Tarea Java #9: Paso de Mensajes a un Hilo

* Nombre de Archivo Java: ***Hilos.java***

* Ejemplo Nombre Archivo Comprimido:

GutierrezAvilesLuisJ9TSOA0413B.zip

Objetivo

Construir una aplicación basada en hilos que contemple la ejecución de un hilo evento y su interacción con un hilo Thread.

Descripción

Implementar una aplicación con 2 hilos de control, uno de los cuales se pueda solicitársele suspender y reanudar su ejecución mediante la ejecución de un evento botón.

Requerimientos Generales

1. Ambos hilos impriman un contador en áreas de texto particulares
2. Un hilo imprima cada segundo y el otro cada 2 segundos
3. Incluir un botón para asociarlo en tiempo compilación a uno cualquiera de los hilos
4. Al presionar el botón, que el hilo correspondiente suspenda su ejecución
5. El hilo reanude su ejecución tras otro botonazo con el mismo botón que el punto anterior
6. Tomar en cuenta que el método Thread.suspend() está depreciado, por lo cual no deberá usarse, y en su lugar la solicitud de suspensión del hilo conlleve a suspender los efectos del hilo sobre su correspondiente área de texto
7. El hilo carezca de atributos públicos
8. El evento del botonazo, provoque el paso de solo un mensaje al objeto hilo, indicándole solo que ha ocurrido un botonazo, por lo que el método a invocar no deberá recibir argumento alguno
9. Para lograr el efecto de la suspensión del hilo se requiera de una bandera y además se opte por una de las siguientes alternativas:
   a) Si la bandera indica que el hilo imprima, entonces se imprime e incrementa el contador
   b) Si la bandera indica suspender, entonces mediante wait() y una variable de instancia asociada ejecutar wait() para evitar momentáneamente (en lo que se presiona de nuevo el botón) que imprima el incremente el contador; cuando se vuelva a presionar el botón se ejecute notify() sobre la variable asociada

PUNTOS EXTRAS (50%)

- **Tendrán puntos extras si lo implementan utilizando wait() y notify().**

Requerimientos y Restricciones Complementarios

A. Todos los identificadores de variables y constantes tengan nombres claros y significativos de modo que sea fácil saber qué dato almacenarán
B. Todo identificador debe ser necesario para la operación del programa
C. Cumplir con las convenciones para nomenclatura de variables y constantes explicados en clase según convenciones Java
D. Todo valor asignado a un identificador debe ser necesario para la operación de alguna parte del programa
E. A excepción de las constantes 0 y 1, los cómputos utilicen únicamente variables y constantes simbólicas
F. A excepción de que un requerimiento lo solicite, no incluir la redacción de los requerimientos en el código fuente
G. Las líneas de código fuente sean máximo de 100 caracteres
H. Código fuente indicando como comentario en la parte superior de cada archivo: nombre del(la) alumn@, sección y no. de tarea
I. Cumplir con las convenciones sobre indentación de código fuente
J. Código fuente libre de instrucciones anuladas mediante comentarios
K. El código fuente libre de errores de compilación y advertencias
L. Código fuente libre de comentarios o impresiones de pantalla que describan el funcionamiento del código
M. Evitar comparar banderas contra constante numérica o simbólica
N. Toda instrucción ejecutada sea necesaria según los datos de entrada; por ejemplo, en un programa que validara una fecha en base a día, mes y año, si se trata del mes de Enero, sería inapropiado comprobar si se trata de un año bisiesto, pues esto solo sería necesario para caso del mes de Febrero.
O. Evitar el uso del operador relacional == de una misma variable contra 2 ó más tipos de datos enteros siempre que en su lugar se pueda utilizar la estructura de control “switch”
P. Programa en ejecución libre de desbordamiento de arreglos
Q. Todos los métodos tengan nombres claros y significativos de modo que sea fácil saber cuál es su función en el programa.
R. Ejercer la programación modular siempre que sea posible la reutilización de código

S. Evitar declarar atributos cuyos datos no formen parte de las propiedades que describan la clase en la que se declaren o bien no representen el estado del programa, y que por tanto sea posible una implementación utilizando variables locales y paso de parámetros.
T. Todas las clases (y sus atributos) cuenten con un nombre (identificador) claro y significativo a modo que a la lectura sea intuitivo saber para qué se utiliza.
U. Toda instrucción del programa en ejecución sea ejecutable sin necesidad de recompilar el programa
V. Cumplir con las convenciones Java (para clases, métodos, etc.).
W. Todo atributo siempre sea privado, a menos que exista una implementación que por eficiencia justifique un menor ocultamiento de información, pero que igualmente el diseño cumpla con la propiedad de encapsulamiento.
X. Si incluye comentarios, estos sean solo para Javadoc; en sus métodos, estos sean breves y referentes solo a los parámetros y resultados a obtener, y de las clases solo su propósito; los anteriores en términos de caja negra.
Y. El Código fuente sea libre de instrucciones para suprimir advertencias
Z. Evitar el uso del modificador static, a menos que sea indispensable el uso de atributos de clase y métodos de clase, justificando su uso con comentarios en código

Criterios de Evaluación

- Cumplir con las “Reglas de Operación y Evaluación del Taller de Sistemas Operativos Avanzados”.
- Fecha de asignación: 11 de Septiembre del 2013
- Fecha planeación de entrega: 18 de Septiembre del 2013
- Fecha límite de entrega (Sólo 55% del total): 09 de Octubre del 2013
- Observación: _Ninguna_____
- Calificación en base a cobertura de requerimientos y fecha de entrega en horas clase