Examen teoría TPP junio de 2022

Debajo reuní el mayor número de preguntas del examen de teoria de TPP del curso 2021-2022. El examen fue más o menos 50% preguntas de teoría y otor 50% entender código. Fueron unas 14 o 15 preguntas, y la puntuación de cada una era 0.5, 0.75 o alguna de 1 punto.

1. Ordena estos **lenguajes** de más a menos abstracto: DSL(especifico de dominio),ensamblado,C, C#
2. Indica si son **paradigma** declarativos o imperativos: Orientada a objetos, funcional, estructurado basado en procedimientos, lógico
3. Ventajas de la **POO**, es decir, programación orientada a objetos.
4. Problemas de la **herencia múltiple**
5. Definir **metaprogramación** (o generación dinámica de código)
6. Te dan código con un problema de **Lock**, y te ofrecen cuatro opciones para bloquear: Lock(¿????) El problema era más o menos : “Tienes una clase comun en la que hay un FileLog, y escribes en ese mismo fichero, a partir de diferentes hilos. Evidentemente el FileLog es un recurso compartido y puede llegar a dar problemas. ¿Cómo arreglas este codigo? Indica para las cuatro opciones que te damos, el porqué sí o porqué no son buenas o malas ideas. Deja claro cual eliges”.

Las diferentes opciones eran algo así:

* 1. This
  2. Objeto anonimo bloqueador, unico para cada clase (es decir, teniamos la clase comun en la que hay un fichero compartido, y otra clase para cada Worker, me parece. Pues este objeto se crea nuevo en cada una de estas)
  3. Console
  4. FileLogger (que es el mismo para todos porque es donde escribe el fichero)

1. Con estos dos métodos, indica si devuelve true o false:

Ejercicio para comprobar el tipado dinámico

**CheckDynamic**(a):

CheckDynamic(b):

CheckDynamic(c):

CheckDynamic(d):

1. Indica a qué tipos **inferencia** el compilador:

Te daba dos métodos y tenias que interpretar qué pensará el compilador para esas variables y siguiendo el codigo que daba

Var Q1:

Var Q2:

Var Q3:

1. Invocar dos veces a una funcion lambda, aplicando el uso de ***true*** y ***false***. Similar a los de seminario. Muy importante no saltarse ningún paso, mejor escribir mucho.
2. Escribir código en c# o pseudocodigo del método **reducir** hecho en los laboratorios. Después escribir una invoación para una lista de numero {1,2,3,4} obtener su producto. (Aquí creo que aunque no te lo especificaba, tenias que ponerle la famosa “semilla”. Porque si empiezas con el acumulador a cero, e intentas hacer 0 por 1, siempre vas a arrastar el cero. Si hubieran pedido suma, y no el producto, no habría este problema)
3. Una pregunta sobre si es **referencialmente transparente**. Decir qué es, y justificar ese caso. Nota: aparecia una variable aux que no se usaba para nada en el retorno, pero su valor se calcula en cada llamada con un random
4. Otro ejercicio sobre qué imprime por consola: si lanza la excepción o muesra el mensaje del código (parecido a los que están en los examenes del 2020, basicamente es para saber cómo funcionan las excepciones)
5. Pregunta si se capturará o no la excepción dentro de ese código con los hilos.
6. Dos metodos: metodo1() y metodo2(). Si hay tres bucles for anidados en cada, pero en metodo1() se hace ParallelFor al for más externo, y en método2 al más interno. ¿Cuál es la mejor opción?
7. Ordena estos **lenguajes** de más a menos abstracto: DSL(especifico de dominio),ensamblado,C, C#
8. Indica si son **paradigma** declarativos o imperativos: Orientada a objetos, funcional, estructurado basado en procedimientos, lógico
9. Ventajas de la **POO**, es decir, programación orientada a objetos.

