XV Jornada de Cursos CITi

Aula 2

Programação Concorrente

Benito Fernandes
Fernando Castor
João Paulo Oliveira
Weslley Torres

Agenda

Conceitos básicos de Threads em Java

Benefícios de Thread

Estados,Métodos, Prioridades

Exercícios

Processos

“Um processo é basicamente um programa em execução...” **Tanenbaum**

Contem seu próprio espaço de memória.

Contem os recursos que ele terá em Runtime.

Thread

O conceito de thread está intimamente ligado ao conceito de processo, assim é fundamental entender o que são processos, como eles são representados e colocados em execução pelo Sistema Operacional, para em seguida entender as threads.

Thread X Processos

Thread	Processo
Mais Leve	Mais Pesado
Recursos compartilhados	Recursos Próprios(I/O, ...)
Enderecamento compartilhado	Endereçamento Próprio
Ambiente de execução Compartilhada	Ambiente de Execução próprio
Existe dentro de um Processo	Possui ao menos um thread

Benefícios de Thread

A criação e terminação de uma thread nova é em geral mais rápida do que a criação e terminação de um processo novo.

A comutação de contexto entre duas threads é mais rápido do que entre dois processos.

Benefícios de Thread

A comunicação entre threads é mais rápida do que a comunicação entre processos.

Multiprogramação usando o modelo de threads é mais simples e mais portável do que multiprogramação usando múltiplos processos.

Thread

As linguagens modernas como Java e C# possuem funcionalidades MULTITHREADING na própria estrutura da linguagem.

C e C++ necessitam de biblioteca especifica para processamento MULTITHREADING

Implementação de uma thread

Existem duas formas de criar explicitamente um thread em Java:

- Estendendo a classe `Thread` e instanciando um objeto desta nova classe.
- Implementando a interface `Runnable` e passando um objeto desta nova classe como argumento do construtor da classe `Thread`.

Nos dois casos a tarefa a ser executado pelo thread deverá ser escrita no método `run()`.

A Interface Runnable

Através da utilização da interface Runnable é possível criar classes que representem um thread sem precisar estender a classe Thread.

A criação de uma nova thread é feita através da instanciação de um objeto thread usando o objeto que implementa a interface Runnable.

Herdando da Classe Thead

```
public class HelloThread extends Thread {

    public void run() {
        System.out.println("Hello from a thread!");
    }

    public static void main(String args[]) {
        (new HelloThread()).start();
    }

}
```

Interface Runnable

```
public class HelloRunnable implements Runnable {

    public void run() {
        System.out.println("Hello from a thread!");
    }

    public static void main(String args[]) {
        (new Thread(new HelloRunnable())).start();
    }

}
```

Exemplo

```
class RunnableThread implements Runnable {

    Thread runner;
    public RunnableThread() {
    }
    public RunnableThread(String threadName) {
        runner = new Thread(this, threadName); // (1) Cria uma nova thread.
        System.out.println(runner.getName());
        runner.start(); // (2) inicia a thread.

    }
    public void run() {
        //mostra informações sobre a Thread atual
        System.out.println(Thread.currentThread());
    }
}
```

```
public class RunnableExample {

    public static void main(String[] args) {

        Thread thread1 = new Thread(new RunnableThread(), "thread1");
        Thread thread2 = new Thread(new RunnableThread(), "thread2");
        RunnableThread thread3 = new RunnableThread("thread3");

        //inicia as threads
        thread1.start();
        thread2.start();

        try {
            //pede para esperar por 1 segundo
            Thread.currentThread().sleep(1000);
        } catch (InterruptedException e) {
        }
        //Apresenta informações sobre a Thread atual
        System.out.println(Thread.currentThread());
    }
}
```

Possíveis resultados

```
thread3
Thread[thread2,5,main]
Thread[thread1,5,main]
Thread[thread3,5,main]
Thread[main,5,main]
```

```
thread3
Thread[thread1,5,main]
Thread[thread3,5,main]
Thread[thread2,5,main]
Thread[main,5,main]
```

