



UNIVERSIDADE FEDERAL DE UBERLÂNDIA

FICHA DE COMPONENTE CURRICULAR

<b>CÓDIGO:</b> GSI516	<b>COMPONENTE CURRICULAR:</b> Programação Orientada a Objetos II	
<b>UNIDADE ACADÊMICA OFERTANTE:</b> Faculdade de Computação		<b>SIGLA:</b> FACOM
<b>CH TOTAL TEÓRICA:</b> 30	<b>CH TOTAL PRÁTICA:</b> 30	<b>CH TOTAL:</b> 60

OBJETIVOS

Capacitar o aluno 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;
- compreender e aplicar Padrões de Projeto
- analisar e decidir 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 particularidades de linguagens de Programação Orientada a Objetos;
- 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 e utilizar *frameworks* reais;
- compreender uma metodologia de desenvolvimento de *frameworks*;
- desenvolver *software* usando as técnicas avançadas de análise e projeto de *software*.

EMENTA

Padrões de projeto. Padrões de análise. Arquitetura de software e padrões arquiteturais. Princípios de projeto orientado a objetos. Projeto de software orientado a objetos. Projeto detalhado de software. Paradigma de desenvolvimento de software orientado a aspectos. Tópicos avançados em projeto de software orientado a objetos.



## PROGRAMA

### 1 – Programação genérica com classes e métodos genéricos

- - Introdução
- - Métodos genéricos: implementação e tradução em tempo de compilação
- - Métodos que utilizam uma variável de tipo como tipo de retorno
- - Sobrecarga de métodos genéricos
- - Classes genéricas
- - Tipos brutos (*raw types*)
- - Curingas (*wildcards*)
- - Genéricos e Herança

### 2 – Multithreading

- - Introdução
- - Estados de uma thread
- - Prioridades
- - Criando e executando threads
- - Sincronização de threads
- - Relacionamento produtor / consumidor
- - Multithreading com GUI
- - Interfaces Callable e Future
- - Aplicações de threads

### 3 – Estudo de Frameworks reais e suas aplicações

- 3.1 - Struts
- 3.2 - Hibernate
- 3.3 - Junit
- 3.4 - Jasper Report / iReport

### 4 - Princípios e Padrões de análise, arquitetura e projeto de Software

- 4.1 - Princípios de projeto 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*.
- 4.2 - 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*.
- 4.3 - Padrões de análise: *party, organization hierarchy, accountability, knowledge level, quantity, range, temporal patterns, accounting patterns*.
- 4.4 - Principais padrões de projeto: *observer, template method, strategy, abstract factory, builder, iterator, composite, decorator, façade, adapter, proxy, singleton, factory method, visitor, bridge, mediator, command, flyweight*.
- 4.5 - Principais padrões arquiteturais: *layer, microkernel, MVC, black board, broker*;



#### **5 - Introdução a orientação a aspectos**

- 5.1- Limitações da orientação a objetos: entrelaçamento e espalhamento de código;
- 5.2 - Definição de *pointcuts* e *advices*;
- 5.3 - Implementação de aspectos: *aspectJ*;
- 5.4- Exemplos de uso de aspectos para melhoria de modularidade em sistemas;
- 5.5 - Conceituação de aspectos: *concerns*, *scattering*, *tangling*, *weaving*;
- 5.6 - Manutenção separada de aspectos com módulos de casos de uso; Estabelecimento de arquitetura de software baseada em casos de uso e aspectos;
- 5.7 - Padrões de uso de aspectos.

#### **6 - Estudo de caso.**

- 6.1- Desenvolver uma aplicação empregando adequadamente princípios, padrões, técnicas e conceitos apresentados ao longo da disciplina.
- 6.2 - Utilizar *frameworks*.

### **BIBLIOGRAFIA BÁSICA**

BAUER, C.; KING, G. **Java Persistence com Hibernate**. [S. l]: Ciência Moderna, 2007.

DEITEL, H. M. **Java: como programar**, 8. ed. São Paulo: Prentice Hall, 2010.

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

### **BIBLIOGRAFIA COMPLEMENTAR**

ECKEL B. **Thinking in Java**. 2. ed. São Paulo: Prentice Hall, 2000.

EVANS, E. **Domain-Driven Design Atacando As Complexidades na Criação do Software**. Rio de Janeiro: Alta Books, 2009.

GAMMA, E. **Design patterns**. Upper Saddle River: Addison Wesley, 1995.

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

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



SERVIÇO PÚBLICO FEDERAL  
MINISTÉRIO DA EDUCAÇÃO  
UNIVERSIDADE FEDERAL DE UBERLÂNDIA



APROVAÇÃO

14 / 03 / 14

7194

Carimbo e assinatura do Coordenador do Curso  
UNIVERSIDADE FEDERAL DE UBERLÂNDIA  
FACULDADE DE COMPUTAÇÃO  
Prof. Dr. Kil Jin Brandini Park  
Coordenador do Curso de Sistema de Informação  
Monte Carmelo - Portaria R Nº 523/13

14 / 03 / 14

Carimbo e assinatura do Diretor da  
Unidade Acadêmica  
(que oferece o componente curricular)

Universidade Federal de Uberlândia  
Prof. Imeio Reis da Silva  
Diretor da Faculdade de Computação  
Portaria R Nº. 757/11