INF0045
Desenvolvimento de Software Concorrente

Prof. Me. Elias Ferreira

Local

Quinta-feira: INF 152

Campus Samambaia

Goiânia

Agenda

Apresentações

Plano de Ensino

Apresentações

- Professor
  - Formação
  - Experiência
  - Expectativa
- Alunos
  - Nome
  - Experiência no mercado
  - Expectativa
  - Etc

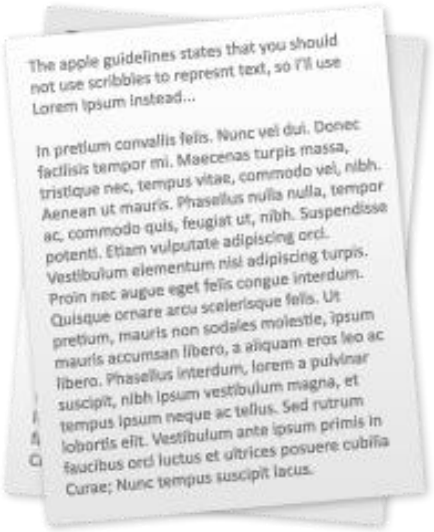
Disciplina

- Desenvolvimento de Software Concorrente
- Carga horária: 64 horas

Ambiente moodle

- ead.inf.ufg.br
- Disciplina: **Desenvolvimento de Software Concorrente 2016/1**
- Código de acesso: **DSC.20161.ES**

Plano de Ensino



Ementa

- **Liveness**: em algum momento o programa entra em um estado consistente
- **Safety**: o programa nunca entra em um estado inconsistente
- **Semáforos**: um tipo de variável (semáforo) que pode sofrer duas operações básicas: DOWN e UP

Ementa

- **Locks**: é um mecanismo de sincronização de processos/threads.
- **Threads**: são entidades entidades escalonadas para executarem na CPU
- **Deadlocks**: situação na qual um, ou mais processos, fica eternamente impedido...
- Implementações de algoritmos concorrentes.

Objetivo Geral

Expor o estudante a conceitos, desafios e ferramentas disponíveis para o desenvolvimento de software concorrente e à prática de tal atividade.

Objetivos Específicos

- Discutir os principais desafios diante da atividade de desenvolvimento de software concorrente;
- Explicar e descrever os conceitos básicos relacionados à concorrência em software;
- Identificar as principais ferramentas disponíveis para o desenvolvimento de software concorrente;

Objetivos Específicos

- Explicar e descrever as características que são desejáveis em uma ferramenta de desenvolvimento de software concorrente;
- Explicar e descrever os principais algoritmos concorrentes;
- Explicar e descrever as propriedades de liveness e safety e o conceito de deadlock;

Objetivos Específicos

- Implementar programas concorrentes em Java usando threads com foco em propriedades de liveness e safety;
- Implementar programas concorrentes em Java usando threads com a técnica de semáforos para lidar com deadlocks.

Relação com Outras Disciplinas

- O bom desempenho do aluno nesta disciplina depende do embasamento teórico e prático do aluno adquirido nas disciplinas relacionadas a Sistemas de Computação, incluindo: Sistema Operacional, Redes e Sistemas Distribuídos.

Relação com Outras Disciplinas

- Além disto, é essencial que o aluno tenha habilidade de programação (Introdução à Programação, Algoritmos: Fundamentos e Estruturas de Dados).
- É desejável ainda o conhecimento de Arquitetura de Software e Método de Desenvolvimento de Software.

Programa

I - Caracterização de software concorrente: conceitos, desafios, ferramentas.

II - Programação concorrente orientada a objetos: programação concorrente, processos e threads, modelo de objetos e concorrência, safety, liveness, execução de construções concorrentes, reusabilidade, adaptadores, padrões de projeto.

Programa

III - Exclusão mútua: aplicações, construção, sincronização, deadlock, modelo de memória do Java, exclusão em métodos e threads, mutexes, read-write locks.

