



UNIVERSIDADE FEDERAL DE UBERLÂNDIA
 FACULDADE DE COMPUTAÇÃO
 COLEGIADO DO CURSO DE SISTEMAS DE INFORMAÇÃO

FICHA DE DISCIPLINA

DISCIPLINA: PROGRAMAÇÃO ORIENTADA A OBJETOS 2

CÓDIGO: GSI020

UNIDADE ACADÊMICA:

FACULDADE DE COMPUTAÇÃO

PERÍODO/SÉRIE: 4º

CH TOTAL
TEÓRICA:

CH TOTAL
PRÁTICA:

CH TOTAL:

OBRIGATORIA: (X)

OPTATIVA: ()

30

30

60

OBS:

PRÉ-REQUISITOS: Programação Orientada
a Objetos 1

CÓ-REQUISITOS: NÃO HÁ

OBJETIVOS

(Após o curso o aluno estará apto a) Aplicar técnicas avançadas de análise e projeto empregadas no desenvolvimento de *software*, enfatizando formas de melhorar o reuso de *software* através do paradigma de Orientação a Objetos. Especificamente:

- compreender o papel dos padrões na reutilização de colaborações entre classes e objetos em modelos conceituais e modelos de *software*;
- projetar uma arquitetura de *software* usando padrões arquiteturais;
- aplicar padrões de projeto, principalmente aqueles mais importantes no desenvolvimento de *frameworks*;
- analisar os padrões de projeto mais apropriados ao *software* a ser desenvolvido;
- compreender alguns padrões típicos de análise, i.e., modelos conceituais de objetos reutilizáveis;
- compreender anti-padrões;
- compreender os conceitos de *frameworks* e como eles permitem reutilizar a análise de problemas e o projeto de soluções, permitindo assim escrever aplicações relacionadas com eficácia;
- analisar *frameworks* concretos;
- compreender uma metodologia de desenvolvimento de *frameworks*;
- desenvolver *software* usando as técnicas avançadas de análise e projeto de *software*.

EMENTA

Padrões de análise. Projeto de software orientado a objetos. Arquitetura de software. Projeto detalhado de software. Princípios de projeto orientado a objetos. Tecnologia de componentes. Padrões arquiteturais. Padrões de projeto. Desenvolvimento de software orientado a aspectos.

[Handwritten signature]

Tópicos avançados em projeto de software.

DESCRIÇÃO DO PROGRAMA

1 – Programação genérica com classes e métodos genéricos

- 1.1 – Introdução
- 1.2 - Motivação para métodos genéricos
- 1.3 - Métodos genéricos: implementação e tradução em tempo de compilação
- 1.4 - Métodos que utilizam um parâmetro de tipo como tipo de retorno
- 1.5 - Sobrecarregando métodos genéricos
- 1.6 - Classes genéricas
- 1.7 - Tipos brutos
- 1.8 - Curingas em métodos que aceitam parâmetros de tipo
- 1.9 - Genéricos e herança
- 1.10 - Conclusão

2 – Multithreading

- 2.1 – Introdução
- 2.2 - Estados de uma thread
- 2.3 - Prioridades
- 2.4 - Criando e executando threads
- 2.5 - Sincronização de threads
- 2.6 - Relacionamento produtor / consumidor
- 2.7 - Multithreading com GUI
- 2.8 - Interfaces Callable e Future
- 2.9 - Aplicações de threads

RMI – Remote Method Invocation

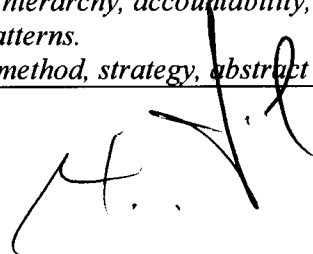
- 3.1 – Introdução à invocação remota de métodos
- 3.2 – Conceitos Básicos
- 3.3 – Camadas do RMI
- 3.4 – Serviços de registro e naming
- 3.5 – Aplicações de RMI

Frameworks para elaboração de MVC, acesso a dados, testes e geração de relatórios

- 4.1 – Struts
- 4.2 - Hibernate
- 4.3 - JUnit
- 4.4 - Jasper Report / iReport

5 – Princípios e Padrões de análise, arquitetura e projeto de Software.

