

COLEXIO VIVAS S.L.	RAMA:	Informática	CICLO:	DAM		
	MÓDULO	Servicios				CURSO: 2º
	PROTOCOLO:	Examen	AVAL:	1ª	DATA:	27/11/2023
	Nombre y Apellidos:					

Normas:

- Crea una solución de Consola en VS2022 denominado **ApellidosNombreSERV_1aEv**.
- Recuerda poner además tu nombre y apellidos como comentario al principio del archivo de código.
- No se permite el uso de funciones no estudiadas de la clase Thread como Abort, Suspend, IsAlive, Interrupt, Resume. Si tienes dudas pregunta al profesor si se permite su uso.
- Tampoco se deben abusar de elementos (lock, booleanas, condiciones, etc.) si no son necesarios.
- La puntuación se basa principalmente en el correcto uso, control y sincronización de los hilos por lo que esto debe realizarse de forma correcta para que el apartado correspondiente puntúe.
- La definición de variables debe estar correctamente tipado. El uso de var o dynamic no está permitido (salvo que se imprescindible en alguna situación).
- En la medida de lo posible debes usar inglés para los nombres de los identificadores así como en las cadenas usadas.

Ejercicio 1. Omelette in Elm Street

Nuestro querido Freddy K. tan metido ahora en el mundo de la restauración que ha terminado por aparecer una eterna (aunque innecesaria) discusión. Y esta ha llegado a un punto de no retorno. Resulta que uno de sus principales colegas del gremio, Moe y el propio Freddy Krueger están en discusión abierta porque el primero dice que la tortilla sin cebolla (y tiene razón), sin embargo Freddy opina que o con cebolla o es una pesadilla. Han decidido solucionar el tema cocinando un montón de tortillas y que un grupo de jueces (uno rojo, uno morado, uno negro, uno naranja, uno amarillo y uno negro; aunque uno es un impostor) decida cual es mejor.

Moe lo tiene fácil, es amigo del Sr. Cangrejo y cocinará las tortillas sin cebolla en dos patadas en el Crustáceo Crujiente. Pero Freddy tiene un plan que a nadie se le ocurriría ni en sueños (putupumpush). Creará un autómata que cocine tortillas con cebolla (si se les puede llamar tortillas) para él. Pero antes del proyecto final prefiere hacer una simulación, por lo que se ha colado en los sueños de los avezados alumnos del Colegio vivas, doctos ellos y de indudable pundonor, para convencerles de que les haga una simulación de dicho sistema, o si no, puede pasar algo... vamos, oferta que no se puede rechazar viniendo de quién viene... Así en los sueños se os coló la siguiente especificación ...

COLEXIO VIVAS S.L.	RAMA:	Informática	CICLO:	DAM		
	MÓDULO	Servicios				CURSO: 2º
	PROTOCOLO:	Examen	AVAL:	1ª	DATA:	27/11/2023
	Nombre y Apellidos:					

Especificación: Se desea realizar un simulador de autómata-fabrica-tortillas. Para ello hay que cocinar dos ingredientes: por un lado cebolla y por otro patatas. Luego, cuando hay cantidad suficiente de cada uno, se puede hacer una tortilla. El objetivo es que se hagan un total de 10 tortillas.

Para esta simulación crea una clase denominada `OmeletteAutomator` que tendrá, como mínimo, las siguientes variables privadas:

NOions: Entero que se inicializa a 0 y en todo momento mantendrá un conteo de las cebollas cocinadas.

NPotatoes: Entero que se inicializa a 0 y en todo momento mantendrá un conteo de las patatas cocinadas.

NOmelettes: Entero que se inicializa a 0 y en todo momento mantendrá un conteo de las tortillas realizadas.

Por supuesto se deben añadir otras variables como booleanas, testigos, etc. según sea necesario.

También se harán las siguientes funciones que luego se lanzarán en hilos.

(2.5p) Omelette: Función que prepara tortillas de forma continua (termina al llegar a 10 tortillas). En cada preparación lo primero que tiene es que comprobar que al menos hay 5 cebollas y 5 patatas cocinadas (en *NOions* y *NPotatoes* respectivamente), si no es así escribe en pantalla *Waiting...* y entra en espera.

Cuando sale de la espera decrementa en 5 tanto las cebollas como las patatas, incrementa las tortillas en una unidad y muestra en pantalla con un único `Write/WriteLine`, en una línea, la cantidad de tortillas, de cebollas y de patatas que hay. Formatea la cadena para que cada cifra ocupe siempre 3 posiciones. Un ejemplo sería:

Omelettes: 6 Onions: 9 Potatoes: 4

Debe quedarse cocinando tortillas mientras haya al menos 5 ingredientes de cada.

Una vez fabricadas las 10 tortillas ya no se puede cocinar nada más: ni cebollas ni patatas.

(2.5p) Ingredient: Función que cocina un ingrediente determinado de forma continua. Tendrá dos parámetros, el primero será un *string* con el nombre del ingrediente (que luego será *Onion* o *Potato*), el segundo el contador del ingrediente pasado por **referencia** (Es decir *NOions* o *NPotatoes*).

