



Universidade Federal de Juiz de Fora  
Departamento de Ciência da Computação

DCC168 Teste de Software – Período 2025-3  
Prof. André Luiz de Oliveira

## Projeto: Teste Funcional, Teste Estrutural e Teste Baseado em Defeitos

### Objetivo

O objetivo do trabalho é aplicar os conceitos de Teste Funcional, Estrutural e Baseado em Defeitos utilizando ferramentas que apóiam a automação de cada uma dessas técnicas de teste. Dessa forma, serão utilizados o framework de Teste de Unidade [JUnit](#), as ferramentas de análise de cobertura de Teste Funcional (baseado na especificação) e Teste Estrutural (teste com base na estrutura do código do programa) [Eclemma](#) (Fluxo de Controle) e [Baduíno](#) (Fluxo de Dados) e a ferramenta de Teste de Mutação [PITest](#) de apoio à técnica de Teste Baseada em Defeitos. Materiais sobre cada uma dessas ferramentas serão disponibilizados nas páginas da disciplina no ambiente Google Classroom. O projeto deverá ser realizado em grupo de até 3 pessoas. Este trabalho está dividido em três partes: **I – Teste Funcional**, **II – Teste Estrutural** e **III – Teste Baseado em Defeitos**.

### Parte I: Teste Funcional

Nesta fase, a especificação do **jogo atribuído (Jogo do Aquário ou Jogo da Velha)** deverá ser analisada e os **critérios** de Teste Funcional: **Particionamento em Classes de Equivalência**, **Análise do Valor Limite** e **Grafo de Causa-Efeito** deverão ser aplicados para identificar as classes de equivalência válidas e inválidas e especificar o conjunto de casos de teste para testar o programa em cada classe de equivalência.

### Parte II: Teste Estrutural

Os critérios de Teste Estrutural baseados em **Fluxo de Controle** e **Fluxo de Dados** deverão ser aplicados para identificar e derivar os casos de teste para testar o **Jogo do Aquário** ou **Jogo da Velha**. Nesta fase, o **framework de testes JUnit** deverá ser utilizado para executar os casos de teste identificados a partir da aplicação dos critérios de **Teste Funcional** e **Teste Estrutural** e as ferramentas de análise de cobertura **Eclemma** e **Baduíno** deverão ser utilizadas na análise da cobertura dos casos de teste em relação aos critérios de teste estruturais baseados em **Fluxo de Controle** e **Fluxo de Dados** respectivamente.

### Parte III: Teste Baseado em Defeitos

Nesta fase, a técnica de Teste Baseada em Defeitos deverá ser aplicada utilizando o critério de **Teste Baseado em Mutação** (de **programas**) para identificar novos casos de teste que não foram identificados durante o Teste Funcional e Estrutural. A ferramenta de apoio ao critério de Teste de Mutação **PITest** deverá ser utilizada para automatizar a aplicação deste critério.

O trabalho proposto envolve a **implementação** (na linguagem **Java** utilizando o **Eclipse IDE**) e a aplicação das técnicas de Teste Funcional, Teste Estrutural e Teste Baseado em Defeitos no programa **Jogo do Aquário** ou do **Jogo da Velha**. As especificações de ambos os programas são descritas abaixo.

## Especificação do Programa: Jogo do Aquário

Considere um aquário representado por uma matriz bidimensional  $M \times N$ . Nesse aquário há dois tipos de peixes, os peixes do tipo A, que comem plâncton, e os peixes do tipo B que comem os peixes do tipo A. Os peixes seguem as seguintes regras:

### **Regras dos peixes do tipo A:**

1. Se houver uma célula livre à sua volta, movimentam-se para a célula livre.
2. Se se movimentarem durante **RA** vezes seguidas e se a sua volta houver uma célula livre, reproduzem-se ficando na mesma célula e o filho na célula livre.
3. Se não se movimentarem durante **MA** vezes seguidas, morrem de fome.

### **Regras dos peixes do tipo B:**

1. Se houver a sua volta algum peixe do tipo A, movimenta-se para lá e come-o. Senão, movimenta-se para uma célula livre.
2. Quando tiver comido **RB** peixes do tipo A e a sua volta não existir nenhum peixe do seu tipo e houver uma célula livre, reproduz-se ficando na mesma célula e o filho na célula livre.
3. Se durante **MB** vezes não comer nenhum peixe do tipo A, morre de fome.

## Detalhes de Implementação

O jogo deverá ser inicializado com os seguintes **parâmetros de entrada**: (1) *uma matriz bidimensional  $M \times N$  com  $X$  peixes do tipo A e  $Y$  peixes do tipo B*; e (2) os valores das variáveis **RA**, **MA**, **RB**, **MB**.  $X$ ,  $Y$ , **RA**, **MA**, **RB**, **MB** representam valores inteiros e positivos.

