**Prioridades das threads Java**

Quando estamos trabalhando com threads em Java, precisamos estar cientes de que cada thread possui uma prioridade de execução. É por meio da prioridade de cada uma que o gerenciador de threads decidirá qual thread deverá ser executada primeiro.

Por padrão, todas as threads possuem prioridade NORM\_PRIORITY. Esta é uma constante que possui o valor 5 e está declarada na classe Thread. Além disso, cada thread herda automaticamente a prioridade da thread que a criou.

As constantes MAX\_PRIORITY (prioridade máxima), MIN\_PRIORITY (prioridade mínima) and NORM\_PRIORITY (prioridade normal) são usadas para definir as prioridades das threads Java. Veja um exemplo no qual temos duas threads. A primeira possui a prioridade máxima enquanto a segunda possui a prioridade mínima:

// criamos uma classe que servirá como thread

class MinhaThread extends Thread{

 private String nome;

 public MinhaThread(String nome){

 this.nome = nome;

 }

 public void run(){

 for(int i = 1; i <= 20; i++){

 System.out.println(nome + ": " + i);

 }

 }

}

public class Estudos{

 public static void main(String[] args){

 // vamos criar duas threads

 MinhaThread t1 = new MinhaThread("Thread 1");

 t1.setPriority(Thread.MAX\_PRIORITY); // prioridade máxima

 t1.start();

 MinhaThread t2 = new MinhaThread("Thread 2");

 t2.setPriority(Thread.MIN\_PRIORITY); // prioridade mínima

 t2.start();

 }

}

Execute este exemplo e veja como a segunda thread só é executada quando a primeira finaliza. Remova as linhas que definem a prioridade e note como o tempo de cada thread é novamente fracionado. É importante ter em mente que aqui estamos falando de um ambiente de processador único. Em ambientes de múltiplos processadores o comportamento pode ser diferente do abordado na dica.

Não devemos confiar em prioridades de threads quando o objetivo é aguardar a finalização de uma thread e só então permitir o processamento das instruções contidas no método run() de outra thread. Para estes casos o recomendável é usar alguma forma para sinalizar as demais threads de que a thread atual concluiu sua tarefa.