Plano de ensino

Curso	Engenharia de Telecomunicações
Unidade curricular	Programação Orientada a Objetos
Semestre	2021-02
Carga horária	72 horas
Professor	Emerson Ribeiro de Mello
Página da disciplina	http://docente.ifsc.edu.br/mello/poo

1 Ementa

Introdução ao paradigma da orientação a objetos: Classes, objeto, associações entre classes, herança. Introdução à linguagem de modelagem unificada (UML): Diagramas de caso de uso, classes, sequência. Introdução a linguagem de programação Java: Tipos de dados primitivos, estruturas de controle, vetores; concepção de projeto orientado a objetos, herança, polimorfismo; interfaces gráficas amigáveis.

2 Objetivos

Ao término da disciplina o aluno será capaz de modelar, implementar e testar software de média complexidade na linguagem Java e de acordo com o paradigma da programação orientada a objetos. Os objetivos específicos da disciplina são:

- Introduzir os conceitos da programação orientada a objetos;
- Apresentar a linguagem de programação Java e a linguagem de modelagem unificada (UML);
- Usar de forma efetiva ferramentas como ambiente integrado de desenvolvimento e sistema de controle de versão para trabalhar de forma colaborativa;
- Modelar software de média complexidade por meio de diagramas UML comportamentais e estruturais.

3 Metodologia

O conteúdo da disciplina será apresentado por meio de aulas expositivas remotas e síncronas. Na parte prática da disciplina serão desenvolvidos exercícios, com tutoria assíncrona do professor, e trabalhos individuais, sendo esses conduzidos pelo próprio aluno. Os alunos serão avaliados por meio dos instrumentos apresentados na Tabela 1.

A nota dos projetos práticos é calculada por meio de uma média geométrica ponderada, com os seguintes pesos $W = \{w_1, w_2, w_3\} = \{3, 4, 2\}$. A nota das listas de exercícios é calculada por meio de uma média geométrica. Por fim, Conceito Final (CF) é calculado por meio de uma média ponderada,

Tabela 1: Instrumentos de avaliação

Quantidade	Atividade	Recuperação
3	Projetos práticos (p)	Dedução de 10% do valor total da nota para cada dia de atraso após o prazo de entrega, sendo 5 dias o prazo máximo para entrega tardia.
5	Listas de exercícios (e)	Dedução de 10% da nota quando entregue depois do prazo, sendo 5 dias o prazo máximo para entrega tardia.

os projetos com peso 0,9 e as listas de exercícios com peso 0,1. Sendo assim, o Conceito Final (CF) se dará por meio da Equação 1:

$$CF = \left\lfloor \left(\prod_{i=1}^{3} p_i^{w_i} \right)^{\frac{1}{\sum_{i=1}^{3} w_i}} \times 0{,}9 + \left(\prod_{i=1}^{5} e_i \right)^{\frac{1}{5}} \times 0{,}1 \right\rceil, \quad CF \in \mathbb{N}. \tag{1}$$

Para a aprovação o aluno deverá possuir no mínimo 75% de presença e ConceitoFinal ≥ 6. A não entrega de qualquer um dos projetos implica em nota zero nesse instrumento de avaliação. A não entrega de qualquer uma das listas de exercício implica em nota zero nesse instrumento de avaliação.

4 Conteúdo programático

1. Fundamentos (6h)
   (a) Paradigmas de programação: sequencial e estruturada
   (b) História e princípios da linguagem Java
   (c) Kit de Desenvolvimento Java (JDK) e Ambiente integrado de desenvolvimento (IDE)
   (d) Tipos primitivos, estruturas de decisão e repetição, vetores uni e multidimensionais
   (e) Leitura de dados do dispositivo de entrada padrão com classe Scanner
   (f) Principais métodos da classe String
   (g) Argumentos de linha de comando
2. Sistema de controle de versão (8h)
   (a) Motivação para uso de sistemas de controle de versão
   (b) Sistema de controle de versão centralizado (SVN) vs sistema de controle de versão distribuído (git)
   (c) Apresentação da ferramenta git
      i. Como iniciar um repositório git e registrar as modificações (commit)
      ii. Introdução a linguagem de marcação Markdown
      iii. Comandos git: status, add, commit, reset, diff, log, rm e mv
   (d) Trabalhando com repositórios remotos no Github
      i. Evitando o controle de alguns arquivos por meio do .gitignore
      ii. Clonando repositório remoto e sincronizando repositório local com o remoto
      iii. Trabalhando de forma colaborativa

iv. Comandos git: remote, clone, push, pull e fetch

(e) Trabalhando com ramos

i. Fluxos de trabalho com ramos em projetos colaborativos

ii. Resolvendo conflitos em uma mesclagem (merge conflict)

iii. Comandos git: branch, checkout, merge e rebase

3. Introdução ao paradigma da orientação a objetos (14h)

   (a) Processos de abstração e representação
   (b) Classes, objetos e membros de classe (atributos e métodos)
   (c) Encapsulamento de dados
   (d) Modelagem de classes
   (e) Métodos construtores
   (f) Sobrecarga de métodos
   (g) Membros estáticos e constantes
   (h) Modificadores de acesso e palavras reservadas

4. APIs Java (6h)

   (a) Tratamento de exceções
   (b) Coleções (lista, tabela de dispersão, conjuntos, pilha e fila)

5. Associação entre classes (8h)

   (a) Diagrama de classes UML
   (b) Dependência, agregação e composição

6. Testes de unidade e documentação de código (4h)

   (a) Desenvolvimento guiado a testes
   (b) Documentação com JavaDOC

7. Herança (4h)

   (a) Sobrescrita de métodos
   (b) Diagrama de classes UML

8. Classe abstrata, interface e polimorfismo (6h)

   (a) Conceito sobre herança múltipla
   (b) Implementação de interfaces
   (c) Diagrama de classes UML

9. Uso de APIs Java (8h)

   (a) Trabalhando com arquivos texto e binários
   (b) Enum e tipos genéricos
   (c) Programação concorrente com threads

10. Processo de desenvolvimento de software (4h)
    (a) Conceitos sobre processo unificado
    (b) Diagrama de casos de uso
    (c) Diagrama de sequência
    (d) Diagrama de colaboração
    (e) Diagrama de atividades
11. Linguagem de programação Python (8h)
    (a) Ferramentas e IDE
    (b) Estruturas básicas da linguagem
    (c) Orientação a objetos

Bibliografia

[1] Eduardo Bezerra. *Princípios de análise e projeto de sistemas com UML*. Campus, 2002.

[2] Caelum. Fj-11 – java e orientação a objetos. Technical report, Caelum Ensino e Soluções em Java, 2008. http://docente.ifsc.edu.br/mello/livros/java/apostila-caelum-java-orientacao-objetos-FJ11.pdf.

[3] H.M. Deitel and P.J. Deitel. *Java Como Programar*. Prentice Hall, 4 edition, 2003.

[4] Cay S. Horstmann and Gary Cornell. *Core Java – Volume I – Fundamentos*. Pearson, 8 edition, 2010.

[5] Ivar Jacbson, Magnus Christerson, Patrik Jonsson, and Gunnar Overgaard. *Object-oriented software engineering: a use case driven approach*. Addison-Wesley, 1992.

[6] Craig Larman. *Utilizando UML e padrões*. Bookman, 2007. https://app.minhabiblioteca.com.br/books/9788577800476/.

[7] Roger S. Pressman. *Engenharia de Software: uma abordagem profissional*. Bookman, 2011. https://app.minhabiblioteca.com.br/books/9788580555349/.