Aula 08

Multithreading

Problemas...

- Coração, pulmão e pensamentos ...
- Corretor ortográfico, impressão em segundo plano, etc...
- Multitarefas na programação...como fazer ?

Multithreading

- Thread - Linhas de execução;
- Execução concorrente;
- Todo programa java é uma thread;
- O coletor de lixo do java é uma thread;
- Há duas formas de programar uma thread
- Extendendo a classe Thread;
- Implementando a interface Runnable;

Extendendo a classe Thread

- Método run() contem o código a ser executado;
- Exemplo:

```
public class PrimeiraThread extends Thread{

    public void run(){
       ...instruções que a thread executará
    }
}
```

Executando uma thread

- Para executar uma thread utilizamos o método start():
- Thread thread = new Thread()
- thread.start
- NUNCA utilize o método run para executar a thread.

Implementado Runnable

- O método run deverá ser implementado;
- Exemplo:

```
public class PrimeiraThread implements Runnable{

    public void run(){
        ...instruções que a thread executará
    }
}
```

Executando

- Utilizamos o método start para executar uma thread;
- Exemplo
- Thread thread = new Thread(UmaThread)
- thread.start
- NUNCA utilize o método run para executar a thread.

Prioridades de thread

- Método setPriority
- As prioridades estão entre 1 (menor prioridade) e 10 (maior prioridade)
- MAX_PRIORITY, MIN_PRIORITY e NORM_PRIORITY

Executando Threads no JSE 5.0

- Preferencialmente utiliza-se threads que implementam a interface Runnable;
- São executadas por um objeto que implementa a interface Executor (java.util.concurrent)
- Cria e gerencia um pool de threads
- ExecutorService thread =
- Executors.newFixedThreadPool(3);
- thread.execute(Thread);

Fazendo um exemplo

- Observe o código:

```
public class ExemploThread implements Runnable{
  private String nome;
  private int i;
  public ExemploThread(String nome){
    this.nome = nome;
  }
  public void run() {
    while( true ){
      System.out.printf("\n%s -> %d", nome, i++);
    }
  }
}
```

Fazendo um exemplo cont...

- Código para execução:

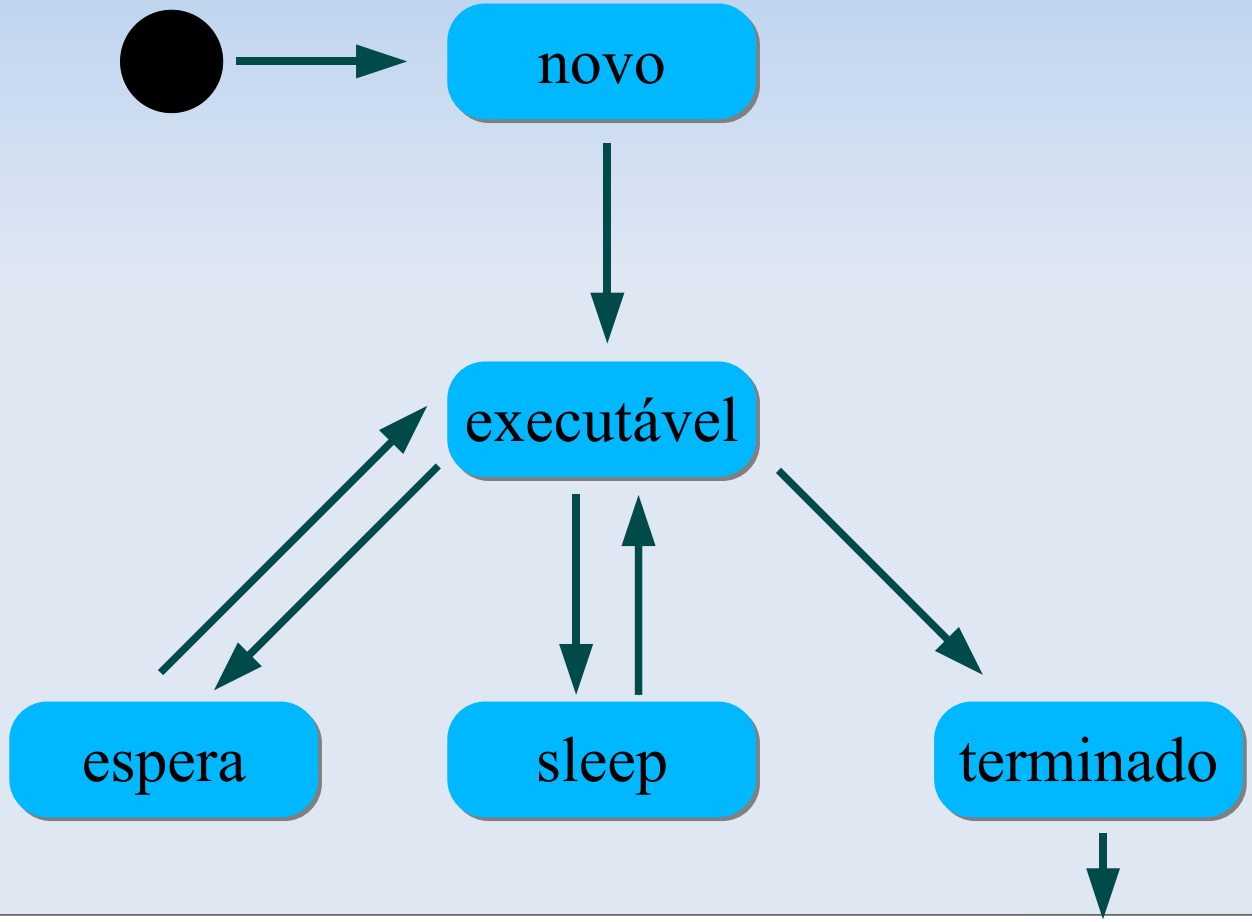
```
import java.util.concurrent.*;
public class TesteThreadNovo {
  public static void main(String[] args) {
    ExemploThread t1 = new ExemploThread("=====Thread 1");
    ExemploThread t2 = new ExemploThread("*****Thread 2");

    ExecutorService threads = Executors.newFixedThreadPool(2);
    threads.execute(t1);
    threads.execute(t2);
    threads.shutdown();
  }
}
```

