Lab. 15.1 : Creating a Thread

Objetivo: Familiarizarse con la creación de hilos por medio de la implementación de la interface **Runnable**

1. **Escriba una clase con el nombre PrintsNumbers que implemente la interface Runnable. Agregue un campo de tipo boolean keepGoing y un constructor inicializa a keepGoing con true.**

2. **Agregue un método a la clase PrintNumbers llamada stopPrinting() que asigne false al campo keepGoing.**

3. **Dentro del método run(), escriba un bucle while usando System.out.println(), el cual despliega los números 1,2,3,4 … mientras el campo keepGoing sea true. Entre los despliegue de cada numero, el hilo debe dormir por un segundo.**

4. **Salve y compile la clase PrintNumbers.**

5. **Escriba una clase con el nombre Print que contenga el método main(). Dentro del método main(), instancie un objeto PrintNumbers. Instancie un objeto Thread el cual será utilizado para correr el objeto PrintNumbers en un hilo separado y luego inicia el hilo.**

6. **El programa Print tomará un sencillo argumento en la línea de comando para representar el numero en milisegundos que el hilo main() dormirá. Convierta este argumento a un valor entero y luego haga que el hilo main() duerma durante esa cantidad de milisegundos.**

7. **Cuando el hilo thread se despierta, haga que este invoque al método stopPrinting() en el objeto PrintNumbers. Luego Despliegue "main() is ending… "**

8. **Salve, compile y corra el programa Print. No olvide pasarle un argumento entero para representar cuanto debe correr el programa en milisegundos.**

Los números 1, 2, 3, … serán desplegados por aproximadamente el numero de segundos que sea especificado con el argumento de la línea de comando. Por ejemplo, si el argumento en la línea de comando es 10,000 UD. deberá ver cerca de 10 números desplegados. Este lab. Demuestra la necesidad común en la programación de hilo: Crear un mecanismo para que un hilo creado detenga su ejecución. El método stopPrinting() puede ser invocado por cualquiera desde otro hilo para informar PrintNumbers que pude terminar de correr.

Lab. 15.2: Simulating a Car Race

Objetivo: Familiarizarse con la codificación de un hilo extendiendo la clase Thread.

1. **Escriba una clase con el nombre RaceCar que extiende a la clase Thread y que contenga el método run().**

2. **Agregue un campo private int con el nombre finish y un campo private String con el nombre name. Agregue un constructor que inicialice ambos campos.**

3. **Dentro de run(), agregue un bucle for que ejecute un numero de veces de finish. Dentro del bucle for, use System.out.println() para desplegar el campo name y la iteración actual del bucle. Por ejemplo, la tercera vez que pasa el tercer bucle debe desplegar "Mario: 3" para el carro llamado Mario. Entonces, hacer que el hilo duerma durante un número aleatorio de veces entre 0 y 5 segundos; en cada i-ésima iteración**

4. **Al final del bucle for, desplegar el mensaje que establece que la carrera de carros ha terminado y desplegar el campo name. Por ejemplo "Mario finished!"**

5. **Salve y compile la clase RaceCar.**

6. **Escriba una clase con el nombre Race que contenga el main().**

7. **Dentro del main(), declare y cree un arreglo con el nombre cars lo suficientemente grande para almacenar cinco referencias Thread.**

8. **Escriba un bucle for que rellene el arreglo con cinco objetos RaceCar. Los nombres se obtendrán de los cinco argumentos de la línea de comando, y el int debe ser el mismo para cada RaceCar. Este valor representará que tan larga será la carrera y este debe ser ingresado desde la línea de comando.**

9. **Escriba un segundo bucle for que invoque a start() en cada Thread en el arreglo.**

10. **Salve, compile y corra el programa Race.**

El programa Race se verá como una carrera de carros que progresa despacio cuando UD. la ve. El ganador será el hilo RaceCar que termina primero.

Lab. 15.3: Usando Timer and TimerTask

Objetivo: Familiarizarse con el uso de las clases Timer y TimerTask

1. **Escriba una clase con el nombre Remainder que extienda a TimerTask. Agregue un campo de tipo String con el nombre message y un constructor que inicialice este campo.**

2. **Dentro del método run() simplemente despliegue el campo message usando System.out.println().**

3. **Escriba una clase llamada TestReminder que contenga a main().**

4. **Dentro del main(), instancie un nuevo objeto Timer.**

5. **Dentro del main(), instancie tres objetos Reminder, cada uno con diferente message.**

6. **Usando el método schedule() de la clase Timer que crea una tarea simple, schedule los tres objetos Reminder con el Timer. Tome inmediatamente el primer Reminder scheduled, tome el segundo reminder después de 30 segundos y el tercer reminder después de dos minutos.**

7. **Salve, compile y corra el programa TestReminder.**

Los tres reminders deben desplegarse en la línea de comando. UD. debe esperar 2 minutos antes de ver el reminder final.

Lab. 15.4: An Applet Game

Este laboratorio junta muchos de los aspectos de Java que UD. ha aprendido hasta ahora.

Objetivo: Escribir un applet que es un juego que prueba las destrezas de un usuario y la rapidez con el ratón. Escribir un juego que despliegue una imagen moviéndose en la pantalla. El jugador de este juego gana puntos haciendo clic en la imagen.

He aquí lo que se espera del juego.

1. **El juego debe ser un applet de tal manera que este embebido en una pagina Web.**
2. **Use Japplet para su applet y los componentes Swing para cualquier componente GUI que use.**
3. **Para crear una imagen que se mueva por la pantalla, UD. puede crear una imagen bitmap usando un programa como Microsoft Paint. Alternativamente, puede usar uno de los métodos de la clase java.awt.Graphics. Verifique la documentación de la clase Graphics y navegue por los métodos. Por ejemplo el método fillOval() puede ser usado para dibujar un circulo, o el método fillRect() dibuja un rectángulo. (Hint: Un rectángulo puede simplificar la matemática considerablemente cuando UD. trata de determinar si el usuario ha hecho clic sobre la imagen.)**
4. **Escriba un hilo que contenga una referencia al recipiente de contenido de JApplet. El hilo debe dibujar la imagen en la pantalla, dormir por una cantidad especificada de tiempo y luego redibujar la imagen en algún otro lugar en la pantalla. UD. puede desplegar la imagen en diferentes tamaños para hacer el método más desafiante.**
5. **Provea un JTextField que despliegue el tiempo que duerme el hilo. El usuario debe ser capaz de cambiar este valor dependiendo de que rápido el piense que sea. Agregue el correspondiente manejo del evento que cambia el tiempo de dormir en la clase thread.**
6. **Agregue un JLabel que despliegue los puntos. UD. puede escoger la puntuación de la manera que quiera. Un simple método de puntuación puede ser 10 puntos si se ha alcanzado el objeto rápidamente o cuando se alcanza el objeto cuando este e mas pequeño.**
7. **UD. necesitará un MouseListener que maneje un evento mouseClicked(). Esta clase determinará cuando el objeto fue alcanzado, y si es así, con cuantos puntos se premia esto.**
8. **Escriba una pagina HTML que empotre su JApplet. Este programa probablemente requiere muchas pruebas, así que el appletviewer puede ser manejado.**

UD. debe ser capas de empotrar su applet en la página Web y jugar el juego.