Robustez, encapsulamiento, asbtracciones de objetos que existen en la realidad

1. Problemas de la **herencia múltiple**

Coincidencia de nombres y herencia repetida (atributos repetidos)

1. Definir **metaprogramación** (o generación dinámica de código)
2. Te dan código con un problema de **Lock**, y te ofrecen cuatro opciones para bloquear: Lock(¿????) El problema era más o menos : “Tienes una clase comun en la que hay un FileLog, y escribes en ese mismo fichero, a partir de diferentes hilos. Evidentemente el FileLog es un recurso compartido y puede llegar a dar problemas. ¿Cómo arreglas este codigo? Indica para las cuatro opciones que te damos, el porqué sí o porqué no son buenas o malas ideas. Deja claro cual eliges”.

Las diferentes opciones eran algo así:

* 1. This No sirve
  2. Objeto anonimo bloqueador, unico para cada clase (es decir, teniamos la clase comun en la que hay un fichero compartido, y otra clase para cada Worker, me parece. Pues este objeto se crea nuevo en cada una de estas) Tampoco sirve
  3. Console.Out También sirve como método de sincronización, porque aunque en este contexto no tiene importancia, es un recurso que comparten todos los hilos y apunta al mismo objeto
  4. FileLogger (que es el mismo para todos porque es donde escribe el fichero) La mejor solución

1. Con estos dos métodos, indica si devuelve true o false:

CheckDynamic(a): False. Age lo guardó como String pero intenta leerlo como int

CheckDynamic(b): False. No tiene Width, da excepción en tiempo de ejecución, se captura y devuelve falso.

CheckDynamic(c): True. Todo normal

CheckDynamic(d): True

1. Indica a qué tipos **inferencia** el compilador:

Nota sobre función Repeat(), hay una tabla(que está unas preguntas más abajo) en clase que tiene muchos métodos utiles que dio en teoría, que podían caer en el examen, como fue este caso.

1. Invocar dos veces a una funcion lambda, aplicando el uso de ***true*** y ***false***. Similar a los de seminario.
2. Escribir código en c# o pseudocodigo del método **reducir** hecho en los laboratorios. Después escribir una invoación para una lista de numero {1,2,3,4} obtener su producto. (Aquí creo que aunque no te lo especificaba, tenias que ponerle la famosa “semilla”. Porque si empiezas con el acumulador a cero, e intentas hacer 0 por 1, siempre vas a arrastar el cero. Si hubieran pedido suma, y no el producto, no habría este problema)
3. Una pregunta sobre si es **referencialmente transparente**.

Ojo, puede haber métodos que tengan asignaciones destructivas (+=, -=,…) y tener igualmente referencia transparencial.

El ejercicio estaba usando una función impura, porque creaba un numero aleatorio con random, como no afecta al resultado sigue siendo referencialmente transparente.

1. Ejercicio en el que pregunta qué imprimirá por consola.

Este ejercicio tenía truco. No devuelve nada porque son **métodos perezosos** , no originan nada por consola porque solo trabajan cuando se les pide. Había que darse cuenta al ver los métodos zip, o take, que son perezosos. No imprime nada. Hasta que no lo utilizas no sacaría nada por pantalla este método.

Tabla

Descripción generada automáticamente

1. 3. Pregunta si se capturará o no la excepció dentro de ese código con los hilos.

El programa bruscamente. Porque el programa lanza la excepción en el hilo, no en el hilo principal, y como el try-cath no está manejado dentro del hilo, no se maneja la excepción y acaba bruscamente. Si en lugar de Thread la excepción la crease una Task, entonces sí la captaría igualmente el hilo principal como *AggregateException.*

1. -
2. Dos metodos: metodo1() y metodo2(). Si hay tres bucles for anidados en cada, pero en metodo1() se hace ParallelFor al for más externo, y en método2 al más interno. ¿Cuál es la mejor opción?

Paralelizar el bucle más externo es la mejor opción, “pocos hilos que ejecutan bastantes cosas”. Metodo2 hace muy pocas operaciones y crea un montón de hilos. Al crear muchos hilos, compiten por la CPU y el rendimiento es peor.