Fazendo algumas alterações

- Coloque mais threads em seu exemplo;

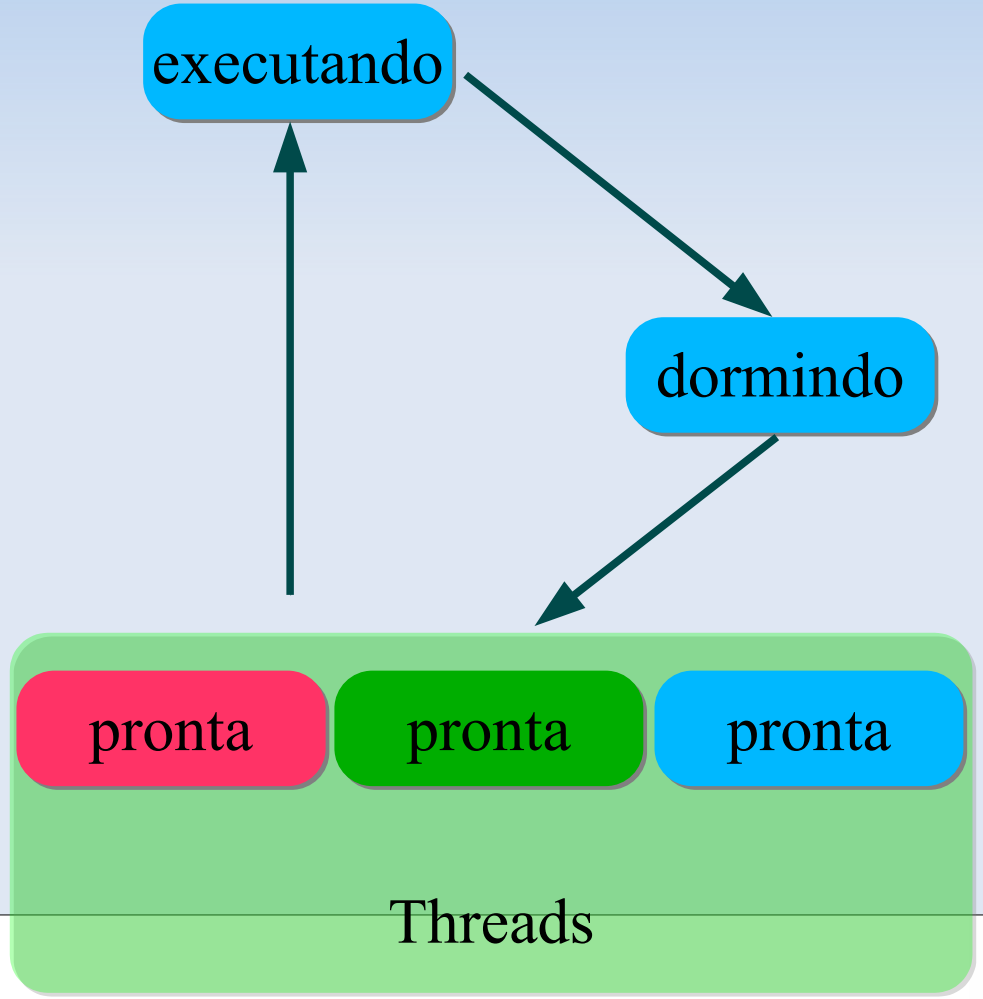
Estados de thread



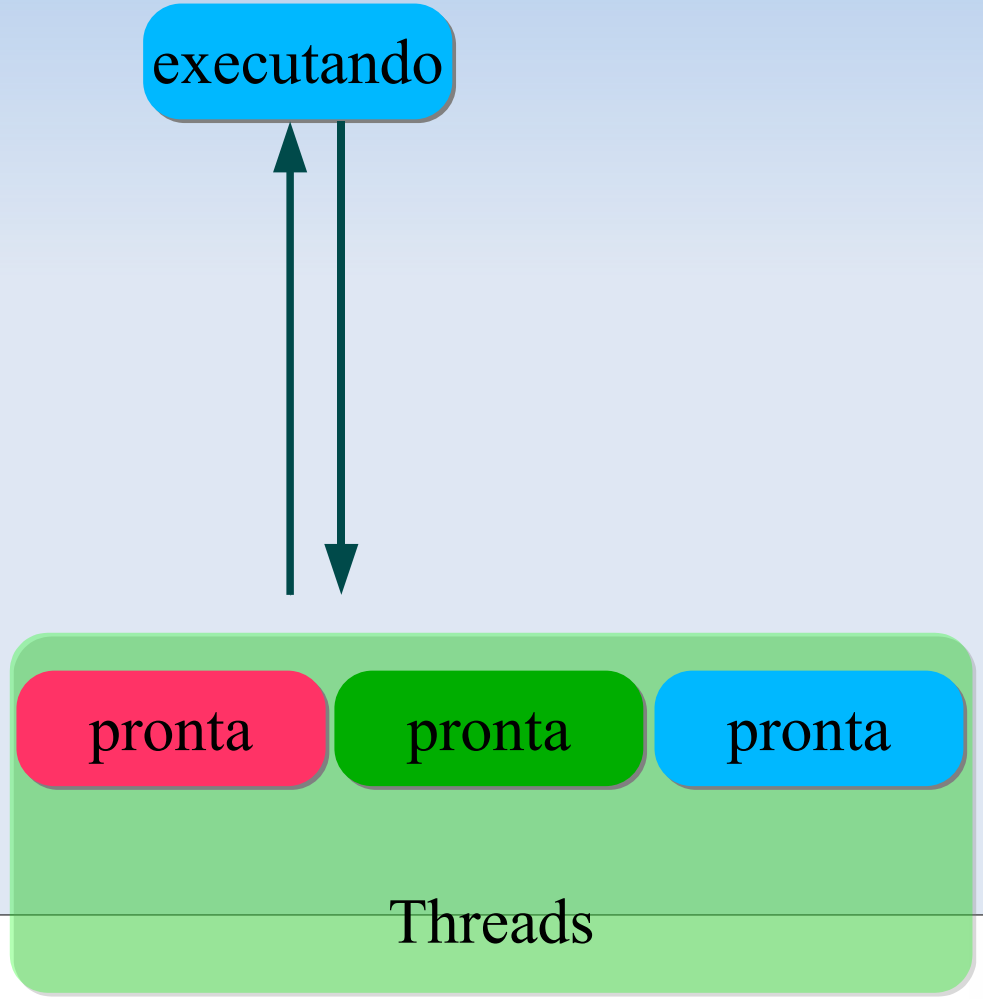
Alguns métodos da classe Thread

- Thread.sleep(int miliseconds);(dormindo)
- Suspende execução da thread atual e faz ela esperar miliseconds para retornar a execução;
- Thread.yield(); (desistindo)
- Faz com que a thread atual desista da execução cedendo a vez para uma outra thread qualquer;

...(dormindo) zzZZZzz



...(desistindo)



Thread

- Não há como prever qual thread entrará em execução primeiro;
- Cada sistema operacional se comporta de maneira diferente;

Um problema...

- Crie a classe:

```
public class Troca {
  int a, b;
  public void faça( boolean bol ){
    if( bol ){
      a = 5;
      b = 7;
    }
    else{
      a = 11;
      b = 13;
    }
    try {
      Thread.sleep(500);
    }
    catch (InterruptedException ex) {
    }
    if( bol ){
      System.out.printf("\n%d = 5 e %d = 7", a, b);
    }
    else{
      System.out.printf("\n%d = 11 e %d = 13", a, b);
    }
  }
}
```

O que aconteceu ??

Sincronizando Threads

- Palavra chave synchronized;
- Permite que uma thread por vez execute o que estiver dentro de um bloco synchronized;
- Uso:
- synchronized(Objeto){ ... }
- public synchronized void metodo(){..}

Alterando nosso exemplo...

- Crie a classe:

```
public class Troca {
  int a, b;
  public synchronized void faça( boolean bol ){
        .
  }
}
```

O que aconteceu ??

AMBIENT
INFORMÁTICA

Mais métodos...

- .wait()
  - Espera até outra thread invocar o método notify;
- .notify();
  - Instrui a thread em espera a entrar no estado de executável;
- .notifyAll();
  - Instrui todas as threads em espera a entrar no estado de executável;

Duvidas??

AMBIENT
INFORMÁTICA

Conclusão

- API de concorrência do Java;
- Técnicas de sincronização de threads;
- Toda aplicação java é uma thread;

Exercicios ... pagina 826

Fim

AMBIENT
INFORMÁTICA