more back for the next 100 items.

Submission

1. On https://codebank.xyz, create a project named `CS431-EX3`. Follow this naming convention precisely including case.
2. Create your own local repository by the following:
   (a) Navigate to your local directory for this project.
   (b) Run `git init` to initialize the repository.
3. Make sure you add the reference to the remote repository in your local repository with:
   ```
   $ git remote add origin https://codebank.xyz/username/CS431-EX3.git
   ```
4. Your project should have a main class named `BlockingQueue` in a file named `BlockingQueue.java`. You can have other files or classes, but it should successfully compile and run by simplying using:
   ```
   $ javac BlockingQueue.java
   $ java BlockingQueue
   ```

You will lose points if your submission is not correct (e.g., incorrect repository name, file names, class names, or package declaration that causes the above commands to fail to run).