Material complementario 17/03/2020

Arno Formella

Este documento recoge más o menos lo que hubiese contado en clases presenciales a lo largo de la exposición paulatina de las transparencias. En clase normalmente descubro diferentes partes del contenido de cada transparencia poco a poco, para no sobrecargar con información de golpe, y para mantener un hilo de pensamiento. En las transparencias distribuidas todo viene de golpe, por eso recomiendo trabajar con ellas lentamente y ver este material complementario a la par.

Estarán incluidas preguntas (a veces sin respuestas) para animar a la reflexión sobre el tema y a la búsqueda de información adicional. Admás proporciono más referencias a la red y la bibliografía.

Ojo, las páginas mencionadas aquí siempre se refieren a los tochos que publico para cada semana. Puede ser (y es casi seguro) que en el documento global (que desarrollo en paralelo) la enumeración de las página va a variar, ya que se añaden transparencias en otros lugares.

P128

Es una repetición de la semana pasada.

La clase permite la medición individual de tiempos transcurridos en los diferentes hilos. Sigue un modelo parecido a los çhips"(botones) que usan en carreras (por ejemplo San Martiño, `https://www.carreraspopulares.com/noticia/como-funciona-el-chip-de-una-carrera`).

El constructor coge como argumento el número de hilos participantes.

`Start` marca el punto de salida. `Stop` marca el punto de llegada.

`Elapsed` devuelve en milisegundos el intervalo entre hora de salida mínima y hora de llegada máxima. Con eso se conoce el intervalo de tiempo durante el cual por lo menos un hilo estaba activo.

Nota: con esta medición no se mide el tiempo que un hilo pueda "gastar" antes de la llamada a `Start` ni después de la llamada a `Stop` (será el tiempo entre el correspondiente `start` del hilo que lanza y del correspondiente `join` de sincronización). ¿Tienes una idea como medir estos tiempos, o por lo menos su suma?

Observa: Java no especifica nada sobre lo que pasa entre las llamadas `start` y `run`. Normalmente los hilos (`Thread`) están implementados con los procesos del sistema operativo. El manual de Java solo dice que después del `start` el hilo correspondiente entre en su `run`, pero no dice nada *cuando* eso ocurra, en principio podría ser el año que viene! (`https://docs.oracle.com/javase/9/docs/api/java/lang/Thread.html#start--`)

Podéis modificar la clase para que funcione con un número dinámico de hilos (es decir, no se sabe en el momento de construcción cuantos hilos van a participar).

P129

Eso ya vistéis en prácticas.

Acostúmbrate pasar argumentos via línea de comando (o ficheros de entrada) para que puedas ejecutar tus programas facilmente con *scripts* sin interacción de un usuario.

Puedes considerar la clase `NumberFormat` para obtener otros tipos de datos (por ejemplo `Int`, `Long`, `double` a partir de las cadenas de argumentos. Mira, por ejemplo, `https://docs.oracle.com/javase/9/docs/api/java/text/NumberFormat.html#parse-java.lang.String-java.text.ParsePosition-`

P130

Mira comparación entre diferentes generadores de documentación (`https://en.wikipedia.org/wiki/Comparison_of_documentation_generators`).

`doxygen` no está mal respecto a los lenguajes de programación sobre culaes puede actuar, sobre características adicionales (p.ej. grafos de clases y funciones) y formatos de salida. Además es multiplataforma y código abierto.

P131

Ten en cuenta que durante la ejecuión de un programa los hilos no se ejecutan necesariamente en paralelo. Imagínate que haya un solo procesador sobre el cual se están conmutando los procesos/hilos, de hecho, en este caso hay solamente un hilo activo en cada instante.

P132

Ya se ha visto en prácticas como generar hilos con las herramientas disponibles en Java.

Sea consciente de la diferencia entre `start` y `run`.

P133

Ya que en Java solamente existe la implementación múltiple de interfaces y no la extensión múltiple de clases, familiarízate con el uso de `Thread` y `Runnable`.

P134

Se ves o usas en un programa uno de estas *cosas* de Java, tienes una buena indicación que se trata de un programa concurrente.

Para escribir programas concurrente de forma correcta (y eficiente) debes tener clara la semántica de todos ellos (y algunos más.

Trabaja con el tutorial sobre programación currente de Oracle cuyo enlace está en la primera práctica.

P135

Las excepciones es un concepto embebido en muchos lenguajes de programación que permiten interrumpir el flujo de control a causa de un evento ocurrido (p.ej. un fallo específico) y seguir con la ejecución en otro punto del programa (ya que no hace sentido seguir en el punto donde se produje la causa de la excepción).

P136

Ten cuidado con lanzar la misma excepción en diferentes puntos del bloque `try`, ya que en `catch` correspondiente no se sabe antemano en cual de los posibles puntos se generó la excepción.

P137

Acostúmbrate realizar los bloques `try` y `finally` incluso si no has pensado incluir un bloque `catch` en tal momento, ya que de está manera todo lo que se debe ejecutar al final ya está agrupado en el bloque `finally` y modificaciones futuras del código y su entendimiento son más fáciles.

P138+P139

No es mala idea de usar siempre una capa fina de propias clases en una aplicación. De esta forma se detecta y se reacciona mejor a posibles excepciones que genera el programa.

Hay que estar muy familiarizado con las formas como un lenguaje en concreto implementa el uso de excepciones.

Una excepción en un proceso/hilo puede provocar una cascada de excepciones en los demás participantes, cuyo orden temporal es difícil de anticipar.

P140

En el último caso las excepciones se tratan en el sistema operativo.

P141

Como cabe esperar, lanzar excepciones en la construcción de objetos no es muy aconsejable. Pero hay que tener en cuenta que siempre pueda ocurrir, p.ej. si ya no queda memoria para tal objeto, y se lanza la excepción *out-of-memory*.

P142

Existen en Jave una multitud de excepciones que pueden ocurrir durante la ejecución de un programa, mira p.ej. `https://docs.oracle.com/javase/9/docs/api/java/lang/RuntimeException.html`

P143

Aunque hay desarrollo de software que se base bastante en el uso de excepciones, recomiendo reducir tal uso en programas concurrentes al mínimo posible.

Imagínate si un hilo lanza una excepción en un momento determinado, ¿qué deben hacer los demás hilos a partir de esto momento, y clomo se enteren que se haya producido tal caso excepcional?

Recomiendo familiarizarse con el patrón de diseño mencionado.