A veces nos interesa que un hilo se quede bloqueado a la espera de que ocurra algún evento, como la llegada de un dato para tratar o que el usuario termine de escribir algo en una interface de usuario. Todos los objetos java tienen el método **wait()** que deja bloqueado al hilo que lo llama y el método **notify()**, que desbloquea a los hilos bloqueados por **wait()**.

**Bloquear un hilo**

Antes de nada, que quede claro que las llamadas a **wait()** lanzan excepciones que hay que capturar. Todas las llamadas que pongamos aquí deberían estar en un bloque **try-catch**, así

try  
{  
   // llamada a wait()  
}  
catch (Exception e)  
{  
   ....   
}

Para que un hilo se bloquee basta con que llame al método **wait()** de cualquier objeto. Sin embargo, es necesario que dicho hilo haya marcado ese objeto como **ocupado** por medio de un **synchronized**. Si no se hace así, saltará una excepción de que "**el hilo no es propietario del monitor**" o algo así.

Imaginemos que nuestro hilo quiere retirar datos de una **lista** y si no hay datos, quiere esperar a que los haya. El hilo puede hacer algo como esto

synchronized(lista);  
{  
   if (lista.size()==0)  
      lista.wait();  
     
   dato = lista.get(0);  
   lista.remove(0);   
}

En primer lugar hicimos el **synchronized(lista)** para "apropiarnos" del objeto lista. Luego, si no hay datos, hacemos el **lista.wait()**. Una vez que nos metemos en el **wait()**, el objeto lista queda marcado como "desocupado", de forma que otros hilos pueden usarlo. Cuando despertemos y salgamos del **wait()**, volverá a marcarse como "ocupado."

Nuestro hilo se desbloqueará y saldrá del **wait()** cuando alguien llame a **lista.notify()**. Si el hilo que mete datos en la **lista** llama luego a **lista.notify()**, cuando salgamos del **wait()** tendremos datos disponibles en la lista, así que únicamente tenemos que leerlos (y borrarlos para no volver a tratarlos la siguiente vez). Existe otra posibilidad de que el hilo se salga del **wait()** sin que haya datos disponibles, pero la veremos más adelante.

**Notificar a los hilos que están en espera**

Dijimos que el hilo que mete datos en la lista tiene que llamar a **lista.notify()**. Para esto también es necesario apropiarnos del objeto lista con un **synchronized**. El código del hilo que mete datos en la lista quedará así

synchronized(lista)  
{  
   lista.add(dato);  
   lista.notify();   
}

Listo, una vez que hagamos esto, el hilo que estaba bloqueado en el **wait()** despertará, saldrá del **wait()** y seguirá su código leyendo el primer dato de la lista.

**wait() y notify() como cola de espera**

**wait()** y **notify()** funcionan como una lista de espera. Si varios hilos van llamando a **wait()** quedan bloqueados y en una lista de espera, de forma que el primero que llamó a **wait()** es el primero de la lista y el último es el último.

Cada llamada a **notify()** despierta al primer hilo en la lista de espera, pero no al resto, que siguen dormidos. Necesitamos por tanto hacer tantos **notify()** como hilos hayan hecho **wait()** para ir despertándolos a todos de uno en uno.

Si hacemos varios **notify()** antes de que haya hilos en espera, quedan marcados todos esos **notify()**, de forma que los siguientes hilos que hagan **wait()** no se quedaran bloqueados.

En resumen, **wait()** y **notify()** funcionan como un contador. Cada **wait()** mira el contador y si es cero o menos se queda bloqueado. Cuando se desbloquea decrementa el contador. Cada notify() incrementa el contador y si se hace 0 o positivo, despierta al primer hilo de la cola.