


UNIVERSIDADE FEDERAL DE UBERLÂNDIA
Faculdade de Computação

Av. João Naves de Ávila, 2121, Bloco 1B - Bairro Santa Mônica, Uberlândia-MG, CEP 38400-902
 Telefone: (34) 3239-4393 - www.facom.ufu.br - facom@ufu.br


PLANO DE ENSINO
1. IDENTIFICAÇÃO

Componente Curricular:	PROGRAMAÇÃO ORIENTADA A OBJETOS 2								
Unidade Ofertante:	FACULDADE DE COMPUTAÇÃO (FACOM)								
Código:	GSIO20		Período/Série:		4°		Turma:	S	
Carga Horária:						Natureza:			
Teórica:	30	Prática:	30	Total:	60	Obrigat ória:	(x)	Optativ a:	()
Professor(A):	João Henrique de Souza Pereira					Ano/Semestre:		2026/2	
Observações:									

2. 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.

3. JUSTIFICATIVA

Programação Orientada a Objetos é um conteúdo para a formação do Bacharel em Sistemas de Informação, e contribui para o discente aplicar tais conceitos em aplicações e estudos de casos. O mercado busca desenvolvedores com formação ampla e versados em técnicas e paradigmas variados, capazes de avaliar problemas, manipular ferramentas que auxiliem projetos de software e que consigam tomar boas decisões em projeto de software. O paradigma de Orientação a Objetos é encontrado no mercado de desenvolvimento de software e na academia, sendo assim, um tema que faz parte a formação do Bacharel em Sistemas de Informação.

4. OBJETIVO

Objetivo Geral:
Aplicar técnicas de análise e projeto empregadas no desenvolvimento de software, enfatizando formas de melhorar o reuso de software através do paradigma de Orientação a Objetos.
Objetivos Específicos:
- 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 metodologia de desenvolvimento de frameworks;
- desenvolver software usando as técnicas avançadas de análise e projeto de software.

5. 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

3- 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

4- 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 openclosed 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, façade, 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.

6. METODOLOGIA

As aulas serão ministradas às terças-feiras, das 19:00 às 20:40, e às quintas-feiras, das 20:50 às 22:00. O curso será desenvolvido por meio de apresentações expositivas dialogadas, teóricas e práticas e desenvolvimento de trabalhos sobre o tema. Para a exposição, serão utilizados slides, com apoio do quadro, em conjunto com a exposição oral. A apresentação será complementada, sempre que necessário, com anotações e demonstrações no quadro da sala.

Como recursos didáticos serão usados laptops, lousa branca/vidro, lousa de giz, datashow e atividades práticas.

A comunicação entre professor e alunos se dará pelo chat geral e particular da plataforma Microsoft Teams, bem como nos horários de atendimento presencial, que serão realizados às quintas-feiras, das 19:50 às 20:50 e aos sábados das 14:00 às 15:00, na sala 1B132, com agendamento prévio em sala de aula ou pelo Teams. No chat geral serão postados comunicados para todos os alunos, avisos sobre enunciados de atividades e comunicações diversas. No chat particular e durante o horário de atendimento presencial serão resolvidas dúvidas pontuais com discentes, vistas de atividades avaliativas e outros assuntos diversos sobre a disciplina que o discente prefira conversar. O atendimento aos discentes será combinado de forma a atender aos horários disponíveis de ambas as partes, ocorrendo de forma presencial.

Serão ao todo quinze terças-feiras e quinze quintas-feiras, totalizando 60 horas-aula presenciais. As 12 horas-aula restantes serão contabilizadas por meio da realização de atividades complementares distribuídas ao longo do semestre.