Exemplo

```
3  class Cavalo3 extends Thread {
4
5      // cada cavalo possui uma qde passos atual e
6      // uma colocação
7      private int passos = 1, colocacao;
8
9      // contador conta a qde de cavalos que
10     // já ultrapassaram a linha de chegada
11     private static int contador = 1, qde_cavalos=0;
12
13     // podemos ter um páreo de no máximo 10 cavalos
14     private static Cavalo3[] cavalos = new Cavalo3[10];
15
16     public Cavalo3(String nome) {
17             // a classe Thread pai pode receber um nome;
18             super(nome);
19             cavalos[qde_cavalos] = this;
20             qde_cavalos++;
21     }
22
23     // imprime a posição atual dos cavalos
24     public static void imprime() {
25             for (int i=0; i< qde_cavalos; i++) {
26                     System.out.print("\nCavalo (" +
27                     cavalos[i].getName() + "): ");
28                     for (int j=0; j< cavalos[i].passos; j++)
29                     System.out.print('>');
30             }
31             System.out.println();
32     }
33
```

Exemplo (cont)

```
34      // trecho executável de cada thread
35      public void run() {
36              int tempo;
37              while(true) {
38                      imprime();
39                      passos++;
40                      try {
41                              if (passos == 51) {
42                                      colocacao = contador++;
43                                      break;
44                              }
45                              // pausa por um tempo randomico
46                              // entre 0 a 500 ms
47                              tempo = (int) (Math.random() * 500);
48                              Thread.sleep(tempo);
49                      }
50                      catch (InterruptedException e) { }
51              }
52      }
53
54      public int getColocacao() {
55              return colocacao;
56      }
57  }
```

```
3   public class CorridaDeCavalo3 {
4       public static void main(String[] args)
5       throws InterruptedException {
6               Cavalo3 c1 = new Cavalo3("1");
7               Cavalo3 c2 = new Cavalo3("2");
8               c1.start();
9               c2.start();
10              System.out.println("Classificacao final:");
11              System.out.println("Cavalo (1): "+c1.getColocacao());
12              System.out.println("Cavalo (2): "+c2.getColocacao());
13      }
14  }
```

Escolhendo uma forma de implementação

Herdar de Thread sempre que possível

Java não permite herança múltipla

Deve ser feito através da implementação da interface Runnable.

Applet deve ser sub-classes da classe Applet, assim applets só podem implementar threads através da implementação da interface Runnable.

Exercícios

- Crie um classe *ContadorTempo* que possui um atributo tick inteiro e um metodo nextTick()
- Crie um classe *Relogio* que **possui** um contadorTempo e atraves de uma thread chama o metodo nextTick a cada segundo
- Crie um classe *Cronometro* que **herda** de ContadorTempo e através de uma thread chama o nextTick()

Estados de uma thread

new

Runnable

Running

Terminated

blocked

Suspended

suspended-blocked

Estados de uma thread

start()

stop()

suspend()

resume()

join()

interrupt()

sleep()

yield()

isInterrupted()

isAlive()

isDaemon()

