Programação Orientada a Objetos

CST em Análise e Desenvolvimento de Sistemas

Prof. Emerson Ribeiro de Mello

mello@ifsc.edu.br

INSTITUTO FEDERAL
Santa Catarina
Campus São José

Licenciamento

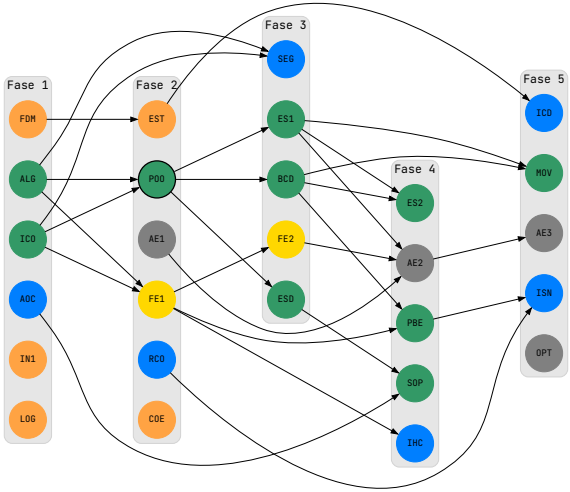
Slides licenciados sob Creative Commons “Atribuição 4.0 Internacional”

Sumário

Apresentação da disciplina

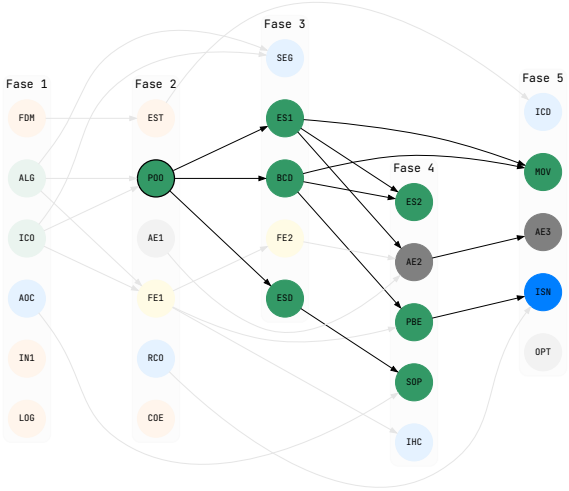
Relação com outras unidades curriculares

https://sigaa.ifsc.edu.br/sigaa/link/public/curso/curriculo/54017903



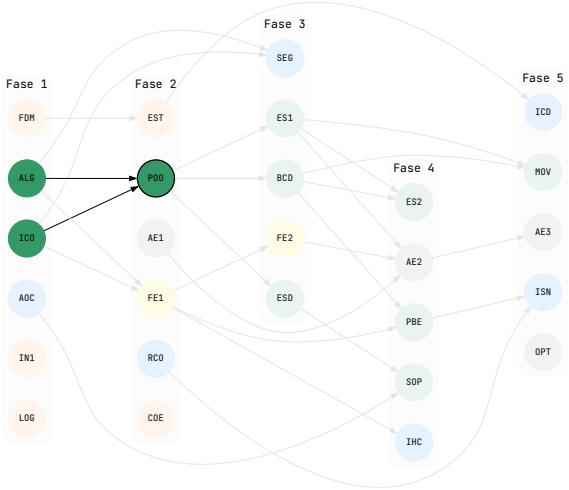
Relação com outras unidades curriculares

https://sigaa.ifsc.edu.br/sigaa/link/public/curso/curriculo/54017903



Relação com outras unidades curriculares

https://sigaa.ifsc.edu.br/sigaa/link/public/curso/curriculo/54017903



Pré-requisitos dessa disciplina

Introdução à computação (ICO)

1. Saber instalar e configurar programas no Linux
2. Saber utilizar o terminal e os comandos básicos do Linux
3. Saber lidar com arquivos, diretórios, caminhos e variáveis de ambiente
4. Saber utilizar o Git no terminal (*init, add, commit, branch, merge, checkout*)
5. Saber utilizar o GitHub (*push, pull, clone*) e autenticação via PAT
6. Saber navegar na Internet pelo computador e usar email de forma efetiva
7. Saber usar o SIGAA