6.1. Cronograma

Semana	Módulo	Atividades Presenciais	Carga Horária Presencial	Data e Horário de Atividades Presenciais	Atividades Complementares	Carga Horária Atividades Complementares
01/09/26	Início Semestre					
1	Apresentação da disciplina. Método de Avaliação. Organização dos grupos.	Aula presencial expositiva. Discussões e debates sobre o módulo.	4 horas-aula	01/09/2026 20:50h 03/09/2026 19:00h		
2	1- Programação genérica com classes e métodos genéricos	Aula presencial expositiva. Discussões e debates sobre o módulo.	4 horas-aula	08/09/2026 20:50h 10/09/2026 19:00h	Estudo dirigido e pesquisas complementares em programação orientada a objetos	4 horas-aula
3	2- Multithreading	Aula presencial expositiva. Discussões e debates sobre o módulo.	4 horas-aula	15/09/2026 20:50h 17/09/2026 19:00h		
4	3- RMI - Remote Method Invocation	Aula presencial expositiva. Discussões e debates sobre o módulo.	2 horas-aula	24/09/2026 20:50h		
5	3- RMI - Remote Method Invocation 4- Frameworks para elaboração de MVC, acesso a dados, testes e geração de relatórios	Aula presencial expositiva. Discussões e debates sobre o módulo.	4 horas-aula	29/09/2026 20:50h 01/10/2026 19:00h		
6	4- Frameworks para elaboração de MVC, acesso a dados, testes e geração de relatórios	Aula presencial expositiva. Discussões e debates sobre o módulo.	4 horas-aula	06/10/2026 20:50h 08/10/2026 19:00h		

	5- Princípios e Padrões de análise, arquitetura e projeto de Software.					
7	5- Princípios e Padrões de análise, arquitetura e projeto de Software.	Aula presencial expositiva. Discussões e debates sobre o módulo.	2 horas-aula	13/10/2026 20:50h	Estudo dirigido e pesquisas complementares em programação orientada a objetos	4 horas-aula
8	6- Orientação a aspectos. Apresentar o desenvolvimento de software orientado a aspectos. Facom Techweek	Aula presencial expositiva. Discussões e debates sobre o módulo.	4 horas-aula	20/10/2026 20:50h 22/10/2026 19:00h		
9	6- Orientação a aspectos. Apresentar o desenvolvimento de software orientado a aspectos. 7- Estudo de caso.	Aula presencial expositiva. Discussões e debates sobre o módulo.	4 horas-aula	27/10/2026 20:50h 29/10/2026 19:00h		
10	Maratona SBC de Programação (Final Brasileira na Facom-UFU)	Aula presencial expositiva. Discussões e debates sobre o módulo.	4 horas-aula	03/11/2026 20:50h 05/11/2026 19:00h	Estudo dirigido e pesquisas complementares em programação orientada a objetos	4 horas-aula
11	7- Estudo de caso. 8- Tema de escolha do grupo (moderno e dentro do escopo da disciplina).	Aula presencial expositiva. Discussões e debates sobre o módulo.	4 horas-aula	10/11/2026 20:50h 12/11/2026 19:00h		
12	8- Tema de escolha do grupo	Aula presencial	4 horas-aula	17/11/2026 20:50h		

	(moderno e dentro do escopo da disciplina). Apresentação de trabalhos em grupo	expositiva. Discussões e debates sobre o módulo.		19/11/2026 20:50h		
13	Apresentação de trabalhos em grupo	Aula presencial expositiva. Discussões e debates sobre o módulo.	4 horas-aula	24/11/2026 20:50h 26/11/2026 19:00h		
14	Avaliação prática individual	Aula presencial expositiva. Discussões e debates sobre o módulo.	4 horas-aula	01/12/2026 20:50h 03/12/2026 19:00h		
15	Resolução e vista da avaliação	Aula presencial expositiva. Discussões e debates sobre o módulo.	4 horas-aula	08/12/2026 20:50h 10/12/2026 19:00h		
16	Atividade avaliativa de recuperação Resolução e vista da atividade avaliativa de recuperação	Aula presencial expositiva. Discussões e debates sobre o módulo.	4 horas-aula	15/12/2026 20:50h 17/12/2026 19:00h		
07/08/2026	Término Semestre	Total de horas-aula presenciais:	60		Total de horas-aula de atividades complementares:	12
Carga Horária Total (presencial + atividades extraclasse):						72 horas-aula

Atendimento ao discente: O atendimento aos alunos ocorrerá semanalmente às quintas-feiras, das 19:50 às 20:50 e aos sábados das 11:00 às 12:00, na sala 1B132. É necessário agendamento prévio em sala de aula ou pelo chat particular da plataforma Microsoft Teams. O horário de atendimento poderá ser flexibilizado e ajustado em conjunto com a turma, de forma a atender aos horários disponíveis de ambas as partes. Além do atendimento presencial, a comunicação entre professor e alunos também ocorrerá de forma assíncrona pelo chat geral e particular da plataforma Microsoft Teams. No chat geral serão postados comunicados para todos os discentes, avisos sobre enunciados de tarefas, resolução de exercícios e avisos diversos. No chat particular serão resolvidas dúvidas pontuais, vistas de atividades avaliativas e outros assuntos que o discente prefira tratar em particular.

