INF319 — Projeto e Implementação Orientados a Objetos

Luiz E. Busato

buzato@ic.unicamp.br

IC, UNICAMP

Objetivo

- Introdução a projeto orientado a objetos (POO) através de **exemplos**.
- Ao final do curso, o aluno deverá ter *melhorado sua capacidade de realização de projeto e implementação orientados a objetos*.

Supõe-se

que os alunos querem *ativamente* aumentar seu conhecimento

- e conseguem interpretar especificações de sistemas computacionais [INF330];
- já têm exposição a orientação a objetos e UML [INF318];
- têm domínio de algoritmos, estruturas de dados e Java básico [graduação, INF323, INF335.];
- conhecem ambientes de programação para Java [INF323 e INF335].

Organização

Curso é *auto-contido*

- Organizado como um conjunto de projetos exemplo de onde se aprende o conhecimento sobre projeto orientado a objetos.
- Os projetos permitem a derivação de exercícios que são resolvidos pelos alunos.
- Neste curso, aprende-se fazendo.

Método de Ensino

- O ensino de projeto orientado a objetos ocorre através do desenvolvimento de soluções para projetos onde alunos, professor e monitor interagem ativamente.
- A complexidade dos projetos orientados a objetos propostos vai de **muito simples** a medianamente complexo.
- Se quiserem, os alunos podem utilizar auxiliares de Inteligência Artificial (IA) para apoiar o seu aprendizado, mas devem indicar no código fonte, via comentário, que os utilizaram e devem saber explicar o código fonte da solução obtida.

Dinâmica de Aula

Aulas realizadas à distância (EAD)

- Aulas síncronas: Professor, monitor e alunos estão presentes no ambiente de aula e interagem sincronamente, isto é, em tempo real.
- Aulas assíncronas: Alunos executam autonomamente exercícios disponíveis no ambiente de apoio ao ensino (Moodle, Jenkins). Dúvidas encontradas nas aulas assíncronas serão resolvidas nas aulas síncronas.

Dinâmica de Aula

Interativo e Iterativo

- Toda aula síncrona é composta por uma sequência de iterações;

  Cada iteração contém as seguintes fases:

  i) estudo da solução do projeto anterior, se houver.
  ii) especificação de um projeto.
  iii) resolução do projeto pelos alunos;
  iv) avaliação de uma solução para o projeto.

Programa

Aula 1	Programa da Disciplina. Projeto Orientado a Objetos (POO). POO: Projetos simples. Exercícios.
Aula 2	Lista de Materiais. Exercícios.
Aula 3	Progressões. Exercícios.
Aula 4	Progressões. Exercícios.
Aula 5	Cafeteira. Exercícios.
Aula 6	Cafeteira. Exercícios.
Aula 7	Cafeteira. Exercícios.

Avaliação

Convenções

- D_i: Data da aula i, com $1 \leq i \leq 7$, de acordo com o cronograma do curso.
- E_k: Exercício k, com $1 \leq k \leq n$; n depende do andamento da turma.
- f = freqüência às aulas (%);
- me = média dos exercícios;
- mf = média final.

Avaliação

Notas

A nota (avaliação) de cada exercício E_k será determinada via a inspeção e teste de código fonte (jenkins) e/ou avaliação (professor/monitor) da resposta fornecida.

Frequência

Contabilizada via Google Meet Attendance List, logs do git e do jenkins.

Exercício (E_k)

Atribuição de notas aos exercícios

- Para cada E_k indicado em D_i e resolvido até um dia antes (24h) da aula seguinte (D_{i+1}):
  - $E_k = 0{,}0$: código não compila ou está incompleto;
  - $E_k = 5{,}0$: código compila mas não passa totalmente nos testes;
  - $E_k = 10{,}0$: código compila e passa em todos os testes;
- Um dia antes (24h) da aula D_{i+1}: Avaliação de E_k é parada. Durante a aula D_{i+1} ocorre a discussão de uma solução para E_k.
- Para cada E_k resolvido após D_{i+1}; até o prazo final de $\min(D_{i+2}, D_7)$:
  - $E_k = 0{,}0$: código não compila ou está incompleto;
  - $E_k = 4{,}0$: código compila mas não passa totalmente nos testes;
  - $E_k = 7{,}0$: código compila e passa em todos os testes.

Critério de Aprovação

Cálculo da média final

$$[0{,}0 \leq E_{k\,(1 \leq k \leq n)} \leq 10{,}0]$$

$$me = \sum_{k=1}^{n} E_k / n$$

$$mf = \begin{cases} me & \text{se } me \geq 7{,}0 \wedge f \geq 75\%\text{: aprovou-se} \\ \lfloor me \rfloor & \text{caso contrário: reprovou-se} \end{cases}$$

Monitor

Arthur V. de Lima Gomes (arthurv@unicamp.br)

Referências

- Ambiente de Apoio ao Ensino (Moodle)
- Jenkins
- Sala de Aula (Google Meet)
- Linguagem de Programação Java
- UML
- Eclipse, Netbeans, IntelliJ, Visual Studio Code
- JUnit
- git, maven
- IA (copilot, chatGPT, claude, gemini, etc)

Dúvidas?