**Drawbridge Simulator - Documentazione**

**Struttura del Progetto**

Il progetto simula l'attraversamento di macchine su un ponte a 4, con gestione della concorrenza tramite semaforo.

**File del progetto**:

1. Ponte.cs: Contiene la logica principale del ponte e della simulazione

2. Program.cs: Gestisce l'interfaccia utente e il flusso del programma

**Classi e Metodi Principali**

**Classe `Ponte`**

Il costruttore inizializza le liste delle macchine e il semaforo, inoltre crea la visualizzazione iniziale del ponte in modo tale che possa situarsi sotto il menù del main.

**Proprietà**

- `PonteLevatoio`: Semaforo che limita a 4 il numero di macchine contemporanee

- `MacchineSx`/`MacchineDx`: Liste delle macchine in coda

- `NrMacchine`: Contatore progressivo delle macchine

**Metodi Pubblici (quelli utilizzabili e utilizzati nel main)**

Aggiunta macchine (aggiunge le macchine nelle code a destra e a sinistra, assegna un id alle macchine)

AttraversanoMacchine (coordina l’attraversamento delle macchine, prima quelle a destra e poi quelle a sinistra)

**Metodi Privati (in quanto compongono la struttura del programma, tenerli pubblici sarebbe rischioso)**

CreateBridge:

Disegna il ponte nella console:

1. Linea separatrice orizzontale
2. 4 corsie rappresentate da linee tratteggiate

PassaMacchinaSxDx, PassaMacchinaDxSx

Gestiscono il passaggio singola macchina con:

1. Controllo del semaforo
2. Assegnazione della corsia (`index % 4`)
3. Gestione delle risorse con `try/finally`

AttraversamentoSxDx, AttraversamentoDxSx

Animano lo spostamento delle macchine:

1. Movimento progressivo (5 caratteri ogni 500ms)
2. Cancellazione della posizione precedente
3. Uso di `lock` per “bloccare” la console

**Classe `Program`**

**Metodo Main**

- Punto di ingresso del programma

- Gestisce il loop principale e le interazioni utente

**Metodi di Supporto**

RedrawMenu

Ridisegna il menu principale mantenendo:

1. Opzioni disponibili
2. Conteggio macchine in coda

**Costrutti Utilizzati**

**1. Programmazione Asincrona:**

- `async`/`await` per gestire gli attraversamenti senza bloccare l'UI

- `Task.WhenAll` per parallelizzare gli attraversamenti

**2. Sincronizzazione:**

- `SemaphoreSlim` per limitare a 4 gli attraversamenti contemporanei

- `lock (Console.Out)` per evitare race condition nell'output

**3. Gestione Console:**

- `Console.SetCursorPosition` per controllo preciso dell'output