# Programação Paralela/Concorrente

José Aélio de Oliveira Júnior

## Programação Concorrente

- O mundo real funciona concorrentemente: várias atividades podem ser executadas em paralelo. Exemplo: uma pessoa pode estar
  - respirando, e,
  - falando, e
  - escrevendo, e
  - lendo, etc.
- Computadores também operam concorrentemente. Exemplo: um computador pode estar
  - compilando um programa, e
  - recebendo uma mensagem, e,
  - imprimindo um arquivo, e,
  - tocando música, etc.



## Objetivos da Programação Concorrente

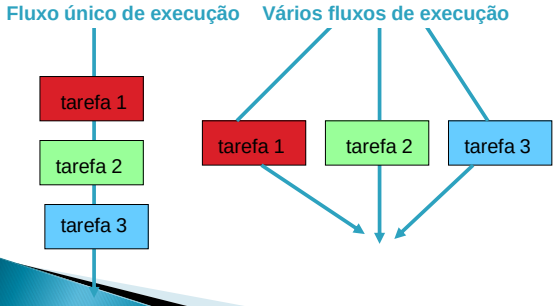
- Reduzir o tempo total de processamento
  - múltiplos processadores
- Aumentar confiabilidade e disponibilidade
  - processadores distribuídos
- Obter especialização de serviços
  - sistemas operacionais
  - simuladores
- Implementar aplicações distribuídas
  - correio eletrônico



## Conceitos de Programação Concorrente

- Uma unidade concorrente é um componente de um programa que não exige a execução sequencial, ou seja, que sua execução seja realizada antes ou após a execução de outros componentes do programa
- O termo programação concorrente é usado no sentido abrangente, para designar a programação paralela e a programação distribuída
- Concorrência relaciona-se com fluxo de controle: em um programa, existe mais de um fluxo de controle ativo.

## Fluxo Seqüencial x Concorrente




5

## Execução Concorrente

- Execução concorrente, também conhecida como execução paralela, não significa execução simultânea
- A execução de unidades concorrentes admite as seguintes possibilidades:
  - **Pseudo-paralela**: Execução em um único processador;
  - **Paralela**: Execução em vários processadores que compartilham uma memória;
  - **Distribuída**: Execução em vários processadores independentes, sem compartilhamento de memória.
- O programa geralmente não possui controle sobre a ordem e o tempo de execução das unidades concorrentes

6

## Conceito de Threads

- Definição básica: " Fluxo de controle sequencial isolado dentro de um programa."
- Programas multithreaded: Múltiplos threads concorrentes de execução num único programa, realizando várias tarefas "ao mesmo" tempo.
  - Exemplo: programa do usuário + coleta de lixo
- Diferentes threads podem executar em diferentes processadores, se disponíveis, ou compartilhar um processador único
- Diferentes threads no mesmo programa compartilham um ambiente global (memória, processador, registradores, etc.)

7

## Conceito de Threads

- A linguagem Java possui funcionalidades MULTITHREADING na própria estrutura da linguagem.
- C e C++ necessitam de biblioteca especifica para processamento MULTITHREADING
  - – Posix p_thread

8

## Threads em JAVA

- Em Java, threads são implementadas como uma CLASSE
  - Pacote java.lang.Thread
  - É uma extensão da classe Thread
  - Contrutores:
    - public Thread (String nome_da_thread);
    - public Thread ( ); // o nome sera Thread
      - Thread-1, Thread-2,...

9

## Principais métodos

- `start()`: inicia a execução da thread (método run)
- `suspend()`: suspende a execução da thread que está executando
- `sleep()`: faz a thread que está executando dormir por um tempo determinado
- `resume()`: resume a execução de uma thread suspensa
- `stop()`: termina a execução de uma thread; a thread não pode ser mais executada.

10

## Principais métodos

- `join()`: método que espera o término da THREAD para qual foi enviada a mensagem para ser liberada.

11

## Estados de uma thread

- Criação: Neste estado, o processo pai está criando a thread que é levada a fila de prontos;
- Execução: Neste estado a thread está usando a CPU;
- Pronto: Neste estado a thread avisa a CPU que pode entrar no estado de execução e entra na fila de prontos;
- Bloqueado: Neste estado, por algum motivo, a CPU bloqueia a thread, geralmente enquanto aguarda algum dispositivo de I/O;
- Término: Neste estado são desativados o contexto de hardware e a pilha é desalocada.
- Esperando e Finalizado.

12

## Prioridade de thread

- Em Java, a prioridade é determinada com um inteiro entre 1 e 10.
- A prioridade padrão é o valor 5.
- 10 é a maior prioridade e 1 é a menor.
- A THREAD herda a prioridade da THREAD que a criou.
- void setPriority(int prioridade);
- int getPriority( );

13

## Threads em Java

- Para que uma thread possa executar um método de uma classe, a classe deve:
  - Herdar (extend) a classe Thread (o qual implementa a classe Runnable em si), ou;
  - Implementar a interface Runnable

14

## Threads em Java

- **Herdando a class Thread:**

```
class MinhaThread extends Thread {
   public void run() {
         System.out.println("Bom Dia !");
   }
}
```

**O método run() contém o código que a thread executa.**

```
class Teste1 {
   public static void main(String Args[]) {
         new MinhaThread().start();
   }
}
```

**Para executar a thread é necessário instanciar a classe MinhaThread e invocar o método start().**

15

## Threads em Java

- **Implementando a classe Runnable:**

```
class MinhaThread2 implements Runnable {
   public void run() {
         System.out.println("Bom Dia !");
   }
}
```

**Desta forma, a classe MinhaThread pode herdar uma outra Classe.**

```
class Teste2 {
   public static void main(String Args[]) {
         new Thread(new MinhaThread2()).start();
   }
}
```

16

## Execução paralela de threads

```
class ImprimirThread_1 implements Runnable {
   String str;
   public ImprimirThread_1(String str) {
         this.str = str;
   }
   public void run() {
         for(;;)
            System.out.println(str);
   }
}
class TesteConcorrente {
   public static void main(String Args[]) {
         new Thread(new ImprimirThread_1("A")).start();
         new Thread(new ImprimirThread_1("B")).start();
   }
}
```



## Referências Bibliográficas


- Oracle. Thread (Java Platform SE 7). Disponível em: <http://download.oracle.com/javase/7/docs/api/java/lang/Thread.html>. Acesso em: 13 agosto de 2011.