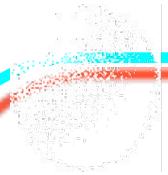

Informática
UFRGS

Java

Threads: interrupções

Autoria

- **Autor**
  - C. Geyer
- **Local**
  - Instituto de Informática
  - UFRGS
  - disciplinas :
    - Programação Distribuída e Paralela (CIC e ECP)
    - Programação com Objetos Distribuídos (PPGC)
  - versão atual:
    - V01, março de 2012

Fontes

- **Tutorial sobre Java**
  - Oracle
  - http://docs.oracle.com/javase/tutorial/index.html

Súmula

- **Súmula**
  - Java Threads x interrupções

- **Bibliografia**
  - Lea, D. Concurrent Programming in Java - Design Principles and Patterns. Addison-Wesley, 1997.
  - Oaks, S. and Wong, H. Java Threads. O'Reilly, 1997.
  - Goetz, B. et al. Java Concurrency in Practice. Addison-Wesley, 2006.

Bibliografia

Bibliografia (cont.)

- Schildt, H. Java The Complete Reference. McGraw-Hill, 2011.
- Schildt, H. Java – a Beginner's Guide. McGraw-Hill, 2011.
- Zakhour, S. The Java Tutorial: a Short Course on the Basics. Prentice-Hall, 2012.
- Campione, Mary e Walrath, K. The Java Tutorial. Addison-Wesley, 2a. ed., 1998.
- Cornell, Gary e Horstmann, Cay. Core Java. Prentice Hall, 2007.
- Flanagan, David. Java in a Nutshell. O'Reilly Assoc., 2a. ed., 1997.

Bibliografia

Bibliografia (cont.)

- Arnold, K. and Gosling, J. The Java Language. Addison-Wesley, 1996.
- Orfali, R. and Harkey, D. Client/Server Programming with JAVA and CORBA. John Wiley, 1997.
- Wutka, M. Java - Expert Solutions. Que, 1997.
- Walnum, Clayton. Java by Examples. Que, 1996.

Bibliografia

- **Bibliografia (cont.)**
  - Grand, Mark. Java Language Reference. O'Reilly Assoc., 2a. ed., 1997.
  - Niemeyer, P. e Peck, Josh. Exploring Java. O'Reilly Assoc., 2a. ed., 1997.

Endereços

- Site Oracle:
  - www.java.com
  - http://www.oracle.com/technetwork/java/index.html
  - http://www.oracle.com/technetwork/java/javase/documentation/index.html
  - http://www.oracle.com/technetwork/java/javase/documentation/tutorials-jsp-138802.html
  - http://docs.oracle.com/javase/tutorial/index.html
  - (acessados em 07/03/2012)

- **Endereços**
  - site da Sun sobre tecnologia Java
    - http://java.sun.com
  - notas técnicas
    - http://java.sun.com/jdc/tecDocs/newsletter/index.html
  - tutorial Java
    - http://javasoft.com/docs/books/tutorial/index.html
  - Documentação Java:
    - http://java.sun.com/docs/white/index.html

Endereços

- **Endereços (cont.)**
  - tutor Java
    - http://www.mercury.com/java-tutor/
  - Java ensina Java
    - http://www.neca.com/~vmis/java.html
  - outro tutorial Java
    - http://www.phrantic.com/scoop/onjava.html

Java Threads x Interrupções

- **Conceitos de interrupções**
  - Uma interrupção é uma indicação de que uma thread deve interromper o que está fazendo e fazer algo mais
  - Cabe ao programador definir “o que mais” a thread vai fazer
  - Frequentemente a thread termina
  - Método “interrupt”
    - Uma thread envia uma interrupção a outra thread
    - Uma thread executa “t.interrupt”
    - “t” é a thread a ser interrompida
    - Semântica depende do que “t”está fazendo (estado, ...)
      - Por exemplo, se bloqueada em “wait” (sinalização) ou em operação de I/O

- **Conceitos de interrupções**
  - A thread interrompida (“t”) deve estar preparada para a interrupção
  - Isto depende em geral do que a thread está fazendo ao ser interrompida

Exemplo com sleep

- Método sleep
  - Thread corrente permanece bloqueada o tempo indicado
  - public static void sleep(long millis) throws Interrupted Exception
- Descrição do exemplo
  - Thread (método run) imprime uma mensagem a cada 4 segundos
  - Se thread for interrompida, deve retornar simplesmente
  - Muitos métodos, como o “sleep”, são criados de modo a cancelar a operação atual e retornar imediamente em caso de receberem uma interrupção
- (tutorial Oracle)

Exemplo sleep

Exemplo com sleep

- Trecho de código para tratar interrupção se em sleep
- Em método run

```
for (int i = 0; i < importantInfo.length; i++) {
    // pausa de 4 segundos
    try {
        Thread.sleep(4000);
    } catch (InterruptedException e) {
        // thread foi interrompida: não emitir mais mensagens
        return;
    }
    // imprime uma mensagem
    System.out.println(importantInfo[i]);
}
```

Verificação de Interrupção

- **Verificação de interrupção?**
  - Se thread permanece muito tempo sem chamar métodos que disparem “InterruptedException” como o “sleep”?
  - Nesses casos, a thread deve periodicamente verificar se foi interrompida
  - Método “Thread.interrupted”
    - Retorna “true” se uma interrupção foi recebida

Verificação de Interrupção

- **Exemplo de verificação de interrupção**
  - Thread executa processamento longo para cada "entrada"
  - A cada passo, thread verifica se foi interrompida
  - Se sim, retorna
  - Trecho de código (tutorial Oracle):

```
for (int i = 0; i < inputs.length; i++) {
    heavyCrunch(inputs[i]);                    // longo processamento
    if (Thread.interrupted()) {
        // thread foi interrompida; não mais "crunching"
        return;
    }
}
```

Estado de Interrupção

- **Estado de interrupção**
  - A implementação de interrupção usa uma variável interna (“flag”) conhecida como “estado de interrupção”
  - Método “Thread.interrupt” coloca a variável em “true”
  - Quando thread interrompida executa “Thread.interrupted”, testando a variável, essa é colocada em “false”
  - Método “isInterrupted” testa se uma (outra) thread foi interrompida
    - Esse método não altera a variável

Resumo Java Threads x Interrupções

- **Resumo Java threads x interrupções**
  - Métodos lançam exceções como sleep
  - Exceções devem ser tratadas
  - Uma thread pode ser interrompida por outra
  - A interrupção também deve ser tratada
  - Thread interrompida tranca atividade atual e executa uma alternativa
  - Isto pode ser feito em exceções de métodos como sleep
  - Frequentemente thread termina, retornando
  - Se thread só calcula não executando métodos bloqueantes como sleep
    - => ela pode testar periodicamente se houve uma interrupção

Revisão

Revisão Java threads x interrupções

- **Revisão de Java threads x interrupções**
  - Qual a relação entre métodos como sleep e exceções?
  - Em caso de interrupções o que uma thread deve fazer?
  - Se thread não chama métodos bloqueantes, o que ela deve fazer?

Java

Threads: interrupções