Bullet points:

(Frame of our system) Warehouse and event

(User input files) Simulator and EventProcessor, out of system boundary

(Details of methods in system) OrderEvent, PickerEvent, readyEvent, DefaultEvent

(Details of methods in system) SequencerEvent, LoaderEvent, replenisherEvent

Process steps:

1. Simulator (read in <u>set up files</u>, <u>orders files</u> to warehouse and pick face of floor)
2. Method called process will analyze one command from myorders.txt
3. Then eventProcessor will assign the command to the corresponding event class to process.
4. And the specific event class will use the corresponding methods in MySystem class to complete the action.
5. Then the actions that the events call may lead to the change of instant variables which keep track of data in MySystem and warehouse.
6. The produced log will provide the trace of the process.

Most important classes: MySystem, Floor, EventProcessor, Simulator

How is the event file processed and what happens after reading an event?: EventProcessor has method called process() which will read lines one by one from the event file. And the line will trigger corresponding events listed in event package, every event will use corresponding methods in mySystem so that the event occurs.

What data are you keeping track of and how to keep track of it?:

<u>bufferOrders</u> keeps track of all the orders that are created which contains at most 4 orders. If one order came in, then this order will be stored in the bufferOrders. Once there are four orders, a request will be formed and this request will be stored in the requesList and bufferOrder will be cleaned up after that.

<u>requestList</u> keeps track of all confirmed but not sequenced requests. When sequenced the first request, it will be removed from the requestList.

<u>replenishList</u> keeps track of all fascias that need to be replenished. Every time a picker picks a fascia, system will check if the amount of the fascias is 5 or less, if so, this fascia will be stored in replenishList and waiting for being refilled by a ready replenisher.

<u>savedList</u> keeps track of all orders that had been sequenced.

<u>scannedSkuList</u> keeps track of all the sku numbers that had been scanned be sequencers.

<u>picked</u> is the priority of fascias for picker to pick corresponding fascia, for example, “1->A 0 0 0” then that means picker should pick fascia at “A 0 0 0” first.

replenisherList keeps track of all the replenisher who is ready to refill some specific fascia.

loadedList keeps track of all the fascias that had been loaded to the truck.

What design patterns did you use, what problems do each of them solve:

Factory and **command** design pattern. The factory design pattern (createEvent) dynamically creates new events every time reads in a line from an event file. The command design pattern(Interface: Event) processes the related events for every specific input command.

Unit coverage: 96.7%

Extensibility: If we have new events that need to be processed in our system. Instead of completely deleting codes, we can just add new event classes and corresponding method in mySystem. That is the reason why we use factory design pattern to implement EventProcessor.