A description of semaphore and how it works:

semaphores coordinating resource to prevent conflict on resources (synchronisation techniques). They are used to implement synchronisation between processes, impose mutual exclusion, and prevent race situations.

Wait (acquire) and signal (release) are the two operations offered by semaphores. The semaphore's value is decreased by the wait operation while it is increased by the signal operation. Any process that executes a wait action will be blocked until another process executes a signal operation when the semaphore's value is zero.

Critical sections—areas of code that can only be performed by one process at a time—are implemented using semaphores. Processes can coordinate access to shared resources, like shared memory or I/O devices, by utilizing semaphores.

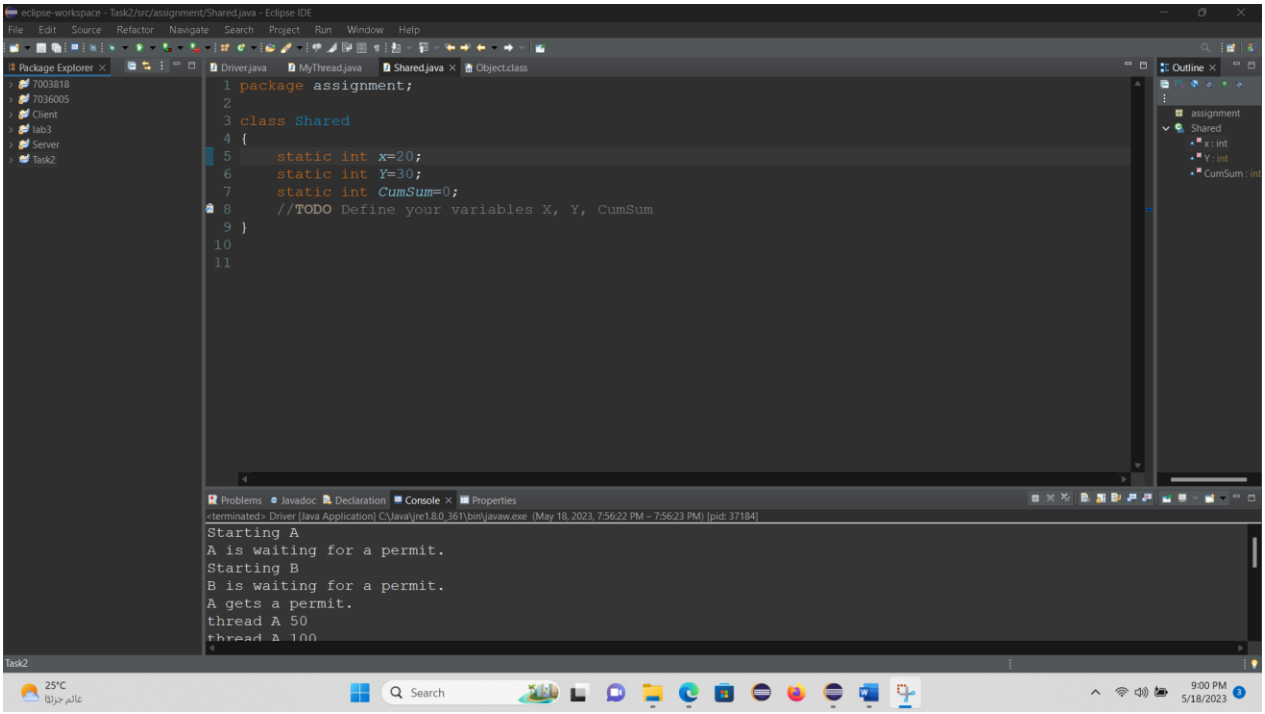
A discussion about what you used in the assignment:

The program uses a semaphore to control access to the CumSum,X,Y variables, which is a static variable within the Shared class. Shared.CumSum performing operation CumSum=((x + y) + CumSum) five times by thread A and performing operation CumSum=(CumSum - (x + y)) five times by thread B .To prevent these two threads from accessing Shared.CumSum and Shared.X and Shared.Y at the same time, access is allowed only after a permit is acquired from the controlling semaphore. After access is complete, the permit is released. In this way, only one thread at a time will access Shared.CumSum and Shared.X and Shared.Y , as the output shows.

