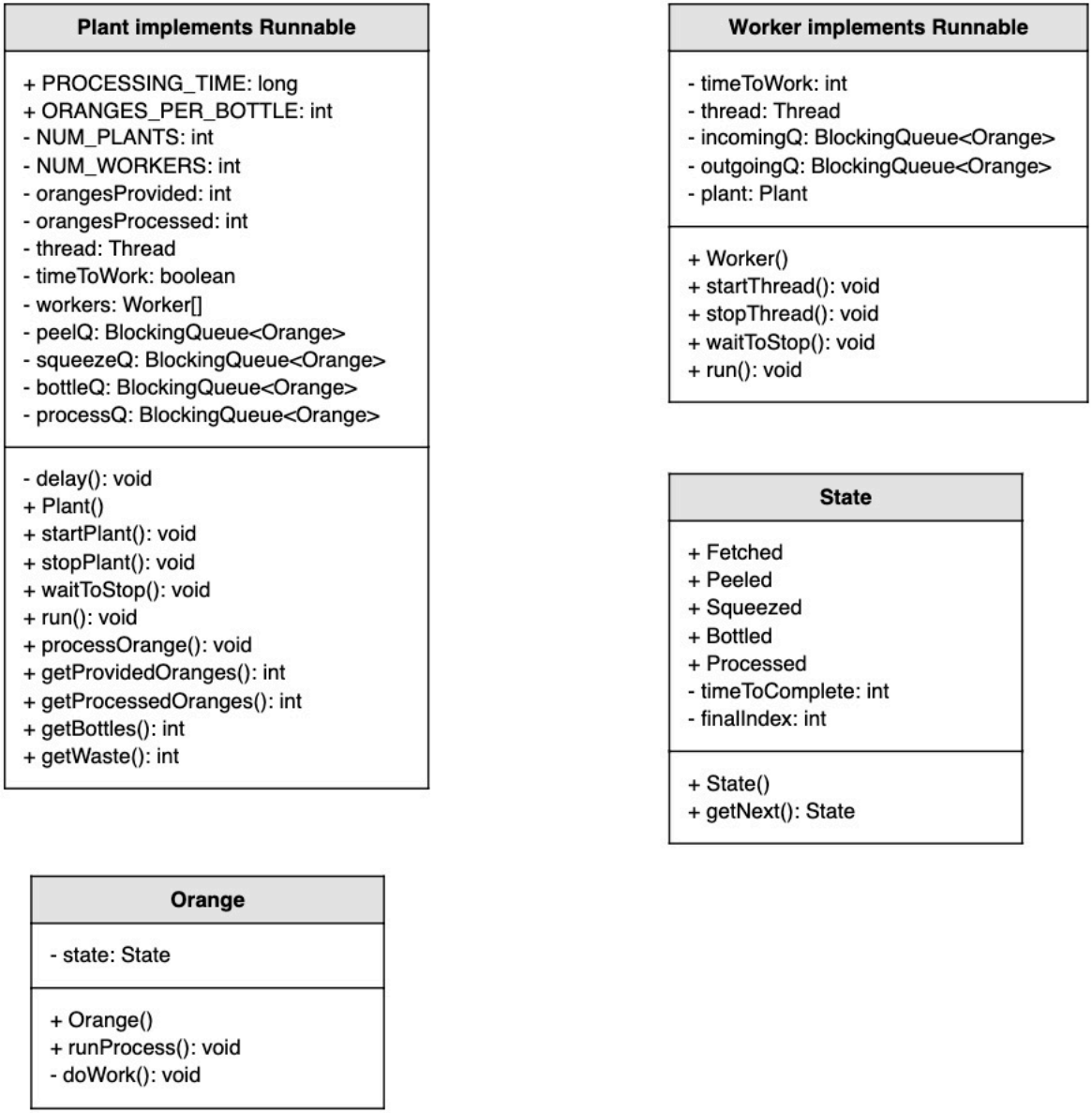
Plant implements Runnable
+ PROCESSING_TIME: long
+ ORANGES_PER_BOTTLE: int
- NUM_PLANTS: int
- NUM_WORKERS: int
- orangesProvided: int
- orangesProcessed: int
- thread: Thread
- timeToWork: boolean
- workers: Worker[]
- peelQ: BlockingQueue<Orange>
- squeezeQ: BlockingQueue<Orange>
- bottleQ: BlockingQueue<Orange>
- processQ: BlockingQueue<Orange>
- delay(): void
+ Plant()
+ startPlant(): void
+ stopPlant(): void
+ waitToStop(): void
+ run(): void
+ processOrange(): void
+ getProvidedOranges(): int
+ getProcessedOranges(): int
+ getBottles(): int
+ getWaste(): int
Worker implements Runnable
- timeToWork: int
- thread: Thread
- incomingQ: BlockingQueue<Orange>
- outgoingQ: BlockingQueue<Orange>
- plant: Plant
+ Worker()
+ startThread(): void
+ stopThread(): void
+ waitToStop(): void
+ run(): void
State
+ Fetched
+ Peeled
+ Squeezed
+ Bottled
+ Processed
- timeToComplete: int
- finalIndex: int
+ State()
+ getNext(): State
Orange
- state: State
+ Orange()
+ runProcess(): void
- doWork(): void

Brief Description / Meeting Requirements

The goal of the lab was to run multiple threads at the same time, at least 5 to be exact. This was accomplished by spawning three plants, one thread each, and within each of those plants we spawned four workers. That gives us a total of 15 threads running each time the program executes. The data structure used was a LinkedBlockingQueue. The entire program was pushed to GitHub and built/ran with Ant.

Challenges Faced

I faced a few challenges during this lab. At first I was trying to run the worker‘s jobs on plant thread which defeats the purpose of workers and having to manage separate threads for them. I find it hard to pick up code that others have written, especially when it‘s on something I‘ve never worked with before, and fully understand it. Therefore, I started by trying to make methods for fetching, peeling, etc. when that was unnecessary. I believe I understood the general concept of the problem but implementing it was difficult.