

# MCTA018 - Programação Orientada a Objetos

## Plano de ensino

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

Q2 2023  
v.29/05

### 1 Informações básicas

- TPI: 2-2-4
- Horários oficiais:

Teoria	Prof. Diogo	seg	19-21h	S-212-0	semanal
Prática (NA1)	Prof. Diogo	qui	21-23h	404-2	semanal
Prática (NA2)	Prof. Paulo	qui	21-23h	407-2	semanal
- Ferramentas de comunicação:
  - Moodle
  - Email: [santana.martins@ufabc.edu.br](mailto:santana.martins@ufabc.edu.br) (informar disciplina e turma na mensagem)
- Plantões:

Prof. Diogo	seg 18-19h	S528-2
Prof. Diogo	qui 20-21h	S528-2
Prof. Paulo	qui 19h30-20h30	S507-2

O objetivo do plantão é esclarecer dúvidas e/ou reforçar temas vistos nas aulas ou outros materiais.
- Sala no Moodle: `poo-2023-q2-noturno`  
<https://moodle.ufabc.edu.br/course/view.php?id=4887>  
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, bem como 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;
- 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.

## 6 Metodologia de ensino-aprendizagem

**Indissociabilidade entre teoria e prática.** As aulas combinam os conteúdos de teoria e prática com dinâmica fluída, ou seja, teremos breves resumos teóricos seguidos de sessões mais longas de programação com introdução de novos conceitos teóricos à medida que mostrem-se necessários nos contextos mais adequados.

**Aprendizagem ativa.** <sup>1</sup> A dinâmica da aula enfatiza a análise, a síntese e a avaliação dos conteúdos, em detrimento do consumo passivo de conhecimento pronto. A abordagem segue a linha do “aprender fazendo” ao invés do “aprender olhando”. Por esta razão, reforça-se a importância de, sempre que possível, seguir os tutoriais de programação sincronamente às aulas. Além disso, a dinâmica das aulas depende da iniciativa individual do aluno, o qual deve tornar-se protagonista do seu aprendizado, resolvendo os problemas e trazendo suas dúvidas, caso contrário seu aproveitamento nas atividades presenciais será baixo.

**Dinâmica das atividades presenciais.** Dispomos de um dia de teoria (em sala de aula) e um dia de prática (em laboratório). Dada essa configuração, utilizaremos a infra-estrutura dos espaços físicos do seguinte modo:

- **Aulas teóricas.** Como não dispomos de computadores individuais, teremos, além da discussão dos conteúdos teóricos, também exercícios focados na análise de códigos prontos, questões mais conceituais, bem como a discussão de dúvidas levadas pelos alunos.
- **Aulas práticas.** Como neste espaço há disponibilidade de computadores individuais, teremos exercícios regulares de programação, com enunciados disponíveis no Moodle. As atividades serão sempre formativas, i.e., não valerão nota. Porém, a entrega das atividades no Moodle contará como frequência. O tempo em aula pode ser usado também para a discussão de dúvidas.

---

<sup>1</sup>[https://en.wikipedia.org/wiki/Active\\_learning](https://en.wikipedia.org/wiki/Active_learning)

## 7 Avaliação

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

$$N_F = 0.3 \cdot N_{atv} + 0.35 \cdot N_{P1} + 0.35 \cdot N_{P2} \quad (1)$$

- $N_{atv}$  é a média das atividades;
- $N_{P1}$  e  $N_{P2}$  são as notas da Prova 1 e Prova 2, respectivamente;

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

$$C_F = \begin{cases} A, & \text{se } N_F \in [8.5, 10.0] \\ B, & \text{se } N_F \in [7.0, 8.5) \\ C, & \text{se } N_F \in [6.0, 7.0) \\ D, & \text{se } N_F \in [5.0, 6.0) \\ F, & \text{se } N_F \in [0.0, 5.0) \\ O, & \text{se ausência exceder 25\%} \end{cases} \quad (2)$$

### 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.

**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. Não serão aceitas atividades entregues com atraso.

Nro.	Tema	Abertura	Fechamento
1	Revisão Java	01/06	15/06
2	Classes e objetos	15/06	29/06
3	Herança e polimorfismo	29/06	13/07
4	Exceções, Prog. genérica	13/07	27/07
5	Padrões de proj., Aplicações	27/07	17/08

### Sobre as Provas

As provas terão as seguintes características:

- Presenciais;
- Com duração de 100 minutos;
- 100% práticas, envolvendo problemas de programação;
- Efetuadas no computador do laboratório (não é permitido o uso de computador pessoal nem de dispositivos móveis);

- Com consulta, mas somente a materiais offline (ou seja, sem acesso à Internet);
- Entregue no sistema de submissão da disciplina.

O prazo será controlado automaticamente pelo Moodle. 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

As atividades e provas serão submetidas via GitHub Classroom e/ou Moodle. 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 qualidade de software, de erros comuns e de detecção de plágio.
- **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:** 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 coesão 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

É preciso apresentar justificativa prevista na Resolução ConsEPE 227. O documento de justificativa deve ser encaminhado por email ao professor, que cadastrará o aluno na prova.

### Mecanismo de recuperação

A recuperação será aplicada apenas aos alunos que tiverem conceito final D ou F. Ocorrera na data definida no plano de ensino, 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.

$$N_{FR} = 0.5 \cdot N_F + 0.5 \cdot N_R \quad (3)$$

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.

$$C_{FR} = \begin{cases} C, & \text{se } N_{FR} \geq 6.0 \\ D, & \text{se } N_{FR} \in [5.0, 6.0) \\ F, & \text{se } N_{FR} < 5.0 \end{cases} \quad (4)$$

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);
- Plágio (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.

<b>Aula #</b>	<b>Data</b>	<b>T/P</b>	<b>Tema</b>
1	29/05	T	Revisão Java
2	01/06	P	Revisão Java
3	05/06	T	Classes e objetos
	08/06		Feriado (reposição em 21/08)
4	12/06	T	Classes e objetos
5	15/06	P	Classes e objetos
6	19/06	T	Herança
7	22/06	P	Herança
8	26/06	T	Polimorfismo
9	29/06	P	Polimorfismo
10	03/07	T	Exercícios
11	06/07	P	Prova 1
12	10/07	T	Exceções
13	13/07	P	Exceções
14	17/07	T	Programação genérica
15	20/07	P	Programação genérica
16	24/07	T	Padrões de projeto e aplicações
17	27/07	P	Padrões de projeto e aplicações
18	31/07	T	Padrões de projeto e aplicações
19	03/08	P	Padrões de projeto e aplicações
20	07/08	T	Padrões de projeto e aplicações
21	10/08	P	Prova 2
22	14/08	T	Revisão de provas
23	17/08	P	Substitutiva
24	21/08	P	Recuperação