Notice the call to sleep() within run() method inside MyThread class. It is used to prove that accesses to Shared.CumSum and Shared.X and Shared.Y are synchronized by the semaphore. In run(), the call to sleep() causes the invoking thread to pause between each access to Shared.CumSum and Shared.X and Shared.Y . This would normally enable the second thread to run. However, because of the semaphore, the second thread must wait until the first has released the permit, which happens only after all accesses by the first thread are complete. Thus, Shared.CumSum is first incremented by value =(Shared.X+Shared.Y) five times by thread A and then decremented by value =(Shared.X+Shared.Y) five times by thread B.

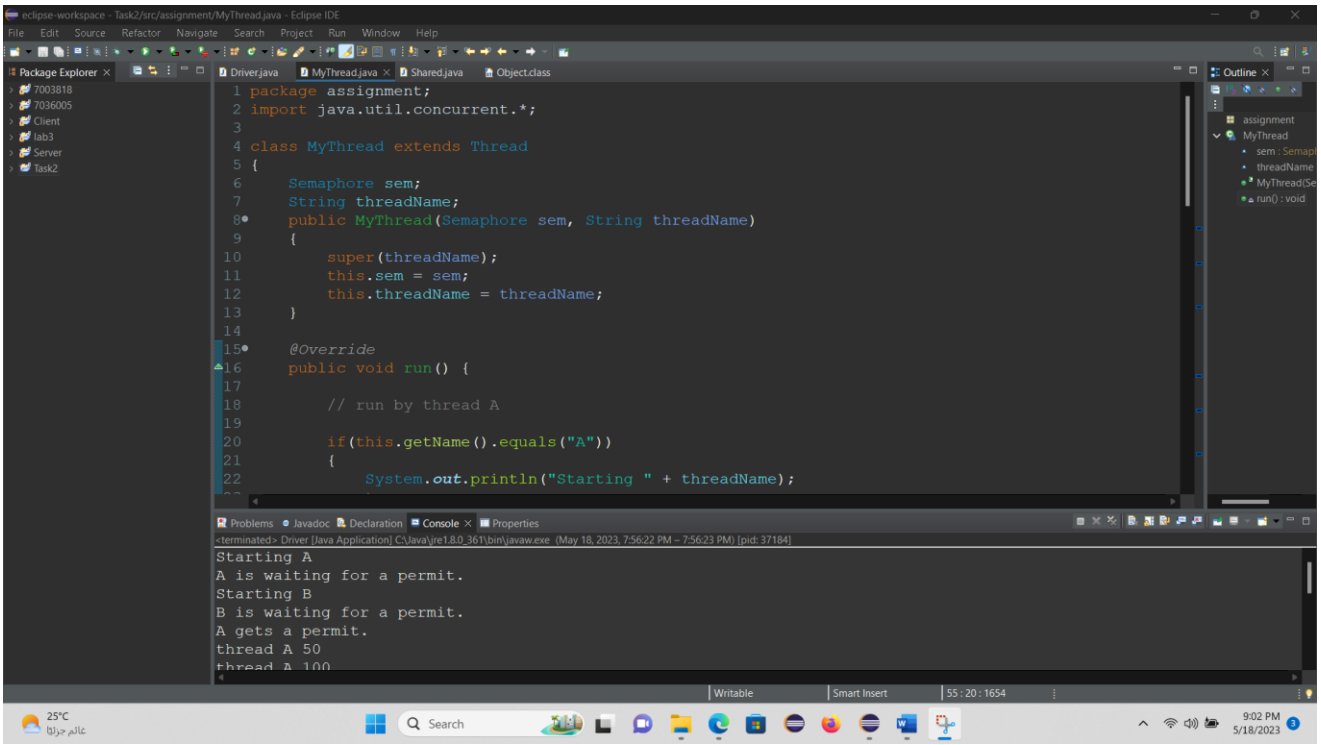
Step 1

We started by initializing the variables X=20 , Y=30 , CumSum = 0 in Shared Class .



