MCTA018 - Programação Orientada a Objetos
Plano de ensino

Prof. Diogo S. Martins
Centro de Matemática, Computação e Cognição
Universidade Federal do ABC

Q2 2021
v.24/05

1 Informações básicas

- TPI: 2-2-4
- Horários oficiais:

seg	10-12h	semanal
qua	08-10h	semanal

- Ferramentas de comunicação:
  - Discord
    Para cadastrar-se na sala, utilizar link disponível no Moodle.
  - Comunicação com o professor
    * Prioritariamente via Discord, por mensagem pública ou privada
    * Por email (caso Discord esteja indisponível): `santana.martins@ufabc.edu.br`
- Plantão de dúvidas:

  qua 10-12h Discord

  Os plantões são nossos únicos eventos síncronos. O objetivo é esclarecer dúvidas e/ou reforçar temas vistos nas vídeo-aulas e outros materiais. Caso haja interesse em participar do plantão, é necessário agendar um slot no Google Calendar. Vide Moodle, para o link de agendamento.

  As comunicações com o professor, exceto no horário de plantão, serão assíncronas, em geral com resposta dentro de 24h em dias úteis.
- Sala no Moodle: `poo-2021.2-da3`

  `https://moodle.ufabc.edu.br/course/view.php?id=1851`

  Todos já foram convidados para a sala, em caso de dificuldade de acesso, entrar em contato com o professor.

2 Descrição da disciplina

A disciplina aborda os fundamentos do paradigma de programação orientada a objetos, os princípios de linguagem de modelagem de software orientada a objetos e alguns padrões de projeto (*design patterns*) fundamentais.

3 Requisitos recomendados

Para participar dessa disciplina é recomendação oficial ter cursado e sido aprovado em:

- BCM0505 - Processamento da Informação
- MCTA028 - Programação Estruturada

4 Objetivos

- Familiarizar-se com os conceitos básicos do paradigma orientado a objetos, a saber: objetos, classes, passagem de mensagens, encapsulamento, herança e polimorfismo;
- Ser capaz de transferir os conceitos básicos para uma linguagem de programação orientada a objetos, por exemplo, Java;
- Ser capaz de projetar software orientado a objetos;
- Ser capaz de expressar o projeto de software via uma linguagem de modelagem (UML);
- Familiarizar-se com os padrões de projeto (*design patterns*) básicos, bem como aplicá-los em problemas concretos.

Ao final do curso, espera-se que o aluno, aprovado com conceito satisfatório, possua habilidades que permitam, a partir de um conjunto de requisitos, projetar software orientado a objeto que seja eficiente, confiável, seguro e de fácil manutenção.

5 Bibliografia

1. Liang, D. Y. Introduction to Java Programming and Data Structures, 11th. edition. Pearson, 2018.
2. Deitel & Deitel. Java, How to Program (Early Objects), 11th. edition. Pearson, 2017.
3. Sedgewick, R. & Wayne, K. Computer Science: An Interdisciplinary Approach. 1st. edition. Addison-Wesley Professional, 2016.
4. Larman, C. Applying UML and Patterns, 2nd. edition. Pearson, 2005.

6 Metodologia de ensino-aprendizagem

Formato das aulas. Assíncronas, em formato de vídeo. Cada aula será disponibilizada no respectivo horário oficial da turma. O conteúdo englobará, além de aulas teóricas expositivas, também tutoriais com práticas de programação, os quais, idealmente, devem ser seguidos de forma síncrona, ou seja, que os alunos tentem implementar os exemplos ao mesmo tempo em que assistem ao vídeo.

Interações. Em duas modalidades: *i)* assíncrona, via Discord canal de texto, com prazo máximo de 24 horas para resposta (em dias úteis); e ii) síncrona, via Discord canal de voz, no horário de plantão de dúvidas (vide seção 1).

Técnicas de ensino aprendizagem:

- **Aprendizagem ativa**[1]. As atividades das videoaulas enfatizam a análise, a síntese e a avaliação dos conteúdos, ou seja, haverá menor ênfase no consumo passivo de informações. Por esta razão, é importante seguir os tutoriais de programação sincronamente aos vídeos.
- **Aprendizagem baseada em problemas**[2]. O conteúdo das aulas é estruturado em problemas-base concretos, os quais servem de contexto para abordar os temas conceituais/abstratos da ementa. O reforço do conhecimento teórico-conceitual é abordado via estudo prévio ou posterior com os *materiais complementares* (referências de estudo, que podem ser textos ou vídeos) indicados no roteiro da respectiva aula.

