MODELO Examen Programación Servicios y Procesos – Trimestre 1 – 2º DAM O/U

NOMBRE Y APELLIDOS:

Ejercicio 2 (1,5 puntos)

Tenemos un programa en el que simulamos que múltiples jugadores están jugando a un juego *online*. Todos los jugadores juegan en un mismo equipo y en un mismo escenario. El objetivo de los jugadores es eliminar a los enemigos del escenario. Cada vez que un jugador elimina a un enemigo, se incrementa la puntuación del equipo. Al finalizar la partida, cuando se han eliminado a todos los enemigos, se quiere mostrar por pantalla la puntuación por jugador. Esta puntuación es la puntuación total del equipo dividido entre todos los jugadores. Por ejemplo, si la puntuación del equipo son 1000 puntos y hay 10 jugadores, la puntuación por jugador será de 100.

La implementación actual no funciona correctamente y no sabemos por qué. Trata de arreglar el código sin cambiar la lógica del programa y haciendo los mínimos cambios necesarios. Justifica claramente en el código los cambios que has hecho.

Ejercicio 3 (3 puntos)

Queremos simular tres jugadores de baloncesto pasándose una pelota. Implementa un programa siguiendo las siguientes indicaciones:

- Existirá una clase *Pelota*. Esta clase únicamente sobrescribirá el método *ToString* para devolver el mensaje "Bota, bota la pelota".
- Existirá una clase *Jugador*. Cada jugador será un hilo que hará lo siguiente:
    1. Si el jugador no tiene la pelota, esperará a recibirla.
    2. Si tiene la pelota:
        2.1. Mostrará un mensaje con el nombre del jugador seguido del *String* devuelto por el objeto *pelota*. Por ejemplo:
        Jugador 1: Bota, bota la pelota
        2.2. A continuación, hará una espera de 4 segundos.
        2.3. Pasará la pelota al siguiente jugador.
        2.4. Volverá a esperar a que le llegue la pelota.

El jugador 1, le pasará la pelota al jugador 2, el jugador 2 al jugador 3 y el jugador 3 al jugador 1. Repitiéndose indefinidamente hasta que paremos el programa.

Importante:

- Existirá una única instancia de la clase *Pelota*. Esta instancia se irá pasando de un jugador al siguiente. Es decir, la pelota no puede estar en posesión de varios jugadores.
- Es imprescindible utilizar los mecanismos de sincronización *notify* y *wait*.

Ejemplo de ejecución:

```
Jugador 1: Bota, bota la pelota
Jugador 2: Bota, bota la pelota
Jugador 3: Bota, bota la pelota
Jugador 1: Bota, bota la pelota
Jugador 2: Bota, bota la pelota
```

MODELO Examen Programación Servicios y Procesos – Trimestre 1 – 2º DAM O/U

NOMBRE Y APELLIDOS:

Ejercicio 4 (1,5 puntos)

Tenemos un programa en C# que llama a un método *TareaCostosa* que simula ejecutar una tarea computacionalmente compleja. La llamada es síncrona y queremos que se ejecute en un hilo mediante una tarea

Para ello, crea un nuevo método *TareaCostosaAsync* que ejecute el método *TareaCostosa* dentro de una tarea. Modifica el método *Main* para llamar a *TareaCostosaAsync* en vez de *TareaCostosa*.

Si los cambios que has realizado son correctos, el mensaje “Este mensaje se debería mostrar mientras se realiza la tarea costosa…” debería mostrarse, como dice el texto, mientras se ejecuta *TareaCostosaAsync* y no cuando finalice. Después de mostrar ese mensaje, el programa debería mostrar también por pantalla el resultado devuelto por *TareaCostosaAsync*. Dicho de otro modo, la llamada a *TareaCostosaAsync* no debería ser bloqueante.