Concurrent and Distributed Programming

Sea port project

Suppose you are programming a system in which, for now, we want to control the inputs and outputs of a sea port. We have two types of ships: those who want to enter and those who want to leave the port. Additionally, we have a Gate by which ships enters and exits.

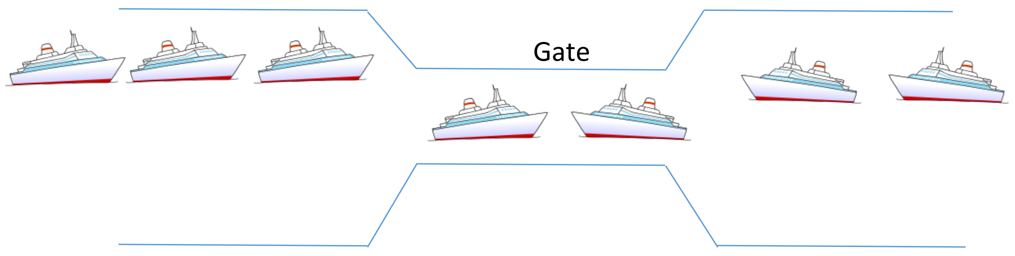

Figure 1 Step 1: Ships entering and leaving the gate (in this case 2 ships collide at the gate)

This practice will grow incrementally. Each session will introduce changes that will help us to understand some of the concepts that we have previously introduced in the theory classes. Each modification will be called step.

Session 1

Step 1.-

We will create a project called `Ships`.

Create a `Gate` class that implements the methods *`enter`* and *`exit`*. Create a *`Ship`* class that calls the method *`exit`* or *`enter`* on the `Gate` class (depending on the type of the ship – exit or enter).

The fact of entering or exiting the port is simulated by a message printed 3 times "The X ship enters / exits", where X is the *`id`* of the *`Ship`* received as a parameter in the constructor. The ship also receives another parameter that indicates if the ship is leaving or entering the port.

Create a *`main program`* to launch various threads *`Ship`*, some who want to enter and the others wanting to leave (for example, 10 of each type).

In this solution, do not worry if the ships collide in the *`Gate`* (Figure 1). This will occur if the messages interleave undesirably.

You have a possible UML class diagram of the solution in Figure 2.

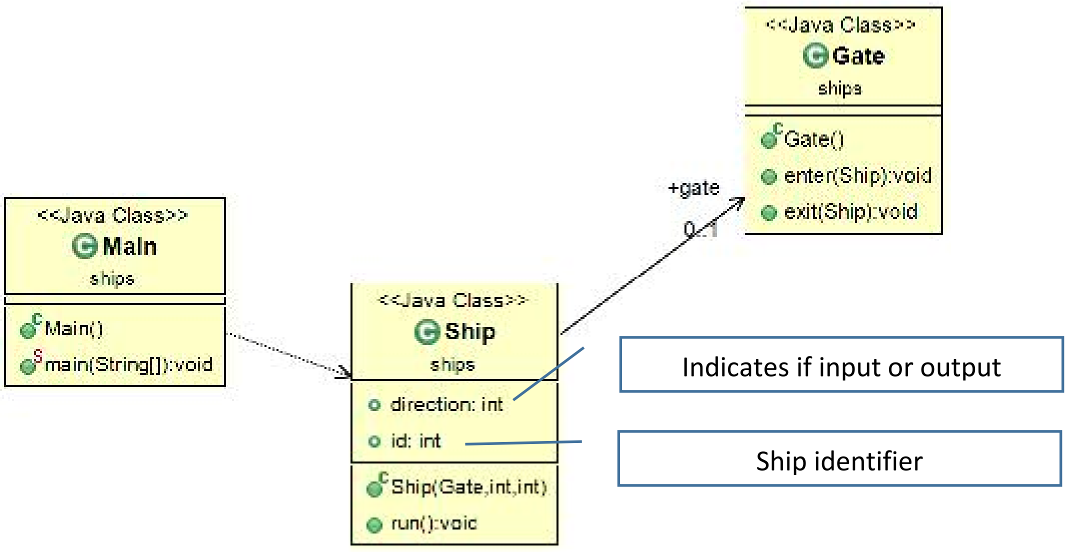

Figure 2 Class diagram for Step 1

Sessions 2 and 3

Step 2.-

Modify the program so that only one ship can pass through the control gate. This requires using synchronized.