Pré-requisitos dessa disciplina

Pensamento computacional e algoritmos (ALG)

1. Saber criar algoritmos utilizando fluxogramas e pseudocódigo
2. Saber realizar teste de mesa
3. Saber criar algoritmos utilizando a linguagem de programação Java
   - estruturas de decisão e repetição
   - vetores e matrizes
   - leitura de dados do teclado e escrita em tela

Ementa

Introdução ao paradigma da programação orientada a objetos; Introdução à linguagem de programação Java; Bibliotecas da linguagem e de terceiros; Ferramentas para desenvolvimento e automatização de projetos; Empacotamento e distribuição de aplicações Java para desktop; Linguagem de modelagem unificada (UML) e ferramentas para confecção de diagramas de classe.

Objetivos da disciplina

- Usar de forma efetiva ferramentas como ambiente integrado de desenvolvimento e sistema de controle de versão para trabalhar de forma colaborativa
- Desenvolver software de média complexidade na linguagem Java e de acordo com o paradigma da programação orientada a objetos
- Criar diagramas de classes usando a linguagem de modelagem unificada (UML), para fins de modelagem e documentação de software

Metodologia

- Aulas expositivas-dialogadas e práticas em laboratório sob a supervisão do professor
- Metodologia de aprendizado baseado em projetos, em que desafios são lançados e o docente orienta os estudantes em suas soluções

Horários

- **Aulas:** Laboratório de Redes

  - 20:40 – 22:30 - segunda-feira
  - 20:40 – 22:30 - quinta-feira

- **Atendimento extraclasse:** Sala de Professores de Tele I

  - 13:30 – 15:30 - terça-feira

Interações do professor com a turma será por meio do SIGAA ou email

Instrumentos de avaliação I

Atividade	Quantidade	Sigla	Peso na média
Laboratórios	n	a	20%
Listas de exercícios	5	e	80%

- **Laboratórios** são pequenas atividades para fixação de conteúdo e deverão ser entregues na aula onde foram propostos ou em data a ser definida pelo docente
- **Listas de exercícios** envolverão mais de um tópico e deverão ser entregues na mesma aula em que foram propostas

Instrumentos de avaliação II

- Para a aprovação o discente deverá possuir no mínimo 75% de presença e Conceito Final ≥ 6, sendo este calculado por meio da Equação 1
- As listas de exercícios possuem pesos diferentes, sendo estes: $W = \{w_1, w_2, w_3, w_4, w_5\} = \{1, 1, 2, 3, 3\}$

$$CF = \left\lfloor \left(\frac{\sum_{i=1}^{n} a_i}{n} \right) \times 0{,}2 + \left(\frac{\sum_{i=1}^{5} e_i \times w_i}{\sum_{i=1}^{5} w_i} \right) \times 0{,}8 \right\rceil, \quad CF \in \mathbb{N}. \tag{1}$$

Sobre a recuperação de estudos

- **Laboratórios** serão apresentados como forma de avaliação contínua e cumulativa, assim, não haverá recuperação para os mesmos
- **Listas de exercícios** entregues poderão ser recuperadas[1]. Notas de listas não entregues não poderão ser recuperadas
  - No final do semestre será proposta uma atividade de recuperação para os discentes que não atingiram a média mínima para aprovação

[1] Salvo os casos previstos no artigo 162 do Regulamento Didático-Pedagógico (RDP) do IFSC, no RDP não está previsto a segunda chamada, situação que ocorre quando o discente não faz a atividade avaliativa na data estabelecida.

Fraude no processo avaliativo

- O **plágio é estritamente proibido**, seja ao copiar trabalhos de colegas, de repositórios *online* ou ao utilizar ferramentas de Inteligência Artificial (e.g. Copilot, ChatGPT etc) para obter a solução completa ou parcial de atividades avaliativas.
- **Caso seja identificado plágio ou fraude, o discente receberá nota 0** na atividade em questão, sem direito à recuperação. Além disso, o caso será encaminhado à coordenação do curso para registro no histórico escolar do discente e aplicação das medidas disciplinares cabíveis.