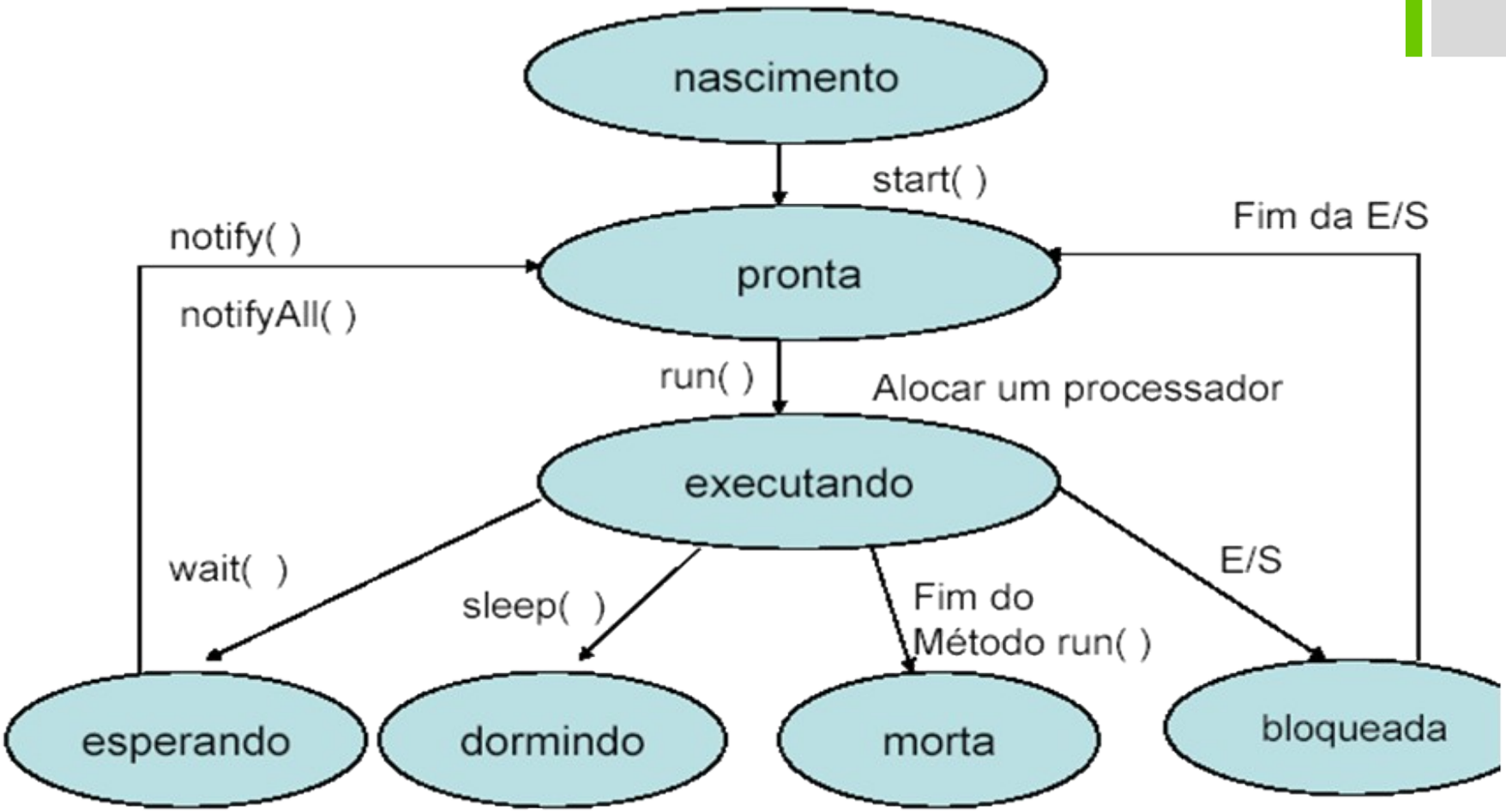
setPriority(*int*)

getPriority()

setName(*String*)

getName()