[1] https://en.wikipedia.org/wiki/Active_learning
[2] https://en.wikipedia.org/wiki/Problem-based_learning

7 Avaliação

A avaliação consiste nos componentes dados pela Equação 1, onde:

$$N_F = 0.5 \cdot N_{atv} + 0.25 \cdot N_{P1} + 0.25 \cdot N_{P2} \tag{1}$$

- N_{atv} é a média de 75% das atividades com as melhores notas. Como total para calcular a porcentagem, consideramos todas as atividades enunciadas durante o quadrimestre;
- N_{P1} e N_{P2} são as notas da Prova 1 e Prova 2, respectivamente;

O conceito final será obtido de acordo com a Equação 2.

$$C_F = \begin{cases} \texttt{A}, \text{ se } N_F \in [8.5, 10.0] \\ \texttt{B}, \text{ se } N_F \in [7.0, 8.5) \\ \texttt{C}, \text{ se } N_F \in [5.5, 7.0) \\ \texttt{D}, \text{ se } N_F \in [5.0, 5.5) \\ \texttt{F}, \text{ se } N_F \in [0.0, 5.0) \\ \texttt{O}, \text{ se ausência exceder 25\%} \end{cases} \tag{2}$$

Sobre as Atividades

Teremos um total de 5 atividades somativas, ou seja, valendo nota. Caberá ao aluno demonstrar disciplina e organização de tempo para executá-las, visto que são planejadas para preencher o componente I (4 horas/sem.) previsto oficialmente para a disciplina.

Auto-avaliação formativa. As atividades contarão, sempre que possível, com recursos de verificação automática de corretude e estilo de programação, com o objetivo de auxiliar o estudante a detectar, de modo automatizado, problemas recorrentes de programação. Na ausência de recursos de verificação automática, o estudante contará com uma rubrica para analisar a aderência do seu resultado ao que é esperado. Cabe ao aluno garantir que o programa passe por *todas* as verificações automáticas. Além disso, nas atividades em que haja rubrica, deve garantir que todos os critérios sejam atendidos. Submissões com falhas terão descontos substanciais na nota.

Avaliação somativa. Além dos testes automáticos, o professor analisará as submissões em atenção aos aspectos qualitativos e éticos detalhados neste plano de ensino. Ou seja, passar em todos os testes automáticos não implica necessariamente em nota máxima, visto que análise posterior pode implicar em desconto parcial ou total na nota da atividade.

Prazos. Cada atividade tem prazo de 15 dias para entrega. É possível entregar atividades atrasadas, sendo o limite máximo para entrega a data da prova 2. Atividades atrasadas terão desconto de nota proporcional ao atraso.

Sobre as Provas

As provas serão remotas. Cada prova ficará aberta pelo prazo de 72h (entre terça-feira e quinta-feira), de modo que o aluno possa escolher o melhor dia desse prazo para executá-la. O aluno terá acesso ao enunciado somente quando iniciar explicitamente a prova. Após iniciada, haverá um prazo, pré-anunciado, de 2 a 3 horas, para entregá-la. O prazo será controlado automaticamente pelo sistema de submissão. Não serão aceitas, em hipótese alguma, provas entregues fora do prazo regulado pelo Moodle ou enviadas por meios de submissão alternativos.

Sobre o sistema de submissão

Tanto as atividades quanto as provas serão submetidas via Moodle e via GitHub Classroom. O conteúdo da primeira semana de aula envolve treinamento sobre o uso combinado destas duas plataformas de submissão.

Critérios de avaliação

Todas as avaliações somativas, isto é, as atividades e as provas, qualificam-se como atividades de programação.

A nota máxima de cada atividade será obtida apenas se a mesma for entregue no prazo e executada correta e completamente.

Os programas solicitados em atividades de avaliação serão submetidos aos seguintes tipos de verificações:

- **Verificações automáticas**. Testes de corretude, de estilo de programação, de erros comuns e de detecção de plagiarismo.
- **Verificações manuais**. O professor inspecionará os programas para verificar os seguintes critérios gerais:
  - **Eficiência**: os programas desenvolvidos deverão ter bom desempenho, o que pode englobar o tratamento adequado dos seguintes fatores:
    - Ler e escrever dados nas quantidades mínimas necessárias para resolver o problema;
    - Não desperdiçar memória primária (RAM);
    - Acessar memória secundária (disco) somente quando necessário e sem redundância;
    - entre outros.
  - **Acurácia**: o programa deverá atender adequadamente a todos os requisitos enunciados para a atividade;
  - **Corretude**: o programa deverá atender a todos os requisitos enunciados e a solução deverá estar funcionalmente correta;
  - **Estrutura e organização do código**: atentar principalmente aos seguintes aspectos:
    - **Auto-documentação**: nomes intuitivos para variáveis, métodos, classes, interfaces e pacotes;
    - **Modularização**: métodos e classes com alta concisão e baixo acoplamento, isto é, que sejam em sua maioria curtos, e que realizem preferencialmente uma única tarefa;
    - **Comentários**: documentação completa porém ao mesmo tempo concisa (sem poluição visual, apenas nos lugares adequados e necessários).
  - **Autenticidade**: o código é original e não foi copiado de outras fontes (i.e. web, livros, terceiros, etc.), sob pena das sanções previstas no código de honra.

