Draft
(it will be finalised when the link is set up)

Main assignment

Pickup restaurant

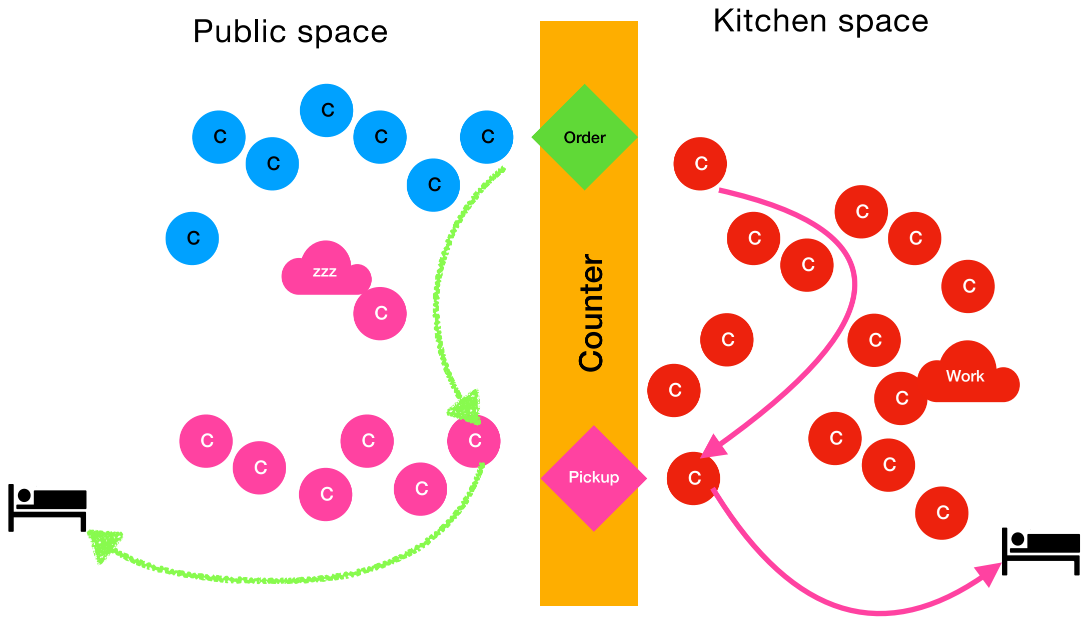
The setting:

We are setting up a fast pickup restaurant! The food is amazing! But there is only one dish served. You order it, wait, and get the first one you can. Are there already ready dishes? Take one and go after placing your order and waiting very little time! Don't worry about the others, they will not complain, they will not even notice it!

In this restaurant, there are two locations where clients and cooks can exchange goods and orders. The first one is where you place your order, the second one is where you pick it up.

After a client places the order, he will wait a certain amount of time and then go to the pickup location to get the order.

The cooks are behind the counter and furiously preparing the same dish repeatedly. It is a dish that takes some time but to avoid boredom for the cooks, every cook only cooks *one single dish every run*. After they are done with a single order, they can go home and rest.



The code:

The provided code might or might not work. It is not implemented concurrently. Your duty as a developer is to complete correctly the given application. The application should become **multithreaded** reaching the maximum level of concurrency, so everything that can be done in a concurrent setting should be implemented that way.

This **concurrent** application (command line only) requires no interaction with the user. Any interaction with the user is forbidden.

You must grant the property of **fairness** and **correctness** in your assignment.

Every part of your code must be completed so no random interleaving can alter the outcome. So, it must be **thread-safe**.

If the strategies to protect your memory are excessive (read: overlocking, or locking things that should not or do not need locks of any kind), they will be evaluated as invalid. Try to remember that is important to minimise the coverage of your locking (or number of semaphores). *Spinning* is not encouraged.

The desired outcome is:

- Each **cook** gets one order (and only one) from the order location and works independently on it to make it “done”, delivers it to the pickup location and terminates.

- Each **client** will generate independently an order at the order location, take a small sleep, and pick up the first available ready order at the pickup location. After that, it will terminate. The clients will get one and only one order (done) produced by the cooks.

A precondition for the assignment is that the number of clients and cooks is always the same at the beginning. So each cook and client will just run once and never be reused. This is a simplification for handling the concurrent work.

There is no need to check for speed of execution. But take care that each task gets executed by who should do it.

