

<div><div></div></div> Plant
<div><div>- peeler: Worker</div><div>- juicer: Worker</div><div>- bottler: Worker</div><div>- orangesProvided: int</div><div>- orangesProcessed: int</div><div>- startPlant(): void</div><div>- stopPlant(): void</div><div>- waitToStop(): void</div><div>- run(): void</div><div>- getProvidedOranges(): int</div><div>- getProcessedOranges(): int</div><div>- getBottles(): int</div><div>- getWaste(): int</div></div>

<div><div></div></div> Worker
<div><div>- startWorker(): void</div><div>- stopWorker(): void</div><div>- waitToStop(): void</div><div>- run(): void</div><div>- get(Orange o, Queue w): void</div></div>

<div><div></div></div> Orange
<div><div>- state: State</div><div>- getState(): State</div><div>- runProcess(): void</div><div>- doWork(): void</div><div>- Orange():</div></div>

**Brief Description:**

- The Plant classes are very similar to those given in class, but now include instances of the Worker class (3 Workers per Plant)
- The Plant instances each contain 4 LinkedBlockingQueues, one for each State of the Orange class
- Each Worker has 2 LinkedBlockingQueues, one for the Oranges that need to be processed and another for the Oranges that have been processed
- Each Worker extends the Runnable Object type, so when we have multiple Plants running, we have at least 8 threads running simultaneously
- Each Worker handles a different task, only running one process of each Orange (each Plant gets a peeler, juicer, and bottler for our task parallelization)
- At the end of the simulation, the statistics show that we can have more than 3 wasted oranges, meaning that juice from one Plant is not being used in another (showing our data parallelization)