Programação Distribuída e Paralela – 2013/01
Grupo: João Gross, Jefferson Stoffel, José Luiz, Hugo Lazzari

Questão 2:

- sobre interrupções (interrupt/interrupted) e sinalizações (wait/notify

- explique eventuais similaridades e diferenças entre os dois mecanismos

O método wait é invocado para suspender a execução da thread e liberando os locks que tenham sobre aquela parte do código. O método wait pode disparar um InterruptedException que forçará a saída do wait. A documentação sugere que seja feita um teste e sempre coloar o wait em um while para testar se a thread deveria mesmo ter sido acordada.

Exemplo fornecido na documentação:

```
public synchronized guardedJoy() {
        while(!joy) {
         try {
                        wait();
              } catch (InterruptedException e) {}
        }
        System.out.println("...");
}
```

O Wait fica dentro de um loop e ele repete até que a condição para a continuação seja válida.
Temos os métodos para acordar que são o notify e o notifyAll.
O método notify acorda uma thread qualquer, não podemos escolher qual será acordada, e é indicada em casos onde temos milhares de threads que executam tarefas semelhantes, onde não importa qual delas será executada, para os outros casos é indicado o uso do notifyAll, pois se a thread deve ser executada ela será, senão provavelmente ela voltará a esperar com um wait. O notifyAll acorda todas as threads que estão com wait, depois disso elas ficam prontas para receber a execução, porém a troca não é imediata, para evitar trocas de contexto sendo que o notifyAll e o notify coloca elas como prontas para receber a execução.
Threads ainda suportam interrupções que seriam como notificações avisando que a thread deve fazer outra coisa, para isso ela tem um método chamado Thread.interrupted(), que retorna true se a thread foi interrompida, é importante lembrar que ao contrário do wait a thread não fica bloqueada, mas sim ela fica sabendo qual ação ela deve tomar, podendo inclusive ignorer a interrupção. Quando o método é chamado o flag interno é alterado, e existe outro método para checar sem mudar o status do flag. Invocando o método Thread.interrupt o flag é setado, sinalizando que a thread foi interrompida. Assim, pode-se até simular o funcionamento do wait/notify com interrupções, para isso deveríamos ter uma lista para colocar threads interrompidas e elas são até semelhantes.

Comparação:

Interrupções tem a vantagem de não serem bloqueantes. Porém, como ficam fazendo pooling, para saber se são interrompidas elas ficam consumindo processamento. O wait tem a vantagem de bloquear a thread sem precisar de nenhum teste adicional chamando somente o wait na thread sendo

que ela não pode ignorar isso. O interrupt pode ser ignorado, pois você não é obrigado a tomar uma ação mesmo que você tenha a flag setada como true. Utilizando wait e notify é mais simples de controlar o programa, já que com interrupções o programador que tem que checar se o flag para interromper foi setado e ele deve tomar alguma ação diferente e com isso ele pode perder tempo tendo que checar várias vezes.

Fontes:

http://docs.oracle.com/javase/tutorial/essential/concurrency/interrupt.html
http://docs.oracle.com/javase/tutorial/essential/concurrency/guardmeth.html