

# PROJETO, IMPLEMENTAÇÃO E TESTE DE SOFTWARE

The background features a dark blue field with a white dot grid. On the right, there are abstract geometric shapes: a red triangle containing a network of white lines and nodes, and an orange triangle with a grid pattern. Several 3D cubes in blue and purple are scattered in the upper right area.

## Aula 05 – Implementação de Software

Professor Fabricio Freire

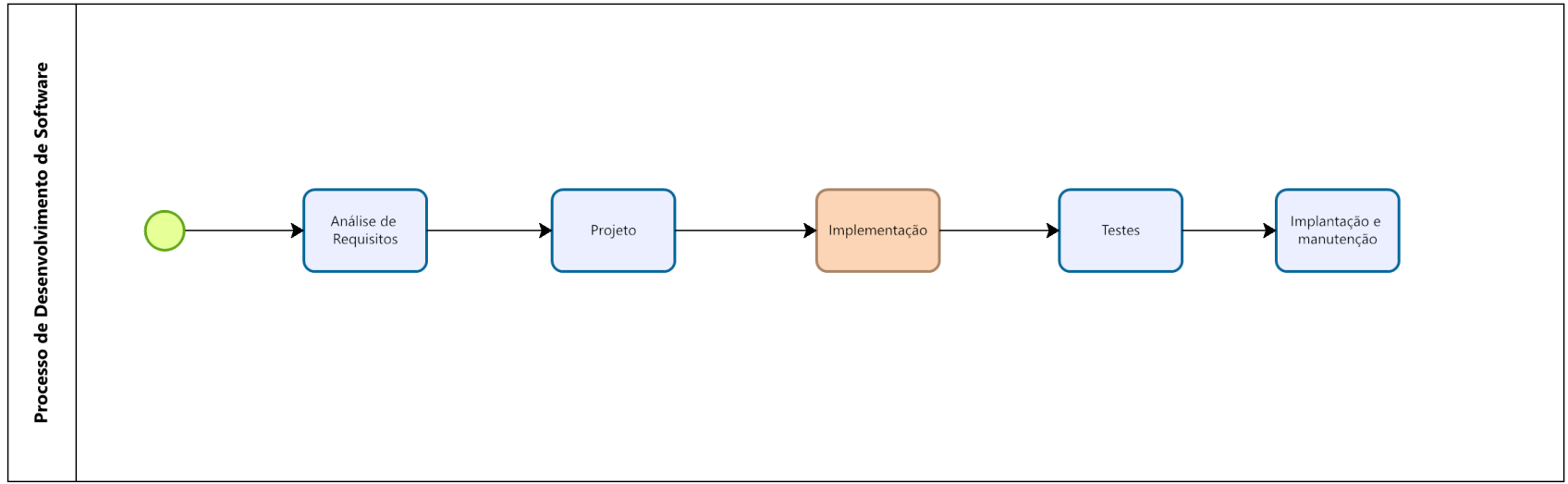
# METAS DE APRENDIZAGEM

- Introdução
  - Recapitulando
  - Aspectos Gerais
- Atividades Realizadas
- Características
- Estilo de programação e codificação
- Comentários

- Projeto da Arquitetura
- Projeto de Componentes
- Projeto de Interfaces
- Projeto de Dados



# ONDE ESTAMOS?



**“o objetivo final da maioria dos projetos de engenharia de software é produzir um programa funcional. O ato de transformar o projeto detalhado em um programa válido em alguma linguagem de programação, juntamente com todas as suas atividades de apoio é aludido como implementação”**

# ASPECTOS GERAIS

- Detalhamento dos componentes previstos no projeto;
- Descrição de todos os componentes de código fonte e código binário, em nível de linguagem de programação ou de acordo com as tecnologias escolhidas.

# ASPECTOS GERAIS

## Implementação

Codificação

Depuração

Compilação

Integração

Testes

# TAREFAS

- Planejamento detalhado da implementação das unidades de cada iteração.
- Implementação das classes e outros elementos do modelo de projeto, geralmente arquivos de código fonte.
- Verificação das unidades, por meio de revisões, inspeções e testes de unidade.
- Compilação, ligação das unidades e integração das unidades entre si.
- Integração das unidades com componentes reutilizados.