Esta función, de forma continua y mientras no se alcance el límite de fabricación de tortillas, hará lo siguiente: Incrementa el ingrediente que se le pasa por parámetro en una unidad y luego muestra por pantalla el nombre (ocupando 7 caracteres) y la cantidad de ingredientes

COLEXIO VIVAS S.L.	RAMA:	Informática	CICLO:	DAM		
	MÓDULO	Servicios				CURSO: 2º
	PROTOCOLO:	Examen	AVAL:	1ª	DATA:	27/11/2023
	Nombre y Apellidos:					

(ocupando 3 caracteres). Además si la cantidad de ambos ingredientes es al menos de 5 avisará a la función *Omelette* que continúe (por si está esperando).

Nota: Si *Ingredient* no sabes hacerlo en una función con parámetro por referencia, puedes usar un único parámetro delegado (crealo tú o usa el delegado *Action*). Cuando llames a *Ingredient* le pasarás una lambda que se encarga del incremento y del mensaje. Esta variante **no penaliza la puntuación**.

Si no sabes hacerlo de ninguna de las dos formas puedes hacer una función única con parámetro el nombre, dependiendo del nombre incrementará *Nonions* o *Npotatoes*. Puntuará menos.

Recuerda que esta función finaliza cuando se acaban de cocinar las tortillas.

No debe usarse ningún *SetCursorPosition* a la hora de mostrar datos en pantalla, todo con *WriteLines*. **Tampoco habrá *Sleeps*** en estas dos funciones.

(1p) Habrá una función pública denominada *Init* la cual ejecuta tres hilos:

tOnion lanza como hilo la función *Ingredient* con parámetros "Onion" y *NOnion*. Hazlo con lambda para poder pasar parámetros.

tPotato lanza como hilo la función *Ingredient* con parámetros "Potato" y *NPotatoes*. Hazlo con lambda para poder pasar parámetros.

tOmelette lanza como hilo la función *Omelette* **y se queda a la espera** de que esta acabe mediante *Join*.

Cuando acabe todo, también en el *Init*, escribe en pantalla: *The End.Moe Wins. No onion rules!*


El programa principal simplemente crea un objeto de la clase *OmeletteAutomator* y ejecuta su función *Init()*.

Ejercicio 2

Realiza un nuevo proyecto dentro de la misma solución con las siguientes especificaciones:

(1p) Escribe un método denominado ***RandomInfo***, sin parámetros y que devuelve un string. Cual cogerá un proceso aleatorio de todos los que se están ejecutando en el ordenador. Crea un string con la siguiente información del proceso aleatorio seleccionado: el **nombre** del mismo y **lista de DLLs** que usa. Cada dato en una fila distinta. Devuelve dicho string.

No hagas el control de excepciones en esta función.

	RAMA:	Informática	CICLO:	DAM			
	MÓDULO	Servicios				CURSO:	2º
	PROTOCOLO:	Examen	AVAL:	1ª	DATA:	27/11/2023	
	Nombre y Apellidos:						

(0.2p) Pon el proyecto en modo 64 bits y pon las credenciales como Administrador.

(1p) Escribe otro método denominado `WindowsFiles1000` sin parámetros y que devuelve string.

Utilizando la clase `Array.FindAll` y lambda crea un vector de `FileInfo` que contenga todos los archivos del directorio `%windir%` cuya longitud sea de 1000 o menos bytes. Luego mediante un `Array.ForEach` que recorra el vector previo debes crear un string con los nombres de los archivos (sin la ruta) y lo que ocupan. Cada uno en una línea del string. Devolverá dicho string.

Nota: Si este apartado no sabes hacerlo con archivos, hazlo con nombres de procesos cuyo pid sea menor que 1000 para poder demostrar el uso de Array y las lambdas. Puntuará menos.

Si no sabes manejar los métodos de la clase Array hazlo con bucles for o foreach. Puntuará menos.

(0.8p) Define un tipo delegado denominado ***ReturnStringDelegate*** que defina funciones sin parámetros y que devuelve un string.

Realiza un método denominado `ExceptionControl` al que se le pase un delegado tipo `t` y haga comprobación try catch genérica de la función del delegado que entra como parámetro. Si entra en el catch devuelve el mensaje "ERROR" si no hay problema devuelve la cadena del propio delegado.

(1p) En el programa principal haz las siguientes tareas:

- Llama a `ExceptionControl` al cual se le pasará la función `RandomInfo` como parámetro para que sea ejecutada con control de excepciones.

La cadena devuelta se guardará **sobreescribiendo** un archivo de texto situado en `%USERPROFILE%` y de nombre `servicios2023.txt`. El archivo debe cerrarse tras esto.

- A continuación llama nuevamente a `ExceptionControl` pero pasándole como parámetro la función `WindowsFiles1000`.

El string devuelto lo **añadirá** al archivo `servicios2023.txt` citado anteriormente.

- Finalmente lee y muestra en pantalla el contenido del archivo `servicios2023.txt`.

No es necesario el control de excepciones en el acceso a archivos.