- 5.1 - Princípios de *design* de classes: (SRP) *Single responsibility principle*; (OCP) *The open-closed principle*; (LSP) *The liskov substitution principle*; (DIP) *The dependency inversion principle*; (ISP) *The interface segregation principle*.
- 5.2 - Princípios de coesão e acoplamento de pacotes: (REP) *The reuse/release equivalency principle*; (CCP) *The common closure principle*; (CRP) *The common reuse principle*; (ADP) *The acyclic dependencies principle*; (SDP) *The stable dependencies principle*; (SAP) *The stable abstraction principle*.
- 5.3 - Principais padrões de análise: *party, organization hierarchy, accountability, knowledge level, quantity, range, temporal patterns, accounting patterns*.
- 5.4 - Principais padrões de projeto: *observer, template method, strategy, abstract*



factory, builder, iterator, composite, decorator, facade, adapter, proxy, singleton, factory method, visitor, bridge, mediator, command, flyweight;

5.5 - Principais padrões arquiteturais: *layer, microkernel, MVC, black board, broker;*

6. Orientação a aspectos. Apresentar o desenvolvimento de software orientado a aspectos.

6.1- Limitações da orientação a objetos: entrelaçamento e espalhamento de código;

6.2 - Definição de *pointcuts* e *advices*;

6.3 - Implementação de aspectos: *aspectJ*;

6.4- Exemplos de uso de aspectos para melhoria de modularidade em sistemas;

6.5 - Conceituação de aspectos: *concerns, scattering, tangling, weaving*; Modelagem e captura de aspectos com casos de uso;

6.6 - Manutenção separada de aspectos com módulos de casos de uso; Estabelecimento de arquitetura de software baseada em casos de uso e aspectos;

6.7- UML: modelagem de aspectos e casos de uso *slices*;

6.8 - Padrões de uso de aspectos.

7 - Estudo de caso.

7.1- Desenvolver uma aplicação empregando adequadamente princípios e padrões, *multithreading*, RMI e *frameworks* apresentados.

7.2 - Utilizar persistência em bancos de dados.

7.3 - Documentar adequadamente todas as fases do desenvolvimento em UML.

BIBLIOGRAFIA

Básica

DEITEL, Harvey M. **Java: Como Programar**, 8ed. São Paulo: Prentice Hall, 2010.

FREEMAN, Eric; FREEMAN, Elisabeth. **Use a cabeça! padrões de projeto**. Rio de Janeiro: Atlas Books, 2005.

JACOBSON, Ivar. **Aspect-oriented software development with use cases**. Upper Saddle River: Addison-Wesley, 2004.

Complementar

GAMMA, Erich et al. **Design patterns**. Upper Saddle River: Addison Wesley, 1995.

LADDAD, Ramnivas. **Aspect in action: practical aspect-oriented programming**. Greenwich: Manning, 2003.

KRUCHTEN, Philippe. **The rational unified process: an introduction**. 3rd ed. Boston: Addison Wesley, 2003.

KRUCHTEN, Philippe. **The rational unified process made easy: practitioner's guide to the rup**. Boston: Addison Wesley, 2005.

LARMAN, Craig. **Utilizando UML e padrões: uma introdução a análise e ao projeto orientados a objetos**. Porto Alegre: Bookman, 2000.

SOMMERVILLE, I. **Engenharia de software**. 8th. ed. Harlow: Addison Wesley, 2007.

Martin Fowler, **Refactoring: Improving the Design of Existing Code**, Addison-Wesley, 2000.

Antonio Mendes, **Arquitetura de Software**, Ed. Campus, 2002

C. Szyperski, **Component Software: Beyond Object-Oriented Programming**, Addison-Wesley, 1998.

LARMAN, C. **Utilizando UML e Padrões: Uma Introdução à Análise e ao Projeto Orientado a Objetos**. Porto Alegre: Bookmann, 2001.



FOWLER, M. Scott, K. **UML Essencial**. POA: Bookmann, 2000.
ECKEL B.: **Thinking in Java**, Prentice Hall, Segunda Edição, 2000.

APROVAÇÃO

30/08/2010

Carimbo e assinatura do Coordenador do curso

Universidade Federal de Uberlândia
Prof. Marcelo Rodrigues de Sousa
Coordenador do Curso de Sistemas de Informação
da Faculdade de Computação-Portaria R nº 812/08

30/08/2010

Carimbo e assinatura do Diretor da
Unidade Acadêmica

Universidade Federal de Uberlândia
Prof. Jamil Salom Barbar
Diretor da Faculdade de Computação
Portaria R nº 672/07