CSD Prueba aula 1 Temas 1 y 2 Feb 2018 **000** **- Solución**

Indica si cada una de las siguientes afirmaciones es verdadera (V) o falsa (F). En cada cuestión las afirmaciones comparten un mismo enunciado, pero son independientes entre sí (el número de afirmaciones ciertas en una cuestión puede variar de 0 a 4). Puntuación: (aciertos-errores) escalado a 10

Ventajas e inconvenientes de la programación concurrente:

V F Es más eficiente que la programación secuencial, pues permite aprovechar mejor los recursos máquina

V F Permite en una misma aplicación múltiples actividades simultáneas, ofreciendo mejor servicio al usuario

V F La depuración es más sencilla que en la programación secuencial

V F La ejecución paralela de múltiples actividades puede plantear problemas en el acceso a algunos recursos

Utilización de hilos en Java:

V F En Java sólo puede crear hilos el hilo inicial (main)

V F El código a ejecutar por cada hilo debe estar contenido en su método start()

V F Para asignar nombre a los hilos se puede pasar una cadena como argumento en su constructor

V F Los objetos definidos en Java se mantienen en el 'heap' y son compartidos por los hilos que tengan una referencia a ellos

Sobre el estado de planificación de los hilos en Java:

V F Cuando un hilo utiliza el método Thread.join() pasa a suspendido hasta que finalice la ejecución de todos sus hijos

V F Thread.yield() hace que un hilo pase del estado 'preparado' al estado 'suspendido'

V F La única forma en que un hilo suspendido con Thread.sleep() vuelva a preparado es cuando finaliza el período de espera (milisegundos indicados como argumento)

V F Un hilo Java no puede pasar directamente del estado 'en ejecución' al estado 'preparado'

Creación y uso de hilos en Java

V F La aplicación finaliza al terminar el hilo inicial

V F Un hilo no puede terminar antes que otro hilo creado por él (espera implícitamente a la finalización del hijo antes de terminar)

V F La estructura habitual es que un hilo crea primero los objetos a compartir y luego los hilos que deben compartirlos, pasando dichos objetos compartidos como argumentos de los constructores de los hilos

V F Cuando varios hilos comparten un objeto, cada uno de esos hilos debe invocar un método distinto de dicho objeto (distintos hilos no pueden invocar el mismo método)

Suponemos que el código completo de todos los hilos de un determinado programa esta protegido por un mismo 'lock' (cada hilo lo cierra al empezar y lo abre justo antes de terminar)

V F Cumple trivialmente la propiedad de exclusión mutua (no es necesario reforzarlo en el código)

V F Fuerza a que los hilos siguieran una ejecución secuencial

V F Obtenemos paralelismo real

V F No hay condiciones de carrera

En cuanto al mecanismo para garantizar exclusión mútua:

V F Cuando aplicamos la etiqueta synchronized en un método de java, genera automáticamente los protocolos de entrada y salida necesarios para convertir el cuerpo de dicho método en una Sección Crítica

V F Es indispensable para compartir objetos mutables de forma segura

V F Sólo es necesario aplicarlo en los métodos que modifican el estado del objeto

V F Hay un lock distinto para cada método protegido por synchronized