Brewery simulation
(a little hard to do – but try anyway)

Mission

To simulate filling bottles in a brewery, by use of threads (task) and using locks, mutex's and semaphores.

Background

- C#Note chap. OOProg4 pp.2-11 + 17-22
- http://www.albahari.com/threading/
- http://www.albahari.com/threading/part2.aspx
- http://www.albahari.com/threading/part4.aspx

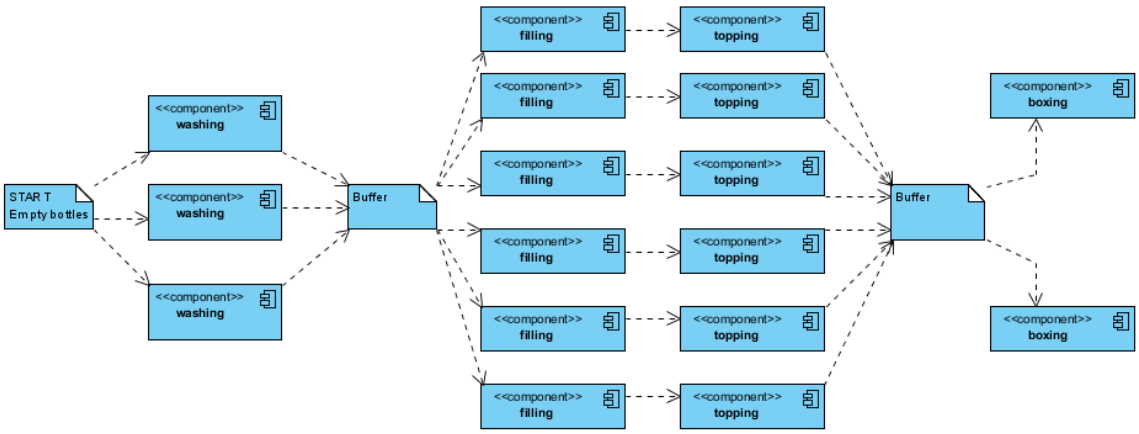
References to MS documentation:

- Lock https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/lock-statement
- Semaphore https://docs.microsoft.com/en-us/dotnet/api/system.threading.semaphore?view=netcore-3.0
- Mutex https://docs.microsoft.com/en-us/dotnet/api/system.threading.mutex?view=netcore-3.0
- Monitor https://docs.microsoft.com/en-us/dotnet/api/system.threading.monitor?view=netcore-3.0

Assignment

The idea is you should simulate the different processes in a bottling plant.

In overview this looks like



Empty Bottles are treated in a pipe-line - they comes in and they are

1. washed
2. filled with liquid
3. top the bottle
4. take the bottles into a bottle-box

In this simulation we have three washing machines in parallel.
You have six machines to fill the bottles as well as six machines as top the bottles but only two machines to take the bottles into boxes.

Some other constrains are

- that a bottle must first be filled if you(the machine) knows that the bottle can be topped immediately i.e. first fill when topping are ready.
- the machine to take bottles into boxes only is activated if there are 24 bottles ready.

You should use differently synchronization mechanism to simulate this brewery.

To solve this the buffers in the picture are to be bounded buffers
(be inspired by the producer – consumer problem):

A **bounded buffer** is a class (preferable a generic class), that hold a data structure (e.g. Queue). It should have two methods one to insert and one to take. A Bounded Buffer can have zero up to an upper limit of element inserted. To control the access introduce two semaphores one for empty and one for full. Before taking an item check the empty semaphore then take the item and possible release the full, and opposite when inserting. Remember that only one thread at the time can take / insert into the queue.

Step 0 create a model class of a bottle (with at least an Id)

Step 1 is to create a bounded buffer class (See above and Exercise Prog-4.4)

Step 2 create a thread that randomly (the time issue) generate bottles to be washed.

Step 3 consider how to start three threads in parallel

Step 4 Consider how to synchronized between Filling and bottling

Step 5 Create a filling Thread

Step 6 consider how to start six threads in parallel

Step 7 Create a bottling Thread

Step 8 consider how to start six threads in parallel

Step 9 Consider how to create a buffer to hold 24 bottles

Step 10 Create a boxing Thread

Step 11 consider how to start two threads in parallel

Additionel

Add a gui to the simulation to show the working threads and the respective buffers