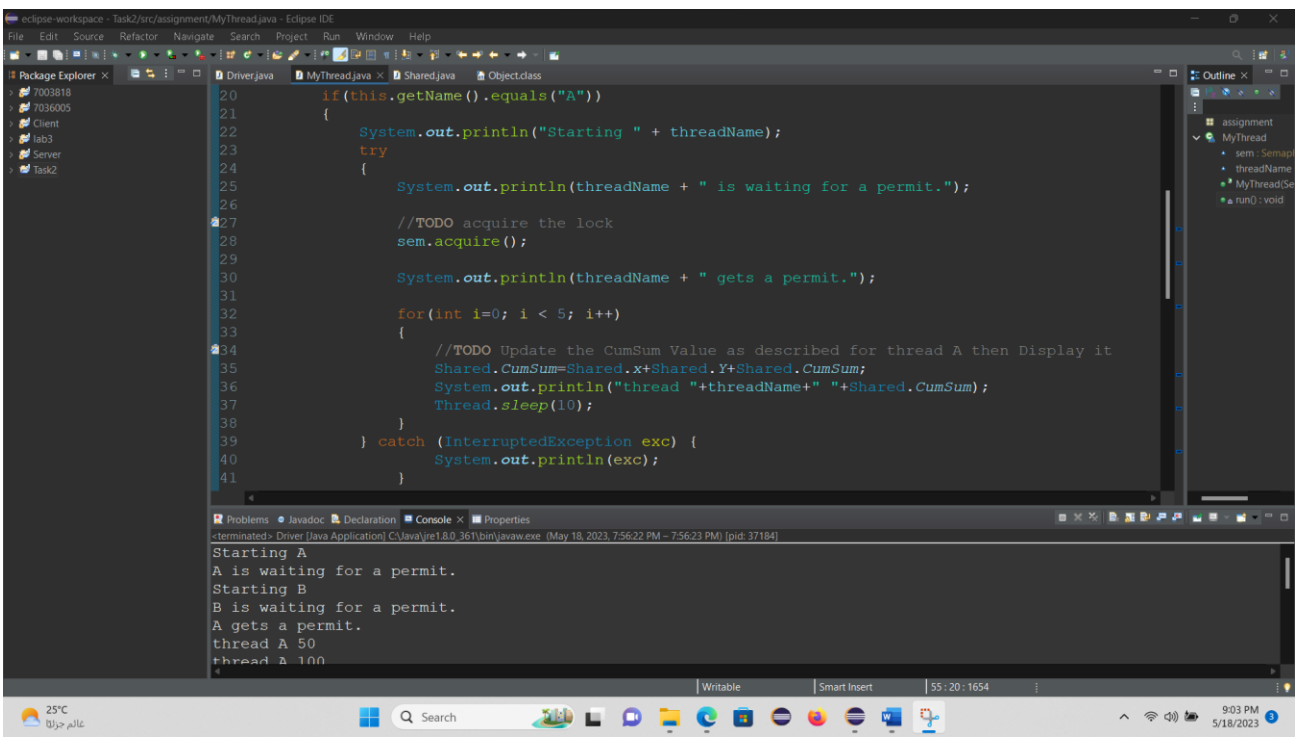
Step 2

Then we extended our MyThread class to thread class to allow threads run in parallel and overriding run method .