# ATIVIDADES DA IMPLEMENTAÇÃO DE SOFTWARE

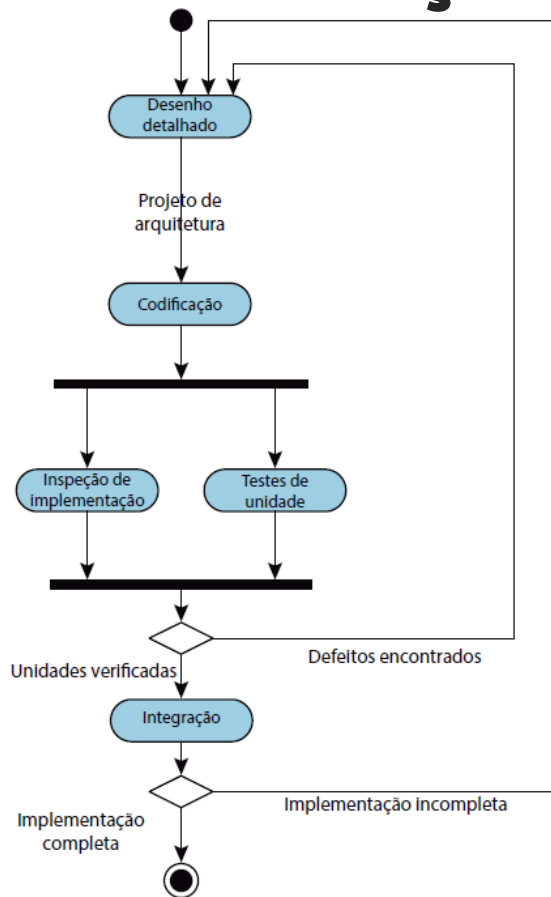
- Detalhamento dos componentes previstos no projeto;
- Descrição de todos os componentes de código fonte e código binário, em nível de linguagem de programação ou de acordo com as tecnologias escolhidas.

# ASPECTOS DE IMPLEMENTAÇÃO

1. **Reuso:** Maior uso possível de códigos já existentes.
2. **Gerenciamento de configuração:** observar as versões do sistema, utilizando um gerenciamento de configuração para o controle.
3. **Desenvolvimento host-target:** o desenvolvimento de um sistema ocorre em um computador (sistema host) e é executado em outro (sistema target), podendo ser do mesmo tipo ou muitas vezes diferentes.

# ATIVIDADES DA IMPLEMENTAÇÃO DE SOFTWARE

Fluxo desenvolvido na implementação de software



# FLUXO DE IMPLEMENTAÇÃO



Desenho detalhado: Traz uma visão mais clara sobre os elementos que foram desenhados em nível arquitetural;

Codificação: tradução do desenho detalhado no código de uma ou mais linguagens de programação. Utilização de IDE (Integrated Development Enviroment).

Ex: Eclipse, IntelliJ, VSCode



# FLUXO DE IMPLEMENTAÇÃO

- Revisão de código: eliminados os defeitos primários, como erros de digitação ou de uso da linguagem mesmo;
- Inspeção de implementação: verificação mais rigorosa o código por meio de um grupo de revisores;
- Testes de unidade: realizado em componentes de teste que devem também ter sido desenhados, revistos, codificados e compilados nas atividades anteriores

# FLUXO DE IMPLEMENTAÇÃO

Integração: liga as unidades implementadas com os componentes construídos em liberações anteriores, reutilizados de projetos anteriores ou adquiridos comercialmente;

*“O estágio de implementação do desenvolvimento de software é o processo de conversão de uma especificação do sistema em um sistema executável.”*

*(Sommerville)*

# CARACTERÍSTICAS DA IMPLEMENTAÇÃO DE SOFTWARE

- **Legibilidade:** o código deve ser facilmente lido e entendido.
- **Manutenibilidade:** o código deve ser facilmente modificado e mantido.
- **Desempenho:** códigos em que a execução seja a mais rápida possível.

