# # Programação Orientada a Objetos

## **## Sigla da disciplina**

POO

## **## Pré-requisitos**

PRG2

## **## Co-requisitos**

## **## Disciplinas recomendadas**

## **## Carga horária**

80

### ### Carga horária da parte teórica

40

### ### Carga horária da parte prática

40

### ### Carga horária da parte de extensão

0

## **## Professor responsável**

Emerson Ribeiro de Mello, Dr. (DE)

## **## Ementa**

Introdução ao paradigma da orientação a objetos. Introdução a uma linguagem de programação orientada a objetos. Introdução à linguagem de modelagem unificada (UML). Introdução sobre processo unificado de desenvolvimento de software. Uso de sistemas de controle de versão. Testes de unidade.

## **## Competências ou objetivos**

**Objetivo geral:**

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.

**Objetivos específicos:**

* 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.

## **## Conhecimentos, habilidades e atitudes ou conteúdos**

1. Fundamentos (6h)
   1. Paradigmas de programação: sequencial e estruturada
   2. História e princípios da linguagem Java
   3. Kit de Desenvolvimento Java (JDK) e Ambiente integrado de desenvolvimento (IDE)
   4. Tipos primitivos, estruturas de decisão e repetição, vetores uni e multidimensionais
   5. Leitura de dados do dispositivo de entrada padrão com classe Scanner
   6. Principais métodos da classe String
   7. Argumentos de linha de comando
2. Sistema de controle de versão (8h)
   1. Motivação para uso de sistemas de controle de versão
   2. Sistema de controle de versão centralizado (SVN) vs sistema de controle de versão distribuído (git)
   3. Apresentação da ferramenta git
      1. Como iniciar um repositório git e registrar as modificações (commit)
      2. Introdução a linguagem de marcação Markdown
      3. Comandos git: status, add, commit, reset, diff, log, rm e mv
   4. Trabalhando com repositórios remotos no Github
      1. Evitando o controle de alguns arquivos por meio do .gitignore
      2. Clonando repositório remoto e sincronizando repositório local com o remoto
      3. Trabalhando de forma colaborativa
      4. Comandos git: remote, clone, push, pull e fetch
   5. Trabalhando com ramos
      1. Fluxos de trabalho com ramos em projetos colaborativos
      2. Resolvendo conflitos em uma mesclagem (merge conflict)
      3. Comandos git: branch, checkout, merge e rebase
3. Introdução ao paradigma da orientação a objetos (14h)
   1. Processos de abstração e representação
   2. Classes, objetos e membros de classe (atributos e métodos)
   3. Encapsulamento de dados
   4. Modelagem de classes
   5. Métodos construtores
   6. Sobrecarga de métodos
   7. Membros estáticos e constantes
   8. Modificadores de acesso e palavras reservadas
4. APIs Java (6h)
   1. Tratamento de exceções
   2. Coleções (lista, tabela de dispersão, conjuntos, pilha e fila)
5. Associação entre classes (8h)
   1. Diagrama de classes UML
   2. Dependência, agregação e composição
6. Testes de unidade e documentação de código (4h)
   1. Desenvolvimento guiado a testes
   2. Documentação com JavaDOC
7. Herança (4h)
   1. Sobrescrita de métodos
   2. Diagrama de classes UML
8. Classe abstrata, interface e polimorfismo (6h)
   1. Conceito sobre herança múltipla
   2. Implementação de interfaces
   3. Diagrama de classes UML
9. Uso de APIs Java (8h)
   1. Trabalhando com arquivos texto e binários
   2. Enum e tipos genéricos
   3. Programação concorrente com threads
10. Processo de desenvolvimento de software (4h)
    1. Conceitos sobre processo unificado
    2. Diagrama de casos de uso
    3. Diagrama de sequência
    4. Diagrama de colaboração
    5. Diagrama de atividades
11. Desenvolvimento de aplicativos Android (12h)
    1. IDE, emulador e SDK
    2. Layout, views e resources
    3. Comunicação entre atividades
    4. Persistência de dados em área não volátil