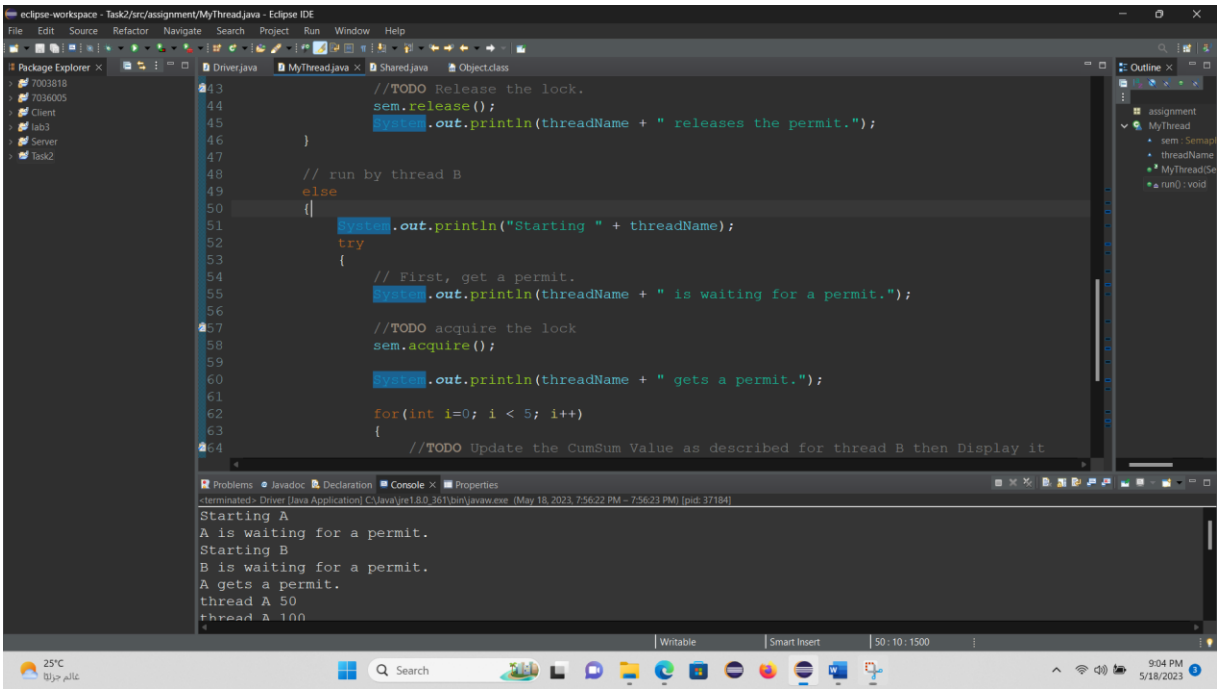
Step 3

We check Name of the thread if it equal to ‘A’ then we print process Name and surround body with try and catch statements as it may throws InterruptedException then in body of try statement printing the Thread name and locking the resource then printing that the thread get a permit then we will keep incrementing the CumSum by value equal to (X+Y) for 5 times and at each time we will print the new value of CumSum in each time and we will make the thread sleep for 10 milli seconds in each iteration and in case of exception has been thrown we will print the exception



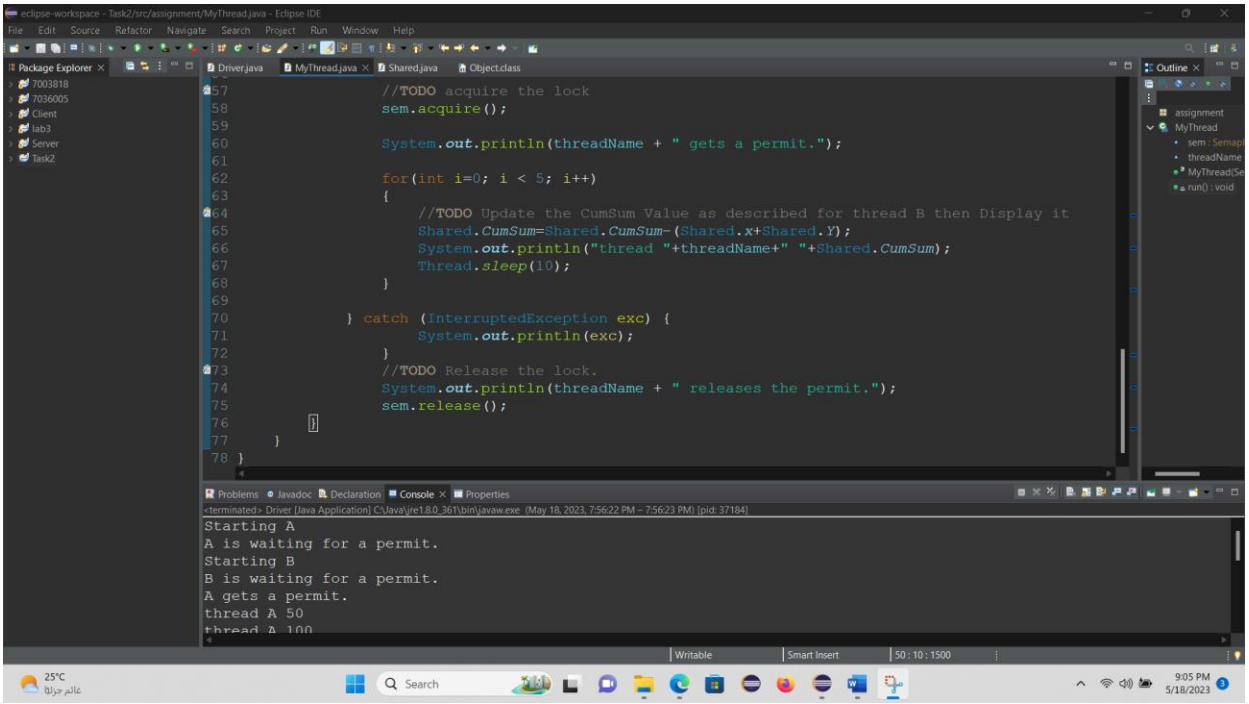
Step 4

We will release the resource and printing that thread released the resource. In case if it is not 'A' then it should be 'B' then we print process Name and surround body with try and catch statements as it may throws InterruptedException then in body of try statement printing the Thread name then we wait to get a permit and printing that thread is waiting untill it accuire the resource after it gets hold on the resource , we lock the resource then printing that the thread get a permit