Mecanismos de avaliação substitutivos

Teremos dois mecanismos de avaliação substitutiva:

- Para as atividades, como consideraremos 75% das maiores notas, para substituir basta entregar com atraso até atingir no mínimo 75% de entregas;
- No caso da prova, que é assíncrona com janela de 72h, fica implícita a possibilidade de substituição, dado que o aluno pode escolher o melhor momento para iniciá-la. Casos omissos serão analisados individualmente.

Mecanismo de recuperação

A recuperação será aplicada apenas aos alunos que tiverem conceito final D ou F. Ocorrerá no início do quadrimestre subsequente, em formato similar ao estabelecido para as provas regulares.

A nota obtida na prova de recuperação (N_R) será usada para obter a nota final com recuperação (N_{FR}), que consiste na média estabelecida pela Equação 3.

$$N_{FR} = \frac{N_F + N_R}{2} \tag{3}$$

O conceito final obtido na recuperação (C_{FR}) é o conceito que entrará no histórico, obtido de acordo com os limiares para a nota final de recuperação (N_{FR}) dados pela Equação 4.

$$C_{FR} = \begin{cases} \texttt{C}, \text{ se } N_{FR} \geq 5.5 \\ \texttt{D}, \text{ se } N_{FR} \in (5.0, 5.5) \\ \texttt{F}, \text{ se } N_{FR} \leq 5.0 \end{cases} \tag{4}$$

No caso dos alunos que não participarem da recuperação, $C_{FR} = C_F$.

8 Código de honra

A aprovação na disciplina é baseada exclusivamente no esforço e trabalho pessoal do discente, ao qual cabe garantir que não ajudará ou receberá ajuda não-permitida em qualquer atividade avaliativa (e.g. provas, trabalhos, listas, etc.).

Exemplos de violação do código de honra incluem:

- Copiar atividades avaliativas (e.g. listas, trabalhos, provas, etc.) ou permitir que outros discentes copiem suas atividades avaliativas;
- Colaboração não-permitida entre indivíduos ou grupos (e.g. oferecer vantagens em troca de soluções prontas, doar trechos para o trabalho de outro grupo, etc.);
- Permitir que outros assumam sua identidade em atividades avaliativas (e.g. entregar trabalho que não fez ou permitir que outros façam provas por você);
- Plagiarismo (i.e. aplicável a textos, programas de computador, etc.), o que envolve copiar porções significativas de textos ou programas de terceiros, sem atribuição de autoria ou, mesmo que haja atribuição de autoria, demonstre-se que não houve trabalho original significativo;
- Receber ou conceder ajuda em atividades avaliativas quando o contexto mostra que não é sensato receber tal ajuda.

Como consequências de violação do código de honra tem-se:

- Reprovação automática na disciplina, com conceito F, sem direito ao mecanismo de recuperação;
- Denúncia na Comissão de Transgressões Disciplinares Discentes da Graduação, a qual decidirá sobre a punição adequada à violação, o que pode levar a advertência, suspensão ou desligamento, de acordo com os arts. 78-82 do Regimento Geral da UFABC.

9 Cronograma de aulas

O cronograma a seguir pode variar de acordo com o aproveitamento aferido nas turmas durante o quadrimestre.

Aula #	Data	Tema
01	24/05	Revisão Java (Programação estruturada)
02	26/05	Revisão Java (Programação estruturada)
03	31/05	Objetos
04	02/06	Objetos
05	07/06	Classes
06	09/06	Classes
07	14/06	Herança
08	16/06	Herança
09	21/06	Polimorfismo
10	23/06	Polimorfismo
11	28/06	Exercícios + Prova 1
12	30/06	Prova 1
13	05/07	Exceções
14	07/07	Exceções
15	12/07	Arquivos
16	14/07	Arquivos
17	19/07	Programação baseada em eventos
18	21/07	Programação baseada em eventos
19	26/07	Programação genérica
20	28/07	Programação genérica
21	02/08	Exercícios
22	04/08	Exercícios
23	09/08	Exercícios + Prova 2
24	11/08	Prova 2