To summarize this homework assessment, I worked on it for 3 days, pretty much dedicating most of each day to the homework.

Day 1:

The first day of working on this assignment consisted of getting familiar with the code. Some of the key things to get back on pace were how the code shared references/objects. Something to get a refreshment on is the java Runnable interface and the Thread constructor, which lets us run multiple threads, and hadn’t used since high school.

With now a decent idea of what the code structure looked like, I proceeded to move to step 1 to make the auxiliary threads to end at day 370. I accomplished this by creating a getter method in the main class, that returned the current day to the caller.



Text

Description automatically generated Note: these were later changed to be non static.

This caller would then be able to end the while loop that kept the thread alive when calling this function. The most important thing to note is that we wouldn’t have any sort of error/corruption in storing our day since one was exclusively writing and the rest only read the value.



With that out of the way, I proceeded to step 2, where I had to guide the flow of states for the elf and Santa. To start simple, we were advised to tell any elf to go to Santa as soon as it encountered a problem.

Text

Description automatically generated

This could happen for multiple threads at once, so we had a data structure to keep track of the elves at the door.





We lastly had Santa help the elves out when he was awake and then going back to sleep. For this we simply took our queue of elves at the door and helped them one by one until there was none left.

Text

Description automatically generated

These data structures are not thread safe, this meant that there would be scenarios that 2 threads may want to add themselves to the door, but only one succeeded, while the other recorded a false positive.

Day 2:

Step 3 got a little more advanced, we now had a lower limit of elves that could go to Santa’s door. This meant that we needed to keep track of the batch of elves that were in trouble but couldn’t go to the door yet, and the batch that was at the door getting help from Santa. With that we represented our batches as two sets of integers.

Text

Description automatically generated

Note: These sets are initialized to a HashSet object

The reason for changing to Hash Sets was that I kept encountering exceptions with Queues and how I removed items from it. With HashSet, I was able to more easily implement reading the contents of the data structure to help all the elves, and then throw it away. Looking back, I could have definitely used either as long as I read only first and then wiped the data structure.

Text

Description automatically generated

Once I had processed everyone in Santa’s door, I was able to just replace that data structure with the other elves waiting. For Santa, he just had to process the filled data structure and clear it when he was done.

Graphical user interface, text

Description automatically generated

With that out of the way, I continued to step 4. This step took the longest to complete. I originally tried to go off using the existing data structures to keep track of what elf went where, and using a single permit Semaphore, which basically acted as a mutex, to protect the code in its critical zones.



Text

Description automatically generated

This approach was:

1. Not the ideal way to use a Semaphore.
2. Used a data structure when we were already using lists to keep track of our elves.

I realized that I needed to create a better version of step 4 that used semaphores more efficiently.

Instead of spinning the threads, I now was going to do the following:

* There are 3 semaphore permits.



* Whenever there was an elf in trouble, it would try to get a semaphore permit.

Graphical user interface, text

Description automatically generated

* When there were no more permits, it is time to go to Santa’s door.

Text

Description automatically generated

* Santa would help and then release all the permits.

Text

Description automatically generated

* To keep track of which elves held the semaphores, we used a Boolean non-static field.



This way of using the semaphores was able to guarantee us that exactly 3 elves could go to Santa at once. If there were any additional elves in trouble, they would wait for the permit to proceed.

Day 3:

With everything that was required now finished, I thought I might give the bonus step a try. With that, I reintegrated the reindeer into the main class and started working on the states of both the reindeers and Santa.

How it works is, we have another semaphore that allows reindeer to go to the sleigh. This semaphore starts out with having no permits. This way any reindeer that are in the warming shed cannot go to the sleigh unless permits are given.



Text

Description automatically generated

However, once we have all the reindeer that arrived from the beach, we are able to tell Santa to wake up. Santa then wakes up and gives permits to the reindeer so that they are able to transition to the at sleigh state.

Text

Description automatically generated Text

Description automatically generated

At which point, Santa is ready for Christmas!