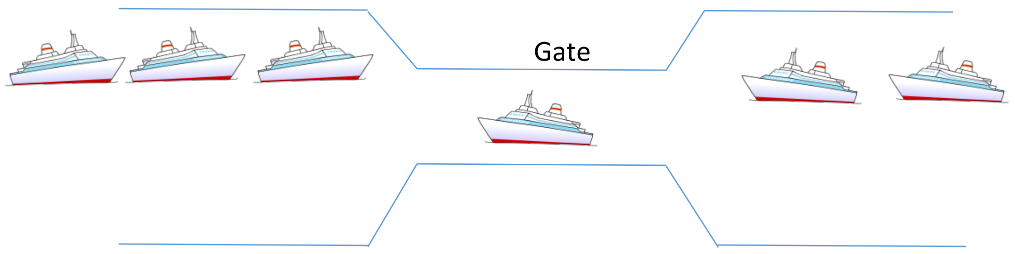

Figure 3 Step 2: Just a ship enters or leaves at the same time

Step 3.-

Now more ships can go through the control gate at the same time, provided that all go in the same direction. If I want to pass and there is another ship passing in the same direction, then I can do it. If there is no one passing, then I can pass. If someone is passing in the opposite direction, then I cannot pass and I have to wait.

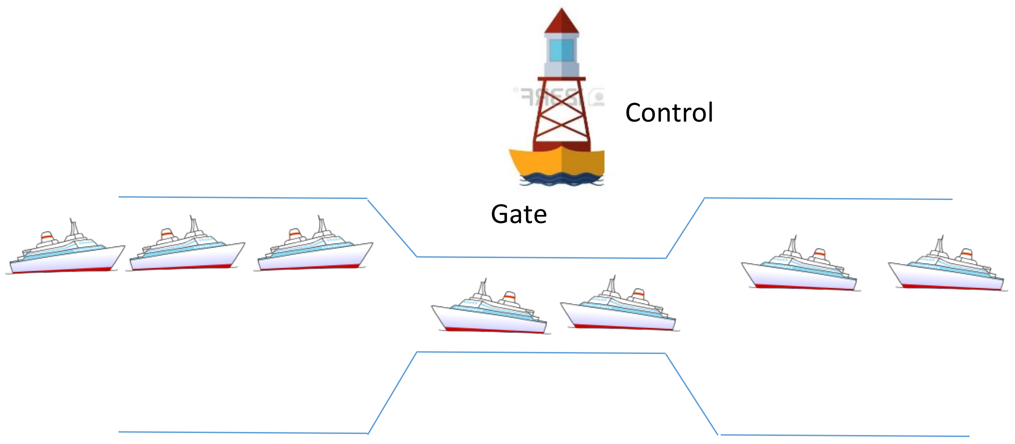

Figure 4 Step 3: Several ships going in the same direction

Since this is going to be complicated gradually, we will choose to create a new class called `Control` which is responsible for managing permits 'entry' and 'exit' through the *`Gate`*. You shall have the following methods in this class: *`entryPermission`*, *`exitPermission`*, *`entryNotification`* and *`exitNotification`*. Now, the ship will invoke the *`entryPermission`* method on the `Control` class. When a *Ship* finishes entering, then the method *`entryNotification`* is invoked so the `Control` class can perform the appropriate actions. The same holds for exiting. In Figure 5 you have a possible class diagram for the solution proposed (you can use other diagram).