Bibliografia básica

- BEZERRA, Eduardo. **Princípios de análise e projeto de sistemas com UML**. Campus, 2002.
- HORSTMANN, Cay S.; CORNELL, Gary. **Core Java – Volume I – Fundamentos**. 8. ed.: Pearson, 2010.

Bibliografia complementar

- DEITEL, H.M.; DEITEL, P.J. **Java Como Programar**. 8. ed.: Prentice Hall, 2010.
- LARMAN, Craig. **Utilizando UML e padrões**. Bookman, 2007. https://app.minhabiblioteca.com.br/books/9788577800476/.
- SCHILDT, Herbert. **Java para iniciantes: crie, compile e execute programas Java rapidamente**. Bookman, 2015. ISBN 9788582603376. Disponível em: https://app.minhabiblioteca.com.br/#/books/9788582603376. Acesso em: 7 dez. 2023.

Ferramentas para essa disciplina

Ferramentas para essa disciplina

Kit de desenvolvimento Java – JDK

Kit de desenvolvimento Java

Java Development Kit (JDK)

- **Java** é uma linguagem de programação de propósito geral
- Para desenvolver aplicações em Java é necessário ter instalado o **JDK**
- JDK contém
  - Ambiente de execução Java (JRE, *Java Runtime Environment*)
  - Compilador, bibliotecas e outras ferramentas para desenvolvimento
- **Na disciplina será usado o JDK 21 LTS**[2]
  - No Linux ou macOS, instale via SDKman (https://sdkman.io)
  - Ou baixe em https://adoptium.net e instale manualmente

[2] https://whichjdk.com

Ferramentas para essa disciplina

Sistema de automação de compilação de projetos

Gradle, sistema de automação de projetos

Qual a necessidade para projetos Java?

- É possível criar uma aplicação Java simples sem usar ferramentas como o gradle, porém para aplicações complexas a tarefa não é trivial

Como instalar o gradle `8.x`?

- **Requisito:** JDK 17 ou superior
- Windows – baixar no site oficial[3]
- Linux ou macOS – via SDKman[4]
  - `sdk install gradle`
  - Linux – Opte por instalar via SDKMan, pois no apt-get a versão está defasada

[3] https://docs.gradle.org/current/userguide/installation.html
[4] https://sdkman.io

Ferramentas para essa disciplina

Ambiente integrado de desenvolvimento

Ambiente integrado de desenvolvimento

***Integrated Development Environment* (IDE)**

Jetbrains IntelliJ IDEA

- IDE multiplataforma completa para Java e Kotlin
- Versão *Community* é gratuita para todos e suficiente para essa disciplina
- Licença gratuita para estudantes do IFSC para versão *Ultimate*, clique aqui
- Contém JDK, gradle, suporte a git etc

Ambiente integrado de desenvolvimento

***Integrated Development Environment* (IDE)**

Microsoft Visual Studio Code

- Editor de propósito geral que pode ser personalizado por extensões para ter suporte a diferentes linguagens de programação
- Extensões necessárias para desenvolver em Java
  - Extension Pack for Java
  - Gradle for Java
- **Requisito:** JDK e gradle instalados no sistema operacional

Uma alternativa seria usar o GitHub Codespace, para isso clique aqui para criar o codespace na sua conta GitHub

Ferramenta para modelagem UML

- **Mermaid** – https://mermaid.js.org
  - Biblioteca JavaScript para criação de diagramas
  - Tem extensão para o Visual Studio Code e IntelliJ
  - Editor online – https://mermaid.live
- **StarUML** – https://staruml.io
  - Windows, Linux e macOS: baixar instalador no site oficial
- **draw.io** – https://draw.io
  - Ferramenta de propósito geral para criação de diagramas