# CARACTERÍSTICAS DA IMPLEMENTAÇÃO DE SOFTWARE

- **Rastreabilidade:** todos os elementos do código devem corresponder a um elemento do projeto.
- **Exatidão:** a implementação deve fazer aquilo que foi definido no levantamento de requisitos e no projeto.
- **Integridade:** todos os requisitos levantados devem ser atendidos.



# ESTILO DE PROGRAMAÇÃO E CODIFICAÇÃO

Identificar os recursos existentes na linguagem de programação utilizada

- Código claro
- Facilidade de manutenção

**Clean Code:** filosofia de desenvolvimento de softwares que consiste em aplicar técnicas simples que facilitam a escrita e a leitura de um código.

# ESTILO DE PROGRAMAÇÃO E CODIFICAÇÃO

Regras para uso de toda equipe, que especificam:

- Atribuição de nomes de variáveis;
- Endentações;
- Estilos de comentários;
- entre outros.

# EXEMPLOS

## Atribuição de nomes

- Escolha de nomes usados em variáveis, classes, métodos e outros elementos de programação
- Consistência: procure utilizar sempre nomes com a mesma palavra ou abreviação para um dado conceito e evite usar o mesmo nome ou abreviação para conceitos diferentes.

# EXEMPLOS

## **Separação de palavras e utilização de maiúsculas/minúsculas**

- Utilização de nomes compostos;
- Verificar padronizações da linguagem;
- Ex: camelCase; PascalCase, sneak\_case, kebab-case

# CONVENÇÕES JAVA

## camelCase:

- variáveis, atributos e métodos;

## PascalCase:

- classes, enum e interfaces;

## SCREAMING\_SNAKE\_CASE:

- constantes.

```
public class Pessoa{  
    public static void main(String[] args) {  
  
        String primeiroNome = "Maria";  
        int idade = 22;  
        double alturaAtual = 1.65;  
        final String MENSAGEM_PADRAO = "Olá";  
    }  
}
```

# EXEMPLOS

## Endentação e espaçamento

- acréscimo de espaços horizontais antes de algumas linhas para melhorar a estrutura do código;
- afeta a legibilidade e a manutenibilidade do código.

# EXEMPLOS

## Tamanho da função/método

- códigos com grandes funções ou métodos são mais propensos a erros do que os menores até certo ponto.

# EXEMPLOS

## Questões de atribuição de nomes de arquivo

- especificar como devem ser atribuídos os nomes de arquivos, facilita a localização de todos os elementos.



# EXEMPLOS

## Elementos particulares de programação

- linguagens de programação diferentes, possuem recursos diferentes;
- muitos destes recursos são considerados perigosos.

# COMENTÁRIOS

*“os comentários são muito importantes e podem ajudar ou prejudicar significativamente a legibilidade e a manutenibilidade”*

*Tsui e Karam (2013, p.136)*

Riscos:

- desviar a atenção do código e tornar o código mais difícil de ler;
- podem estar errados, aumentando a incidência de erros.

# COMENTÁRIOS

- Repetição do código
- Explicação do código
- Marcador no código
- Resumo de código
- Descrição do objetivo do código
- Referências externas

```
//  
// Dear maintainer:  
//  
// Once you are done trying to 'optimize' this routine,  
// and have realized what a terrible mistake that was,  
// please increment the following counter as a warning  
// to the next guy:  
//  
// total_hours_wasted_here = 16  
//
```

# COMENTÁRIOS

- Durante a execução de uma atividade, alguns erros podem ocorrer.
- Tratamento de exceções das linguagens é um avançado mecanismo que auxilia neste controle de erros e de mensagens.
- É necessário estendê-lo para criar um mecanismo capaz de controlar exceções de negócio.

**“Comentários redundantes são apenas lugares para coletar mentiras e desinformação.”**

Uncle Bob

**BONS ESTUDOS**