# Part A – Events

## Problem 1 – Water Tank Level Alert (void delegate)

• Create a class WaterTank with an event TankFull (no parameters).

• Method Fill() increases water level.

• Trigger TankFull when level reaches 100.

• Subscriber: StopPump() → prints 'Pump stopped, tank is full!'

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Problem 2 – Light Sensor (delegate with parameters)

• Create a class LightSensor with an event LightChanged that sends int luxLevel.

• Method SetLightLevel(int level) triggers the event.

• Subscribers:

- TurnOnLamp(int luxLevel) → prints 'Lamp turned on, lux: {luxLevel}'

- LogLightChange(int luxLevel) → prints 'Light level changed to {luxLevel}'

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Problem 3 – Online Order (EventHandler built-in)

• Create a class Order with an event OrderPlaced using EventHandler.

• Method PlaceOrder() triggers the event.

• Subscriber: SendConfirmation → prints 'Order placed successfully!'

## Problem 4 – Door Lock (EventHandler with Custom EventArgs)

• Create a class DoorEventArgs inheriting EventArgs with a boolean IsLocked.

• Create class DoorLock with an event DoorStateChanged using EventHandler<DoorEventArgs>.

• Method LockDoor() triggers the event.

• Subscribers:

- ShowDoorStatus(object sender, DoorEventArgs e) → prints 'Door is locked!' or 'Door is unlocked!'

- PlayLockSound(object sender, DoorEventArgs e) → prints 'Lock sound played!'

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Part B – Async & Await

## Problem 7 – Coffee Machine (Async delay)

• Create an async method BrewCoffeeAsync() that waits 3 seconds (Task.Delay) and prints 'Coffee is ready!'.

• Call it from Main and await the result.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Problem 8 – File Processing (Parallel Async)

• Create two async methods:

- ReadFileAsync() → waits 2 seconds, prints 'File read complete'

- AnalyzeFileAsync() → waits 1 second, prints 'File analysis complete'

• Call both from Main simultaneously and wait for both using Task.WhenAll.