Breaks (“thread.sleep") are there for specific purposes and should not be removed or moved from their logical spot, there is no need to add more either. Similarly, printouts are there for testing and logical purposes. Do not remove them or alter them in the final delivery. During testing, feel free to debug it in the way you prefer, even altering or adding I/O operations, remember to clean up your code before the final delivery.

The main method will start elements in phases (already indicated in the method), feel free to change the content of the function call as needed.

You can create new classes if needed.

You can add variables if needed.

You can alter the signatures of some of the methods (where allowed) if needed.

We highly encourage testing: rise and/or diminish the number of dishes/tasks to be completed that should be prepared. Remember that testing over different memory conditions (over extra load/with less load) can facilitate the evidence of some errors or abnormal behaviours.

The provided code has comments to guide you and help you to address where to edit and where not to edit. Try to add as less changes as needed. Respect the structure and be considerate of what the comments are saying.

Note: the lists of orders and pickups at the end must always be zero. Every order must be ordered and picked up.

The delivery:

Every delivery group is made of **up to 2 students.** Only one student must submit the file(s). It is not allowed to work across classes unless explicitly approved. If the classes are joined (shared theory and/or practice) it is possible to cooperate without explicit approval. Retakers can choose anyone, no approval is needed.

If we can use groups in codegrade, you will find the information on how to enter the data there. If not, the names on the code will represent who is part of the group. In both cases, your code MUST include the names and student numbers of each component.

The delivery will be in codegrade. Late submissions, email submissions, and chat submissions will be discarded.

Deliveries should NOT contain the bin/obj directory.

The link to the submission will be here, any other info regarding the submission of your delivery will be linked from this link.

Deadline: **January 26, 2024 at 11:30 pm**.

The testing and evaluation:

If your assignment fails to meet any of the following points, your grade might get a **negative assessment** with no further testing.

- You do NOT require user interaction.
- Your code compiles correctly and does NOT require versions of ".Net" other than ".Net 6.x".
- Your code does NOT show any syntax errors.
- Your code does NOT crash.
- You are NOT using an auto-locking data structure (concurrent queues, blocking data structures, other data structures, or primitives that handle threads automatically, ex.: task library or "promises-like" functions of different kinds).
- You are NOT using parallel-for or other facilitating in-language control systems that avoid using threads in a raw context. Async-await structures are part of this category and thus: NOT allowed.
- You are NOT using external libraries that are not the standard Threads/mutex/locks/semaphores.
- Your code is cross-platform (the solution will NOT only run on your system).
- You understand every single part of your code and which are the implications (to keep in mind: according to the course description an oral check is possible at any time to clarify your contribution).
- Your code is original (to keep in mind: according to the course description an oral check is possible at any time to clarify your contribution).
- Your delivery does NOT contain *bin* and *obj* directories.
- Your code implements what is requested.
- Your code does NOT implement some kind of a sequential version of the task provided.
- The delivery format is as requested (file naming, compression method for the solution, etc…).

The usage of any code-generating facilitation is forbidden (**chatgpt**-like or other autogenerating systems). Keep in mind that platforms of that kind tend to give always similar results to each other.

Code linter/autocomplete—> ok

Copilot-like—>**forbidden**.

What will be tested?

Besides the previous requirements and minimal decency of code quality, we will check that all the shared memory is correctly protected and that the algorithm that synchronises the various tasks is carried out correctly by who should do it. Your code must be granting fairness and correctness in the application (for fairness, it should be all delegated to the OS, which is assumed fair). We will check if the protection applied is sufficient and necessary, and evaluate negatively what is overprotected.

How will it be tested?

Partially automatically and partially by hand. The "automatic" testing will be on compiler errors/warnings, formatting of the files, and printout results (in case the reviewer needs to, will also run these tests by hand). The "by hand" part will be carried by command line calls as "dotnet run" in the project directory. Are you using Visual Studio? Consider testing if everything is working by running the code via the command line interface. So, you can be sure that everything is OK, and with the same setting as the provided project.

Common questions:

Do you doubt a concept relevant to your code? You can ask in the forum, but please, refrain from asking questions (in public or private) as "Is this ok?" or "Is it sufficient like this?". Teachers are not allowed to answer such questions to prevent unfair evaluation.

Do you not know where to start? Check all the exercises, we are making use of the information needed there, and we used them and discussed them in class.

I forgot XYZ in the delivery, can I change it? You will have time to change it till the deadline, and past the deadline, the assignment will be considered as delivered.

I forgot to fill in the name of my partner, can I add it? You will have time to change it till the deadline, and past the deadline, the assignment will be considered as delivered.