**Milestone #3 – Task and LineManager Classes**

This milestone will implement and test the **Task** and **LineManager** class. The **Task** class is a simulation of a station on the assembly line. Each **Task** is responsible for a particular **Item** through inheritance. The **LineManager** class is responsible for the execution and movement of **CustomerOrders** along the assembly line (from start to finish). The definitions of the classes are defined below.

**Starting Point**

Re-use the same solution space your milestone #2 is in, but make the following modifications to the AssemblyLine.cpp file:

* Uncomment lines 18,19,20, and 21
* Comment out line 36 – RunM2(os);
* Uncomment line 37 – RunM3(os);
* Uncomment the RunM3 function starting at line 134

**Task Class**

A **Task** object manages a single **Item** on the assembly line. **Task** inherits a single **Item** and contains the following additional information:

* **Orders** – is a double ended queue with new CustomerOrders coming in one side and exiting out the other once filled.
* **pNextTask** – A pointer to the next task on the assembly line

**Constructor**

Upon instantiation, a **Task** object receives a reference to an unmodifiable **std::string**.  This string contains a single record (one line) that has been retrieved from the *inventory.txt* file specified by the user to be used for **Item** instantiation. The constructor will also set the **pNextTask** to a safe state.

**Copy/Move Capabilities**

A **Task** object represents a single location on the assembly line for filling **Items** into **CustomerOrders**. Therefore, a **Task** object cannot be copied or moved. You must make sure this capability has been deleted from your **Task** definition.

**Member Functions**

* **void RunProcess(std::ostream&)** – runs a single cycle of the assembly line for the given **Task**. If there are **CustomerOrders** in the queue, it will verify the fill status of the last **CustomerOrder** and fill it if the order contains the **Item** specified by the current **Task**.
* **bool MoveTask()** – moves the last **CustomerOrder** in the queue to the *next task* on the assembly line if the orders *fill state* for the current **Item** is *true*. Otherwise, the **CustomerOrder** should remain in the queue for the next processing cycle. If the queue is empty, return *false*.
* **void setNextTask(Task& task) –** stores the provided **Task** objects reference into the **pNextTask** pointer.
* **bool getCompleted(CustomerOrder& src)** – Removes the last **CustomerOrder** from the queue, places it in **src** and returns *true*. If the **CustomerOrder** queue is empty, return *false*.
* **Void Validate(std::ostream& os)** – writes the **name** of the **Item** this **Task** is responsible for to **os**. If **pNextTask** exists, writes the **name** of the next task on the assembly line.
* **Task& operator+=(CustomerOrder&& newOrder)** – Moves the **newOrder** onto the front of the **CustomerOrder** queue.

**LineManager Class**

A **LineManager** object manages the entire assembly line and contains the following information:

* **Std::vector<Task\*> AssemblyLine** – A container containing all the references of the Task objects on the assembly line
* **Std::deque<CustomerOrder> ToBeFilled** – A queue of **CustomerOrder’s** to be filled
* **Std::deque<CustomerOrder> Completed** – A queue of **CustomerOrder’s** completed
* **Unsigned int CustomerOrder\_Count** – The number of **CustomerOrder** objects the assembly line started with

**Constructor**

Upon instantiation, a **LineManager** object receives the following arguments:

* a reference to an unmodifiable **std::string**.  This string contains the filename specified by the user to be used for linking the assembly line objects (example: *AssemblyLine.txt*)
* a reference to a **std::vector<Task\*>** that contains the addresses of all the **Task** objects created for the assembly line
* a reference to a **std::vector<CustomerOrder>** that contains all the **CustomerOrder** objects to be filled

The constructor will:

* Read the AssemblyLine.txt file and setup all the **pNextTask** references in the **Task** objects. Linking each **Task** object together to form the assembly line (as specified in the *AssemblyLine.txt* file)
* Validate all the **Task** objects have been linked correctly
* Move all the **CustomerOrder** objects onto the front of the **ToBeFilled** queue
* Copy all the **Task** objects into the **AssemblyLine** container

**Member Functions**

* **bool Run(std::ofstream& os)** – This function performs one cycle of operations on the assembly line by performing the following:
  + If there are any **CustomerOrder** objects in the **ToBeFilled** queue, move the last **CustomerOrder** object onto the starting point of the **AssemblyLine**
  + Loop through all Tasks on the **AssemblyLIne** and run one cycle of the **Tasks** process
  + Loop through all Tasks on the **AssemblyLine** and move the **CustomerOrder** obects down the line.
    - HINT: If they are completed, pop them off the current **Task** and push them onto the **pNextTask**.
  + Return **TRUE** if all **CustomerOrder** objects have been filled, otherwise return **FALSE**.

**Milestone #3 – Sample Output**

The output for this milestone should look as follows:

CPU [123456] Quantity 5 Description: Central Processing Unit

Memory [654321] Quantity 10 Description: Basic Flash Memory

GPU [456789] Quantity 2 Description: General Porcessing Unit

SSD [987654] Quantity 5 Description: Solid State Drive

Power Supply [147852] Quantity 20 Description: Basic AC Power Supply

Manual Validation

getName(): CPU

getSerialNumber(): 123456

getSerialNumber(): 123457

getQuality(): 5

getQuality(): 4

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Elliott C.

Gaming PC

[0] CPU - MISSING

[0] Memory - MISSING

[0] GPU - MISSING

[0] GPU - MISSING

[0] GPU - MISSING

[0] SSD - MISSING

[0] Power Supply - MISSING

Chris S.

Laptop

[0] CPU - MISSING

[0] Memory - MISSING

[0] SSD - MISSING

[0] Power Supply - MISSING

Mary-Lynn M.

Desktop PC

[0] CPU - MISSING

[0] Memory - MISSING

[0] Power Supply - MISSING

Chris T.

Micro Controller

[0] GPU - MISSING

[0] GPU - MISSING

[0] Power Supply - MISSING

[0] SSD - MISSING

Filled Elliott C., Gaming PC[Power Supply]

Filled Elliott C., Gaming PC[CPU]

Filled Chris S., Laptop[Power Supply]

Filled Chris S., Laptop[CPU]

Filled Elliott C., Gaming PC[GPU]

Filled Elliott C., Gaming PC[GPU]

Filled Elliott C., Gaming PC[GPU]

Filled Mary-Lynn M., Desktop PC[Power Supply]

Filled Mary-Lynn M., Desktop PC[CPU]

Filled Elliott C., Gaming PC[Memory]

Filled Chris T., Micro Controller[Power Supply]

Filled Chris S., Laptop[Memory]

Filled Elliott C., Gaming PC[SSD]

Filled Mary-Lynn M., Desktop PC[Memory]

Filled Chris T., Micro Controller[GPU]

Filled Chris T., Micro Controller[GPU]

Filled Chris S., Laptop[SSD]

Filled Chris T., Micro Controller[SSD]

COMPLETED

Elliott C.

Gaming PC

[123458] CPU - FILLED

[654321] Memory - FILLED

[456789] GPU - FILLED

[456790] GPU - FILLED

[456791] GPU - FILLED

[987654] SSD - FILLED

[147852] Power Supply - FILLED

Chris S.

Laptop

[123459] CPU - FILLED

[654322] Memory - FILLED

[987655] SSD - FILLED

[147853] Power Supply - FILLED

Mary-Lynn M.

Desktop PC

[123460] CPU - FILLED

[654323] Memory - FILLED

[147854] Power Supply - FILLED

Chris T.

Micro Controller

[456792] GPU - FILLED

[456793] GPU - FILLED

[147855] Power Supply - FILLED

[987656] SSD - FILLED