# Assignment: Smart Parking System Using Async/Await in .NET

## Objective

This assignment requires students to develop a smart parking system simulation using async/await in .NET.  
Students will need to:  
- Implement asynchronous methods to simulate different steps of a parking system.  
- Ensure that some steps run sequentially while others run concurrently using Task.WhenAll().  
- Use Task.Delay() to simulate real-time processing.

## Scenario

A smart parking system is being developed to handle the following process:  
1. Vehicle arrival: The system checks the parking ticket before allowing entry.  
2. Entry process: If the ticket is valid, the barrier opens, and the vehicle parks.  
3. Exit process: The vehicle makes a payment, and the system updates the database before allowing exit.  
4. Data update: The system records entry and exit times for reporting.  
  
Some steps must be executed sequentially (e.g., a ticket must be verified before opening the barrier), while others can run concurrently (e.g., updating the database and opening the exit barrier).

## Requirements

- Implement an asynchronous smart parking system using C# and .NET.  
- Some steps must be executed sequentially using await.  
- Some steps can be executed in parallel using Task.WhenAll().  
- Simulate real-time execution using Task.Delay().  
- Follow object-oriented principles.

## Implementation Details

### 1. Define the SmartParkingSystem class

using System;  
using System.Threading.Tasks;  
  
class SmartParkingSystem  
{  
 // Simulate checking a parking ticket  
 public async Task CheckTicketAsync(string carNumber)  
 {  
 Console.WriteLine($"[Car {carNumber}] Checking ticket...");  
 await Task.Delay(2000); // Simulating processing time  
 Console.WriteLine($"[Car {carNumber}] Ticket is valid!");  
 }  
  
 // Simulate opening the entry barrier  
 public async Task OpenEntryBarrierAsync(string carNumber)  
 {  
 Console.WriteLine($"[Car {carNumber}] Opening entry barrier...");  
 await Task.Delay(1500);   
 Console.WriteLine($"[Car {carNumber}] Entry barrier opened.");  
 }  
  
 // TODO: Implement parking logic  
 public async Task ParkCarAsync(string carNumber)  
 {  
 // Student implementation  
 }  
  
 // TODO: Implement payment processing  
 public async Task ProcessPaymentAsync(string carNumber)  
 {  
 // Student implementation  
 }  
  
 // TODO: Implement logic for opening the exit barrier  
 public async Task OpenExitBarrierAsync(string carNumber)  
 {  
 // Student implementation  
 }  
  
 // TODO: Implement database update logic  
 public async Task UpdateDatabaseAsync(string carNumber)  
 {  
 // Student implementation  
 }  
  
 // Implement the process for a car entering the parking lot  
 public async Task CarEnterAsync(string carNumber)  
 {  
 await CheckTicketAsync(carNumber);  
 await OpenEntryBarrierAsync(carNumber);  
 await ParkCarAsync(carNumber);  
 Console.WriteLine($"[Car {carNumber}] Successfully parked.\n");  
 }  
  
 // Implement the process for a car exiting the parking lot  
 public async Task CarExitAsync(string carNumber)  
 {  
 await ProcessPaymentAsync(carNumber);  
 await Task.WhenAll(OpenExitBarrierAsync(carNumber), UpdateDatabaseAsync(carNumber));  
 Console.WriteLine($"[Car {carNumber}] Successfully exited.\n");  
 }  
}

### 2. Implement the Main method

using System;  
using System.Threading.Tasks;  
  
class Program  
{  
 static async Task Main(string[] args)  
 {  
 SmartParkingSystem parking = new SmartParkingSystem();  
  
 Console.WriteLine("Smart Parking System is starting...\n");  
  
 // Simulate a car entering the parking lot  
 await parking.CarEnterAsync("A123");  
  
 // Simulate waiting time before exit  
 await Task.Delay(5000);  
  
 // Simulate a car exiting the parking lot  
 await parking.CarExitAsync("A123");  
  
 Console.WriteLine("Parking system transaction completed!\n");  
 }  
}

## Tasks for Students

1. Complete the ParkCarAsync(), ProcessPaymentAsync(), OpenExitBarrierAsync(), and UpdateDatabaseAsync() methods.  
2. Ensure that:  
 - The entry process follows sequential execution (check ticket → open barrier → park).  
 - The exit process runs payment first, then executes opening the barrier and updating the database in parallel.  
3. Run the program and verify that the output follows the expected behavior.  
4. Modify the Task.Delay() times to experiment with different delays.

## Expected Output

Smart Parking System is starting...  
  
[Car A123] Checking ticket...  
[Car A123] Ticket is valid!  
[Car A123] Opening entry barrier...  
[Car A123] Entry barrier opened.  
[Car A123] Searching for a parking spot...  
[Car A123] Successfully parked.  
  
(Wait for some time...)  
  
[Car A123] Processing payment...  
[Car A123] Payment successful!  
[Car A123] Opening exit barrier...  
[Car A123] Updating database...  
[Car A123] Exit barrier opened.  
[Car A123] Successfully exited.  
  
Parking system transaction completed!

## Additional Challenges

- Implement a concurrent parking lot system where multiple cars can enter and exit at the same time.  
- Introduce a maximum parking limit where cars must wait if the parking lot is full.  
- Add error handling (e.g., if a ticket is invalid, the car cannot enter).

## Submission

Students must submit:  
1. The completed SmartParkingSystem class.  
2. A brief explanation of how async/await and Task.WhenAll() are used in their implementation.  
3. A screenshot of the program output.

Yêu cầu: In ra ProcessId , ThreadId đang chạy, đếm Total Thread của Process này