MCTA018 - Programação Orientada a Objetos
Plano de ensino

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

QS 2020
v.21/09

1 Informações básicas

- TPI: 2-2-4
- Horários oficiais:

ter	19-21h	semanal
sex	21-23h	semanal

- Plantão de dúvidas:

  ter 18-19h Sala Discord

- Comunicação com o professor:
  - Prioritariamente na sala Discord. Link de convite: `https://discord.gg/qfgy5QA`
    Usar o link para se inscrever. Preferencialmente, usar seu nome real no perfil, para facilitar a comunicação. Os plantões são nossos únicos eventos síncronos. A comunicação será via voz, texto, vídeo e compartilhamento de tela, com o objetivo de esclarecer dúvidas e/ou reforçar temas vistos nas videoaulas e outros materiais.
  - Por email (caso Discord indisponível): `santana.martins@ufabc.edu.br`

  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-2020qs-na2`

  `https://moodle.ufabc.edu.br/course/view.php?id=701`

  Todos foram convidados para a sala, verifique seu email institucional.

2 Descrição da disciplina

A disciplina apresenta conceitos básicos de programação orientada a objetos, abrangendo tópicos de análise e desenvolvimento de software dentro desse paradigma. 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: classes, objetos, 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. Deitel & Deitel. Java, How to Program (Early Objects), 11th. edition. Pearson, 2017.
2. Larman, C. Applying UML and Patterns, 2nd. edition. Pearson, 2005.

6 Metodologia de ensino-aprendizagem

Formato das aulas. As aulas serão assíncronas, em formato de vídeo-aula, duas aulas por semana. Em semana de prova, reduziremos a frequência para apenas uma aula. Cada aula será disponibilizada no respectivo dia e horário oficial da turma. O conteúdo englobará tutoriais, os quais, idealmente, devem ser seguidos de forma síncrona, ou seja, que os alunos tentem implementar os exemplos ao mesmo tempo que em assistem ao vídeo. Essa abordagem tende a tornar mais explícitas as eventuais dificuldades e dúvidas.

Interações. Em duas modalidades: i) assíncrona, via Discord canal de texto, com prazo 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 conteúdos;
- **Aprendizagem baseada em problemas**[2]. O conteúdo das aulas é estruturado fundamentalmente em problemas-base concretos, os quais servem de contexto para abordar os temas conceituais/abstratos da ementa;
- **Aula invertida**[3]. O tempo da aula será utilizado essencialmente para a resolução dos problemas-base. A aquisição do conhecimento teórico-conceitual é abordado via estudo prévio ou posterior com os materiais (referências de estudo) indicados no roteiro da respectiva aula. Como as referências de estudo são divulgadas juntamente com as videoaulas, cabe ao aluno decidir se deve estudá-las antes ou depois de assitir aos vídeos. Embora cada aula inclua uma sucinta revisão/contextualização dos aspectos teórico-conceituais, é de fundamental importância que haja o estudo individual das referências de estudo para garantir a profundidade do tema.

Ciclos. O curso é estruturado em ciclos, sendo que cada ciclo dura uma semana e é composto dos seguintes momentos:

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

1. **Estudo prévio ou posterior**. Alunos estudam os conteúdos do ciclo, com base nos recursos indicados pelo professor (textos, vídeos, tutoriais interativos, etc.), antes ou depois da aula, a depender do tema;
2. **Aula**. Resolução de problemas ou mini-projetos sobre o conteúdo estudado, durante a aula online;
3. **Avaliação formativa**. As atividades para entrega contarão, quando possível, com recursos para verificação automática de corretude e boas práticas de programação, ou mesmo rubricas para auto-avaliação manual, as quais devem ser analisadas e garantidas pelos alunos antes da submissão.
4. **Avaliação somativa**. Após a aula, é necessário entregar uma atividade desenvolvida para a verificação de conhecimento ou propor uma resolução para um problema relacionado. Adicionalmente, teremos duas avaliações globais com restrição de tempo de execução.