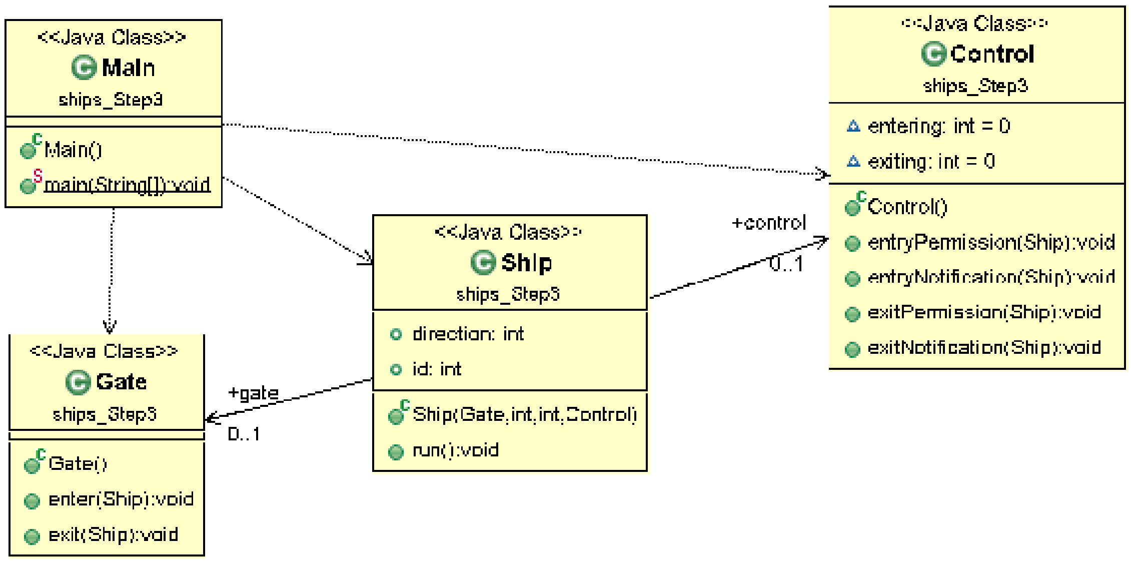

Figure 5 Step 3: Class diagram for step 3

Step 4.-
Additionally, now those ships who want to leave have preference. If there is any ship that wants to enter, but there is a ship that is waiting to leave, then he cannot enter. It will only do it when no one is exiting or waiting to exit.

Step 5.
Besides the above, we want to respect the arrival order. If a ship wants to enter and it cannot, we must ensure that when it enters it do it before those ships that arrived to enter after him. The same holds for exiting.

The final deliverable of these first 5 steps is one project. We'll comment in class how to submit the project.

Session 5 - Lesson 4.2 Monitors

Step 6.

One of the ships entering the port is a merchant ship carrying containers of sugar, salt and flour. When the ship enters the port, it goes to a different Cargo area in which there are waiting three cranes. The crane 1 can only get containers of sugar. Crane 2 only can get flour and crane 3 only can get containers of salt. The ship also has its own crane. While the ship has containers, it will leave them in a platform (1 by 1) due to the platform having capacity of 1 container. The appropriate crane of the Cargo area will get the container depending on the type of the container (salt, flour or sugar).

Suppose that initially there are 12 containers of sugar, 20 flour and 5 of salt. When the ship has discharged all of its containers, then it leaves the port.

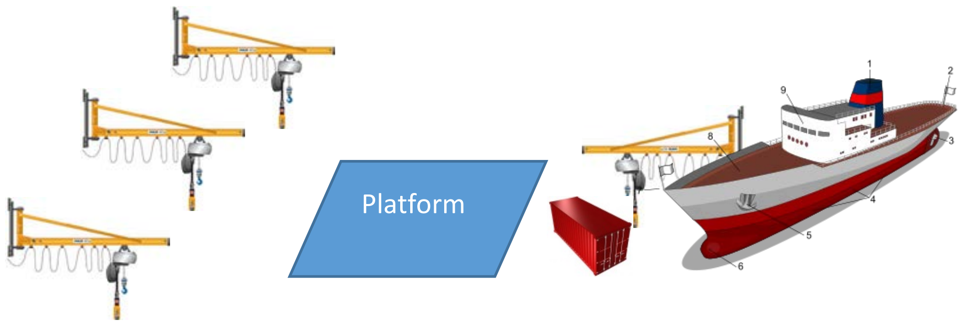

Figure 6 Step 8: Zone for unloading containers

Session 5 - Lesson 4.1 Semaphores

Step 7

Five of the ships entering the port will be Oil ships.[1]. As soon as they enter the port they go to a Cargo area. There they can get oil and water. For each ship the cargo area has a private oil container for that ship, but the water is common to all 5 ships (they share a common container). When the 5 ships have reached the loading area **(only when the 5 have reached**), then the behavior of each ship is as follows: a) load 3,000 liters of oil; b) load 5,000 liters of water.

