MCTA018 - Programação Orientada a Objetos
Plano de ensino

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

Q2 2024
v.26/06

1 Informações básicas

- TPEI: 2-2-0-4
- Horários oficiais:

Teoria	Prof. Diogo	qui	10-12h	S-214-0	semanal
Prática	Prof. Diogo	seg	08-10h	407-2	semanal

- Ferramentas de comunicação:
  - Moodle
  - Email: `santana.martins@ufabc.edu.br` (informar disciplina e turma na mensagem)
- Plantões:

  Prof. Diogo qui 09-10h S528-2

  O objetivo do plantão é esclarecer dúvidas e/ou reforçar temas vistos nas aulas ou outros materiais.
- Sala no Moodle:

  https://moodle.ufabc.edu.br/course/view.php?id=2178

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

2 Objetivos

A disciplina aborda os fundamentos do paradigma de programação orientada a objetos e aplica os conceitos com a linguagem de programação Java.

3 Requisitos recomendados

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

- BCM0505 - Processamento da Informação
- MCTA028 - Programação Estruturada

4 Conteúdo programático

- Objetos e mensagens;
- Classes, construtores e encapsulamento;
- Herança e polimorfismo;
- Sobrecarga e sobrescrita;
- Classes abstratas e interfaces;
- Tratamento de exceções;
- Linguagem de programação orientada a objetos;
- Estruturas de dados orientadas a objetos;
- Introdução aos diagramas UML.
- Padrões de projeto de software.

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

6 Metodologia de ensino-aprendizagem

- Aulas expositivas e dialogadas;
- Tutoriais de programação em laboratório;
- Atividades práticas de programação;
- Atividades de fixação e avaliação;
- Plantões de dúvidas.
- Uso de ferramentas de verificação automática de programas.

7 Avaliação

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

$$N_F = 0.1 \cdot N_{atv} + 0.45 \cdot N_{P1} + 0.45 \cdot N_{P2} \tag{1}$$

- 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.

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

Sobre as Atividades

As atividades consistem em listas de exercícios que devem ser entregues no Moodle.

Estrutura. Cada atividade será composta por um ou mais exercícios.

Frequência. Teremos uma atividade por semana, exceto nas semanas da prova.

Prazo. Cada atividade tem prazo de 15 dias para entrega. Não serão aceitas atividades entregues com atraso.

Verificação automática. As atividades contarão, sempre que possível, com recursos de verificação automática de corretude, 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.

Sobre as Provas

As provas terão as seguintes características:

- Presenciais;
- Com duração de 100 minutos;
- Efetuadas no computador do laboratório (não é permitido o uso de computador pessoal nem de dispositivos móveis);
- Sem consulta;
- Entregue no Moodle.

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.

Critérios de avaliação

A nota máxima de cada componente de avaliação (atividades e provas) será obtida apenas se o componente for entregue no prazo e executado correta e completamente.

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

- **Verificações automáticas**. Executadas pelo sistema de submissão.
  - **Sintaxe e semântica**. O programa deve compilar e executar sem erros ou avisos. **Programas que apresentem erros de compilação terão desconto de 100% na nota.**
  - **Corretude**. Os programas serão submetidos a testes unitários — falha em qualquer um dos testes resultará em desconto parcial ou total na nota, dependendo do peso do respectivo teste.
  - **Autenticidade**. Os programas serão submetidos a testes de autenticidade para detectar plagiarismo (de outros alunos, de outros quadrimestres, da web, de chatbots, etc.). Cópias implicarão nas sanções previstas no código de honra.
- **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;
- **Estrutura e organização do código**: atentar principalmente aos seguintes aspectos:
    - **Modularização**: modelar as classes de modo a favorecer o reúso, isto é, que possam ser usados por diferentes programas principais, em diferentes contextos, sem a necessidade de "copiar e colar" código;
    - **Auto-documentação**: nomes intuitivos para variáveis, métodos e classes, seguindo as convenções da linguagem Java, e organização do código de modo a facilitar a leitura e compreensão do que está sendo feito;
    - **Comentários**: documentação completa porém ao mesmo tempo concisa (sem poluição visual, apenas nos lugares adequados e necessários), sendo que os comentários explicam *o que* o código faz, e não *como* faz;
- **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. A prova substitutiva será realizada na data definida na seção 9.

Mecanismo de recuperação

A recuperação será aplicada apenas aos alunos que tiverem conceito final D ou F. Ocorrerá na data definida na seção 9, 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.

$$N_{FR} = 0.5 \cdot N_F + 0.5 \cdot N_R \tag{3}$$

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.

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

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 (de empresas, de pessoas ou de chatbots), 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 Disciplinar Discente da Graduação (CDDG), 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	T/P	Tema
01	03/06	P	Revisão Java
-	06/06	T	Greve
-	10/06	P	Greve
-	13/06	T	Greve
-	17/06	P	Greve
-	20/06	T	Greve
02	24/06	P	Revisão Java
03	27/06	T	Classes e objetos
04	01/07	P	Classes e objetos
05	04/07	T	Classes e objetos
-	08/07	-	Feriado
06	11/07	T	Classes e objetos
07	15/07	P	Herança
08	18/07	T	Herança
09	22/07	P	Herança
10	25/07	T	Herança
11	29/07	P	Prova 1
12	01/08	T	Interfaces
13	05/08	P	Interfaces
14	08/08	T	Polimorfismo
15	12/08	P	Polimorfismo
16	15/08	T	Programação genérica
-	19/08	-	Feriado
17	22/08	T	Programação genérica
18	26/08	P	Exceções
19	29/09	T	Exceções
20	02/09	P	Padrões de projeto
21	05/09	T	Padrões de projeto
22	09/09	P	P2
23	12/09	T	Sub
24	16/09	P	Rec (reposição 08/07)