Fatec Araras

Tópicos Especiais em Sistemas para Internet I

Prof. Orlando Saraiva Júnior
orlando.nascimento@fatec.sp.gov.br

Fatec Araras

"Do. Or do not. There is no try."

YODA

The Empire Strikes Back

Tópicos Especiais em Sistemas para Internet I

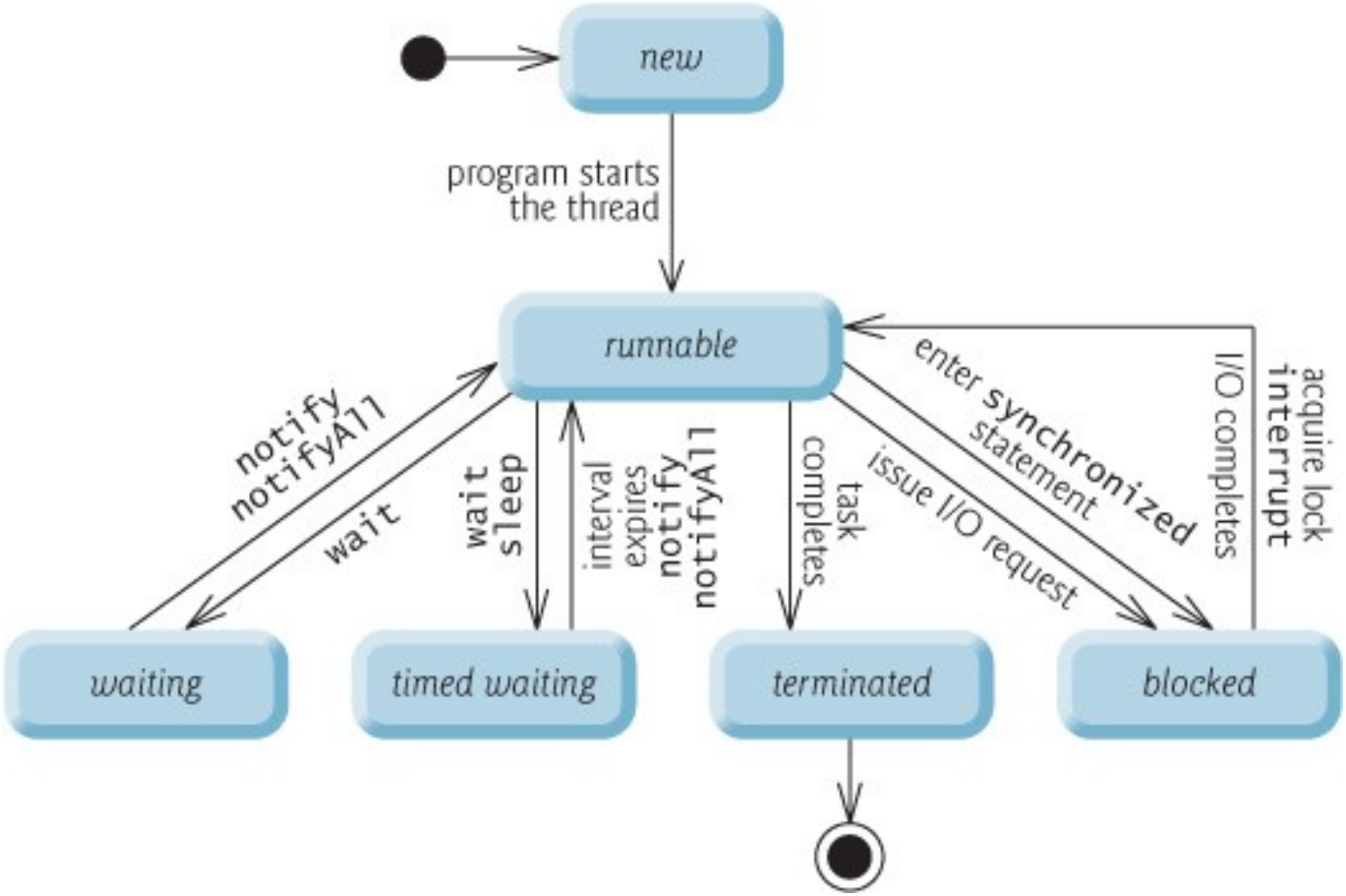
Objetivo da aula

Fatec Araras

Novos conceitos:

- *Multithreading*
  - *Runnable*
  - Framework *Executor*
  - Sincronizar *threads*

Ciclo de vida de uma thread



Prioridades de thread

Fatec Araras

Toda thread do Java tem uma prioridade de thread que ajuda a determinar a ordem em que são agendadas. As prioridades variam entre MIN_PRIORITY (uma constante de 1) e MAX_PRIORITY (uma constante de 10). Por padrão, toda thread recebe a prioridade NORM_PRIORITY (uma constante de 5). Cada nova thread herda a prioridade da thread que a cria.

Todas as constantes citadas encontram-se na classe **Thread**. Recomenda-se não criar ou utilizar explicitamente objetos Thread para implementar concorrência mas, em vez disso, utilizar a interface Executor.

Criando Threads

Fatec Araras

A forma preferida de criar aplicativos Java de múltiplas threads é implementando a interface ***Runnable*** (do pacote java.lang). Um objeto Runnable representa uma “tarefa” que pode ser executada concorrentemente com outras tarefas.

A interface *Runnable* declara o método ***run*** único, que contém o código que define a tarefa que um objeto *Runnable* deve realizar.

Programa:

ThreadCreator

PrintTask

Threads com o framework Executor

Fatec Araras

Embora seja possível criar threads explicitamente, recomenda-se utilizar a interface ***Executor*** para gerenciar a execução dos objetos *Runnable* para você.

Em geral, um objeto Executor cria e gerencia um grupo de threads chamado pool de threads para executar Runnables.

Programa:

ThreadCreator

PrintTask

Sincronização de Threads

Fatec Araras

Quando múltiplas threads compartilham um objeto e ele é modificado por uma ou várias delas, podem ocorrer resultados indeterminados. O problema pode ser resolvido fornecendo somente uma thread, por vez, acesso exclusivo, para manipilar o objeto compartilhado. A este processo, denominamos **sincronização de Threads**.

Nos programas a seguir, entenderemos os perigos de compartilhar um objeto sem a sincronização apropriada

Programa:

ArrayWriter / SimpleArray / SharedArrayTest (main)

Sincronização de Threads

Fatec Araras

O erro apresentado pode ser atribuito ao fato de que o objeto compartilhado, SimpleArray, não é seguro para threads, ou seja, suscetível a erro se acessado concorrentemente por múltiplas threads. O problema reside no método add, que armazena o valor de writeIndex, coloca um novo valor neste elemente e, então, incrementa writeIndex.

Para resolver este problema, desejamos que as três operações (armazenar writeIndex, gravar no array e incrementar writeIndex) seja uma **operação atomica**. A atomicidade pode ser alcançada com a palavra reservada ***synchronized***

Programa:

ArrayWriter / SimpleArray / SharedArrayTest

Fatec
Araras

Dúvidas

Prof. Orlando Saraiva Júnior
orlando.nascimento@fatec.sp.gov.br

Para casa

Defina cada um dos seguintes termos:

Thread

Multithreading

Estado executável

Estado de espera sincronizada

Documentação:

https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/Executor.html
https://docs.oracle.com/javase/8/docs/api/java/lang/Runnable.html