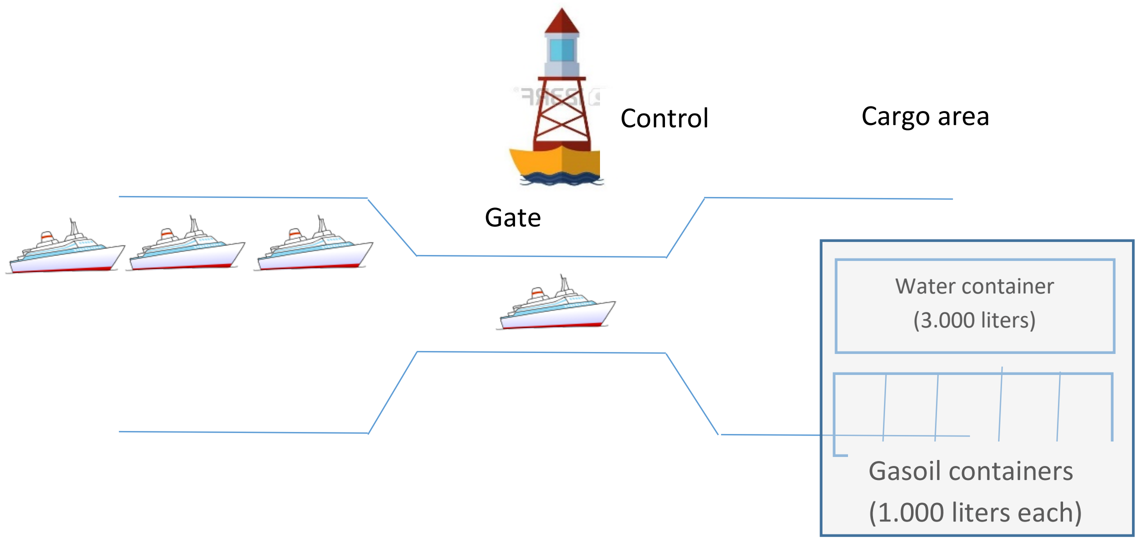

Figure 7 Step 6: Loading zone diesel and water

It is assumed that the amount of water is infinite, but not the amount of oil in each container of the Cargo area, which is 1,000 liters. When the oil is finished in the 5 containers in the cargo area (**the 5 containers must be empty**), then a process `Filler` will fill the 5 containers of oil of the Cargo area. The container capacity of each ship is 3,000 liters of oil and 5,000 liters of water. The ship picks up always a quantity of 1,000 liters from the containers of the Cargo area. When the ships are full, then they leave the port again (is not necessary to wait for each other). When a ship is getting water, no other ship can get water. However, 5 ships can be filling their containers of oil at the same time.

You have to solve it with semaphores.

Session 6 – Lesson 5 Advanced Concurrent Programming in Java

Step 8

Now the filling of oil and water will be done concurrently. To do this we will create two threads for each ship, each one doing a task. When the two threads have completed their task, the ship can continue. You should create the threads with Executor. Observe that now the threads are not created in the main method. You should create them in the appropriate ships.

[1] This will involve adding a new attribute to ship oil indicating whether it is an oil ship or not (and a parameter to the constructor)

Step 9

In theory classes we will have seen different classes and frameworks that allow us to simplify the creation and synchronization of threads.

In this step of the practice is asked to redo the sea port project using the new classes and frameworks learned. The solution will be quite simpler and we will understand perfectly why those classes and frameworks arose. Feel free to use them as you want. You should use at least three of them.

In your defense of the practice you should discuss what classes and frameworks you used, where and why.

End of the statement of the concurrent part