Ciclo de Vida



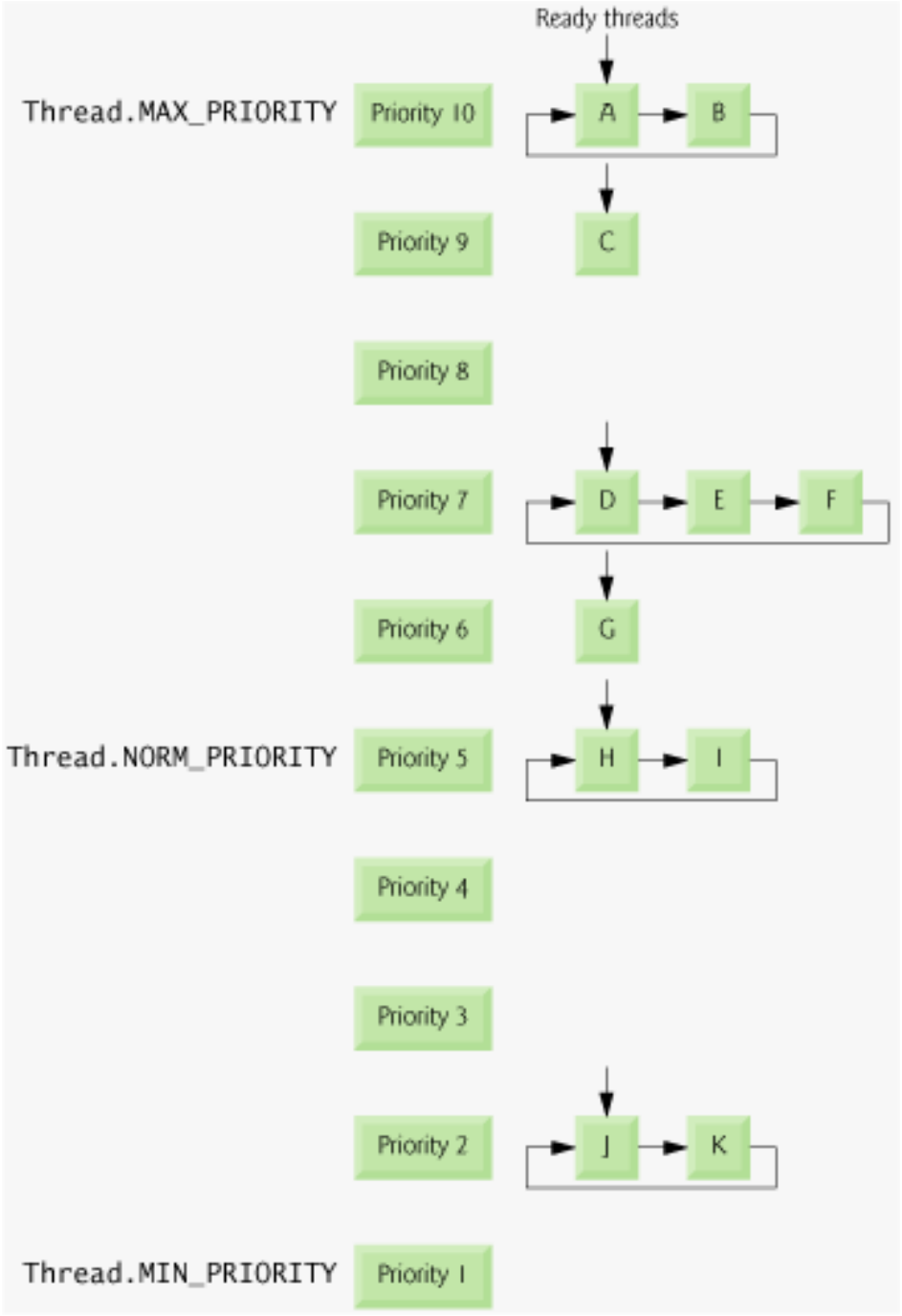
Escalonamento

Mecanismo que determina como os threads irão utilizar tempo de CPU.

Somente os threads no estado *runnable* são escalonados para ser executados.

Java permite a atribuição de prioridades para as threads.

Threads com menor prioridade são escalonados com menor freqüência.



Fonte: Java, Como programar. Deitel, 6ª Edição

Constantes de prioridades

MAX_PRIORITY

MIN_PRIORITY

NORM_PRIORITY

Exercícios 2

Implemente uma Corrida de Sapos!

Crie um Classe sapo que herda de Thread

- Atributos: distanciaPercorrida, distanciaPulo...

Crie uma Classe Corrida de Sapos

- Atributos: distanciaCorrida, NumSapos

Referências

www.di.ubi.pt/~operativos/teoricos/capitulo6.ppt

http://wiki.sintectus.com/bin/view/GrupoJava/SlidesSincronizandoTh

Sistemas Operacionais Modernos - Andrew S. Tanenbaum