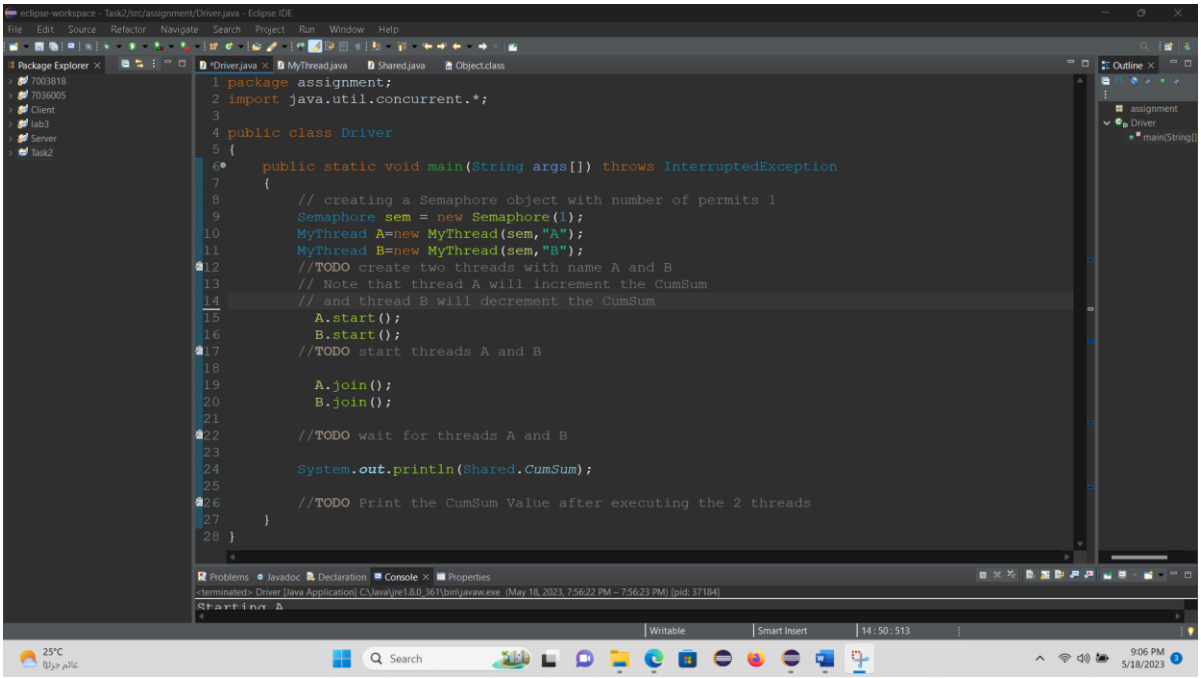
Step 5

then we will keep decrementing the CumSum by value equal to (X+Y) for 5 times and at each time we will print the new value of CumSum in each time and we will make the thread sleep for 10 milli seconds in each iteration and in case of exception has been thrown we will print the exception and then We will release the resource and printing that thread released the resource.



Step 6

In driver class we will create a semaphore (sem) and initialize it with 1 which is initial permit count and we will create 2 Threads from MyThread , MyThread A will take a value (sem, 'A') , and MyThread B will take a value (sem, 'B') then will start the 2 threads to run in parallel Then we will wait for thread A and B to finish running without conflict on resource each one at a time , one will wait till the other running till death then the waiting one will start running till death also Then printing the final value for the CumSum



Step 7

In the output we can see that B started running then A , then A waited for B, then B waited for A , then A get the permit and we kept incrementing the CumSum in Thread A by value X+Y which is 50 for 5 times (50,100,150,200,250) then A thread died ,then B thread gets a permit to use the resource as A finished and ,then A released it , then kept decrementing the CumSum in Thread B by value X+Y which is 50 for 5 times (200,150,100,50,0) then A thread died released the resource