Project Requirements (and how they were met):

Must have AT LEAST two Plants running (data parallelization):

The requirement of having at least two Plants running is met by defining NUM_PLANTS as 2 and creating two Plant threads in the main method. Each plant independently produces oranges, which demonstrates data parallelization because production is distributed across multiple producer threads running concurrently.

Must have multiple Workers per plant operating on the Oranges (task parallelization):

The requirement for multiple Workers per plant is satisfied by defining NUM_WORKERS as 2 and creating two Worker threads for each plant. These workers retrieve oranges from the plant’s shared BlockingMailbox and process them simultaneously, demonstrating task parallelization since multiple threads perform processing work on oranges at the same time.

Challenges Faced:

Synchronization:

One challenge I faced during this project was implementing correct synchronization. Plants produce oranges while workers consume them, so threads must coordinate access to the shared mailbox. Without proper synchronization using the wait(), notifyAll(), and synchronized methods, the program could experience race conditions or deadlocks which results in lost oranges.

Shared Data:

Another challenge I faced is protecting shared data. Multiple worker threads update the plant’s processed counter, which creates a risk of race conditions. This requires identifying critical sections and using synchronization to ensure thread updates are safe and have accurate statistics.

Thread Lifecycle:

The last major challenge I faced is managing the thread lifecycle. The system must stop producing oranges while allowing workers to finish processing remaining items in the mailbox. Designing this with a volatile flag and proper loop conditions was difficult, but worked to prevent early thread termination or lost work.