Veja mais detalhes sobre as ferramentas usadas nessa disciplina em https://emersonmello.me/ensino/poo/#modelagem-uml

Ferramentas para essa disciplina

Sistema de controle de versão

Sistema de controle de versão

(*Version Control System* – VCS)

- Ferramentas de software que **ajudam o time de desenvolvimento a gerenciar as mudanças de código** ao longo do tempo
- Protege o código fonte de catástrofes e a degradação natural provocada por erros humanos
  - Possível saber por quem e quando tal arquivo ou trecho do arquivo foi alterado
- **Soluções como Dropbox e Google Drive não seriam adequadas** para cenários onde o VCS é necessário

Git e GitHub

- **git**
  - Sistema de controle de versão distribuído e de código aberto
  - Criado em 2005 por Linus Torvalds para gerenciar o desenvolvimento do kernel Linux
  - Clique aqui para acessar folha de dicas com os principais comandos
- **GitHub**
  - Plataforma de hospedagem de projetos de software que faz uso do git para controle de versão
  - Conta gratuita permite ter repositórios privados ou públicos, além de permitir a interação com projetos de terceiros
  - Estudantes do IFSC possuem benefícios – http://education.github.com

Configuração inicial para uso do git[5]

Baixe e instale o git https://git-scm.com/downloads

- Configure seu nome e email (usado para identificar o autor dos commits)

```
git config --global user.name "Seu Nome Completo"

git config --global user.email "seu-email@example.com"
```

- Crie um apelido chamado `tree` para o comando `log`

```
git config --global alias.tree "log --oneline --graph --decorate --all"
```

[5] As configurações ficarão salvas no arquivo `$HOME/.gitconfig`

Trabalhando com repositório git

1. Crie um diretório para o seu projeto

```
mkdir primeiro-projeto
cd primeiro-projeto
```

2. Inicialize um novo repositório

```
git init
```

3. Trabalhe nos arquivos

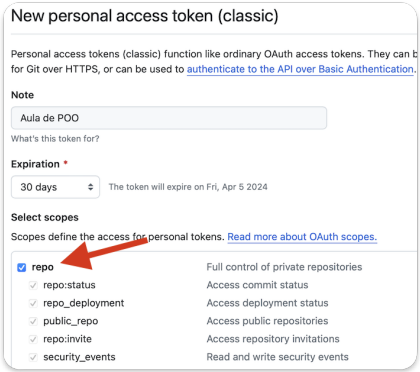
```
echo -e "Olá mundo\n" > ola.txt
echo -e "Primeiro projeto com git\n" >> ola.txt
```

4. Persista as alterações no repositório

```
git add ola.txt
git commit -m "Criado arquivo ola.txt"
```

GitHub e credenciais de acesso

- GitHub não aceita mais autenticação via senha para acesso via linha de comando
  - Somente por *Personal Access Token* (PAT)
- Acesse `https://github.com/settings/tokens`
  - Ao criar o token marque a permissão para `repo`



Salvar em *cache* as credenciais de acesso ao GitHub

- Para não ter que digitar o token a cada operação via linha de comando, é possível usar o *credential helper* do git

```
# Irá armazenar as credenciais na memória por 15 minutos
git config --global credential.helper cache

# Caso queira armazenar por mais tempo, use (irá armazenar por 1 hora)
git config --global credential.helper 'cache --timeout=3600'
```

- No seu computador pessoal talvez queira usar o Git Credential Manager[6]

[6] https://github.com/git-ecosystem/git-credential-manager

Markdown é uma linguagem de marcação simples

Extensão do arquivo: `.md`

- Documentos com formatação básica sem usar um editor de texto rico
- Uma boa opção para criar READMEs de projetos no GitHub

- https://www.markdownguide.org/cheat-sheet
- https://guides.github.com/features/mastering-markdown

Tutorais e documentação sobre git

- https://www.atlassian.com/br/git/tutorials/setting-up-a-repository
- https://git-scm.com/book/pt-br/v2
- https://training.github.com/downloads/pt_BR/github-git-cheat-sheet/