7 Avaliação

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

$$N_F = 0.4 \cdot N_{atv} + 0.3 \cdot N_{P1} + 0.3 \cdot N_{P2} \tag{1}$$

- N_{atv} é a média de 75% das atividades somativas com as melhores notas. Como total para calcular a porcentagem, consideramos todas as somativas enunciadas durante o quadrimestre (por volta de 12);
- 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 por volta de uma atividade por semana, com prazo de entrega de 7 dias. Todas as atividades são somativas, ou seja, valem nota. As atividades serão entregues no ambiente Github Classroom. O conteúdo do ciclo 1 engloba treinamento sobre o uso da plataforma. 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. Em ambos os casos, convencionamos como critérios de auto-avaliação:

- **Satisfatório**: solucionou todos os problemas/requisitos da atividade;
- **Pouco satisfatório**: solucionou alguns problemas/requisitos da atividade;
- **Insatisfatório**: não tentou fazer a atividade.

Com base em ofertas anteriores em que a metodologia foi utilizada, pode-se afirmar que há uma correlação forte e positiva entre desempenho satisfatório nas atividades e a nota da prova, ou seja, quem efetua as atividades costuma ser aprovado na disciplina. Portanto, caso o aluno sistematicamente não efetue as atividades, deve preocupar-se, pois provavelmente terá desempenho insatisfatório nas provas, o que tende a implicar em reprovação.

Sobre as provas

As provas serão remotas, via Moodle. Cada prova ficará aberta pelo prazo de uma semana, de modo que o aluno possa escolher o melhor dia desse prazo para executá-la. Porém, o aluno terá acesso ao enunciado somente quando iniciar explicitamente a prova. Após iniciada, haverá um prazo pré-determinado (de 2 a 3 horas) para entregá-la. O prazo será controlado automaticamente pelo sistema de submissão. Não serão aceitas provas atrasadas ou entregues por meios de submissão alternativos.

Critérios de avaliação

Todas as avaliações somativas, isto é, atividades e provas, qualificam-se como atividades de programação. No caso das atividades que envolvam prazo de entrega, não serão aceitas as que forem entregues fora do prazo. 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. Parte dos testes (por volta de 70%) serão distribuídos juntamente com as atividades, de modo que os alunos podem executá-los localmente antes de entregar. A outra parte dos testes (por volta de 30%) serão privados, i.e., não serão divulgados para os alunos.
- **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 passar em todos os testes publicados pelo professor, bem como deverá apresentar boa cobertura (por volta de 80%) de testes nos módulos adicionais implementados pelo aluno;
  - **Estrutura e organização do código**: atentar principalmente aos seguintes aspectos:
    - **Auto-documentação**: nomes intuitivos para variáveis e métodos/funções;
    - **Modularização**: funções/métodos/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.

7.1 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 uma semana, fica implícita a possibilidade de substituição, dada a possibilidade do aluno escolher o melhor momento para submeter.

7.2 Mecanismo de recuperação

A recuperação será aplicada apenas aos alunos que tiverem conceito final D ou F. Ocorrerá na data estabelecida no cronograma, em formato similar ao estabelecido para as outras provas.

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 Cronograma de aulas

Cada semana corresponde a um ciclo. O cronograma a seguir pode variar de acordo com o aproveitamento aferido nas turmas durante o quadrimestre.

Semana #	Tema
1	Revisão Java
2	Objetos e classes
3	Métodos
4	Herança
5	Polimorfismo
6	Estudo de caso + Prova 1
7	Tratamento de exceções
8	Programação genérica
9	Programação baseada em eventos + GUI
10	UML
11	Padrões de projeto
12	Revisão + Prova 2
13	Prova de recuperação

9 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 usada pela equipe docente para fins de avaliação (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;
- 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;
- 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.