IV - Dependência: exceções, guardas e variáveis de condição, monitores, espera ocupada, semáforos, criando transações, protocolos acquire-release.

Programa

V - Criando programas concorrentes baseados em threads: modelagem, threads trabalhadoras, polling e eventos, callbacks, fork/join.

Critério de Avaliação

- Três componentes integram a avaliação:
  - (A) Média das atividades em sala de aula, laboratório e extra-classe.
  - (P1 e P2) 1a. e 2a. Provas individuais.
  - (PA) Projeto de Aplicação
- As notas das Atividades, Provas e do Projeto de Aplicação terão valor de zero a dez.

Critério de Avaliação

- O Projeto de Aplicação trata-se de um aplicação completa que deverá ser desenvolvida pelos alunos. Eles serão divididos em equipes que farão a especificação e a implementação.
- A NF será calculada conforme fórmula abaixo:
  - NF =(A*0.1) + (P1*0.3) + (P2*0.3) + (PA*0.3)

Critério de Avaliação

Observações:

- Cada avaliação vale dez pontos. Não haverá avaliação substitutiva, exceto nos casos previstos no regulamento da UFG.
- Cada trabalho será previamente definido pelo docente em sala de aula especificando o escopo do trabalho, a composição do grupo (se for o caso de trabalho em grupo), e os critérios de correção.

Critério de Avaliação

- Nas atividades em grupo, poderão ser atribuídas notas diferentes para os integrantes de um mesmo grupo, se forem observadas diferenças nos esforços e resultados produzidos por esses integrantes.
- Nenhum trabalho será recebido em atraso, salvo algum caso previsto no regulamento da universidade

Critério de Avaliação

- Em qualquer atividade ou produto avaliado, a ocorrência de plágio leva à atribuição da nota zero. O aluno deve se familiarizar com os mecanismos de citação de trabalhos alheios para evitar riscos de plágio por descuido no reconhecimento de autoria de trabalhos ou ideias citadas. O mesmo tratamento será dado para cópias entre os próprios alunos da turma.
- As atividades serão recebidas pelo Moodle do INF quando realizadas em laboratório.

Critério de Avaliação

- As provas serão individuais e cobrirão o conteúdo desenvolvido até a data de sua aplicação.
- Será atribuída a nota 0,0 (zero) a qualquer atividade ou trabalho não realizado ou não entregue na data estipulada.

Critério de Avaliação

- O pedido de segunda chamada deverá ser protocolado no prazo máximo de 3 (três) dias úteis após a realização da prova, apresentando a comprovação da impossibilidade de seu comparecimento à primeira chamada da prova conforme condições estipuladas na Resolução CONSUNI Nº 06/2002.
- **Será considerado aprovado(a) o aluno(a) que obtiver NOTA FINAL igual ou superior a 6.0 (seis pontos), além de frequência igual ou superior a 75% das aulas (48 horas)**

Previsão de Realização das Avaliações

- A1. 02/06/2016
- A2. até 14/07/2015

- *Prova substitutiva: se o aluno perder avaliação poderá fazê-la no dia 21 de julho de 2016, sem que seja necessário pedir segunda chamada.*

Bibliografia Básica

- Concurrent Programming in Java: Design Principles and Patterns, Douglas Lea, Addison-Wesley, 3rd edition, 2006; Pattern-Oriented Software Architecture Volume 2: Patterns for Concurrent and
- Networked Objects, Douglas Schmidt et al., Wiley, 2000; Pattern-Oriented Software Architecture Volume 3: Patterns for Resource Management, Michael Kircher, Wiley, 2004;

Bibliografia Complementar

Java Concurrency in Practice, Brian Goetz et al., Addison-Wesley, 2006

Bibliografia Sugerida

DEITEL, H. M.; DEITEL, P. J. Java: Como Programar. 8. Ed. São Paulo: Pearson Education, 2010.