7. AVALIAÇÃO

A avaliação será feita por meio de dois trabalhos avaliativos (TA1 e TA2) em grupo e uma prova prática individual (P1) sobre o conteúdo ministrado, a serem realizadas de forma presencial em sala de aula. A prova terá duração de 1h20min e abrangerá o conteúdo trabalhado em sala de aula e em laboratório. A prova P1 valerá 30 pontos. Os trabalhos consistirão em apresentações teórica e prática sobre o programa deste componente curricular, com temas do programa da ficha do componente curricular da disciplina e valerá 35 pontos cada, sendo apresentado em sala de aula. A nota final será dada pelo somatório dos trabalhos e da prova avaliativa, conforme a fórmula:

$$NF = TA1 + TA2 + P1$$

De acordo com o Art. 141 das Normas de Graduação (Res. CONGRAD Nº 46/2022), haverá uma avaliação de recuperação de aprendizagem, que terá valor de 100 pontos. A mesma tem caráter substitutivo, ou seja, a pontuação obtida pelo aluno na avaliação de recuperação substituirá a nota final do semestre, caso seja maior que a mesma. A prova de recuperação abrangerá todo o conteúdo visto no semestre. Ainda, de acordo com o Art. 141, somente fará jus ao direito de realizar a avaliação de recuperação o(a) discente que não obtiver o rendimento mínimo de aprovação (60 pontos) e que possuir no mínimo 75% de frequência na disciplina. Em nenhuma hipótese a nota final após a recuperação será inferior à nota obtida durante o semestre. A nota final do discente, na prova de recuperação, será saturada em 60 pontos.

Adicionalmente, será aplicada prova substitutiva aos alunos que não realizarem a P1 e apresentarem justificativa formal de ausência. A nota obtida substituirá a avaliação não realizada e a prova substitutiva valerá os mesmos 30 pontos da prova original. Também caso algum discente não compareça (inclui atraso ou sair mais cedo) na data de sua apresentação de trabalho avaliativo, ele poderá fazer a P1 com adição da pontuação que não obteve, devido sua ausência.

8. BIBLIOGRAFIA

Básica

- 1) DEITEL, Harvey M. Java: Como Programar, 8ed. São Paulo: Prentice Hall, 2010.
- 2) FREEMAN, Eric; FREEMAN, Elisabeth. Use a cabeça! padrões de projeto. Rio de Janeiro: Atlas Books, 2005.
- 3) JACOBSON, Ivar. Aspect-oriented software development with use cases. Upper Saddle River: Addison-Wesley, 2004.

Complementar

- 1) GAMMA, Erich et al. Design patterns. Upper Saddle River: Addison Wesley, 1995.
- 2) LADDAD, Ramnivas. Aspect in action: practical aspect-oriented programming. Greenwich: Manning, 2003.
- 3) KRUCHTEN, Philippe. The rational unified process: an introduction. 3rd ed. Boston: Addison Wesley, 2003.
- 4) KRUCHTEN, Philippe. The rational unified process made easy: practitioner's guide to the rup. Boston: Addison Wesley, 2005.
- 5) LARMAN, Craig. Utilizando UML e padrões: uma introdução a análise e ao projeto orientados a objetos. Porto Alegre: Bookman, 2000.
- 6) SOMMERVILLE, I. Engenharia de software. 8th. ed. Harlow: Addison Wesley, 2007.
- 7) FOWLER, Martin, Refactoring: Improving the Design of Existing Code, Addison-Wesley, 2000.

- 8) MENDES, Antonios, Arquitetura de Software, Ed. Campus, 2002.
- 9) SZYPERSKI, C., Component Software: Beyond Object-Oriented Programming, Addison-Wesley, 1998.
- 10) LARMAN, C. Utilizando UML e Padrões: Uma Introdução à Análise e ao Projeto Orientado a Objetos. Porto Alegre: Bookmann, 2001.
- 11) FOWLER, M. Scott, K. UML Essencial. POA: Bookmann, 2000.
- 12) ECKEL, B.: Thinking in Java, Prentice Hall, Segunda Edição, 2000.

9. **APROVAÇÃO**

Aprovado em reunião do Colegiado realizada em: ____/____/____

Coordenação do Curso de Graduação: _____