O jogador poderá ver o resultado de cada iteração e ir avançando para ver os resultados. Entende-se por uma iteração, uma movimentação de todos os peixes possíveis. A cada iteração, o programa deverá mostrar na tela o resultado e dar a opção ao jogador de continuar ou encerrar o jogo.

Ao final, o jogo deverá informar a pontuação obtida para um dado conjunto de entrada. A pontuação do jogo será o número de iterações ocorridas do início ao término do jogo. O jogo termina quando não houver mais peixes do tipo B ou o jogador encerrar o jogo. Com os parâmetros corretos o jogo pode nunca acabar, resultando em um ecossistema sustentável.

## Especificação do Programa: Jogo da Velha

O Jogo da Velha (Zaslavsky, 1982), também conhecido como Tic-Tac-Toe, é um de estratégia simples disputado entre dois jogadores, representados pelos símbolos “X” e “O”, cujo o objetivo é ser o primeiro a formar uma linha com três símbolos iguais, seja na horizontal, vertical ou diagonal em uma matriz de de dimensão 3 x 3.

Os jogadores se alternam para preencher as células da matriz com os símbolos “X” e “O” respectivamente. O jogo termina em empate se todas as células da matriz forem preenchidas sem um vencedor, ou seja, sem uma linha horizontal, vertical ou diagonal preenchida com símbolos iguais como ilustrado na Figura 1.



Figura 1. Jogo da Velha.

## Desenvolvimento da Atividade de Teste

### Parte I - Teste Funcional

Com base na análise da especificação do programa **Jogo do Aquário** ou **Jogo da Velha**, **identifique** as classes de equivalência válidas e inválidas, **derive** e **especifique** os casos de teste com base nos critérios de Teste Funcional: **Particionamento em Classes Equivalência**, **Análise dos Valores Limite** e na aplicação do **Grafo de Causa-Efeito**.

Como resultado desta atividade, elabore um documento contendo as seguintes informações: **tabela** (s) de **equivalência** gerada (s) em conformidade com o exemplo da **Tabela 1**, e a **tabela de casos de teste** que deverá conter as seguintes colunas: **identificador único** do caso de teste (**ID**), as **condições de entrada**, a **saída esperada**, as classes de equivalência exercitadas e a **saída obtida** como ilustrado na **Tabela 2**.

**Tabela 1** – Classes de Equivalência.

Condição de Entrada	Classes de Equivalência Válidas	Classes de Equivalência Inválidas
Cadeia de Caracteres (T)	$1 \leq T \leq 20$ ( <b>V1</b> )	$T < 1$ ( <b>I1</b> ) e $T > 20$ ( <b>I2</b> )
...	...	...
...	...	...

**Tabela 2** - Casos de Teste.

ID	Condições de Entrada	Saída Esp.	Classes Eq. Exercitadas	Saída Obtida
CT <sub>1</sub>	<valor <sub>1</sub> , valor <sub>2</sub> , valor <sub>n</sub> >	Valor (es)	V <sub>1</sub> , V <sub>2</sub> , V <sub>n</sub>	Valor (es)
CT <sub>2</sub>	<valor <sub>1</sub> , valor <sub>2</sub> , valor <sub>n</sub> >	Valor (es)	I <sub>1</sub>	Valor (es)
CT <sub>3</sub>	...	...	...	...
CT <sub>n</sub>	...	...	...	...

Os casos de teste gerados com a aplicação dos critérios funcionais correspondem ao conjunto de teste **TestSet-Func**.

**Data de Entrega:** 21/11/2025 até às 23:59.

## Parte II: Programa a ser Testado e Aplicação dos Critérios de Teste Estrutural

Implemente uma versão do programa **Jogo do Aquário** ou **Jogo da Velha** na linguagem **Java** utilizando o **Eclipse IDE**. O objetivo nesta fase é realizar a atividade de teste nas funcionalidades do **Jogo do Aquário** ou **Jogo da Velha** utilizando os critérios de Teste Estrutural baseados em **Fluxo de Controle** e de **Dados**, então não se preocupe com interface gráfica do programa. A saída do programa poderá ser na forma de exibição de uma matriz na tela (console) ou em arquivo texto.

### Parte II - A: Automatização do Teste Funcional

Implemente os casos de testes gerados pela aplicação dos critérios de Teste Funcional (**TestSet-Func**) utilizando o **framework JUnit**. Avalie a cobertura do conjunto de casos de teste funcional (**TestSet-Func**) para o programa **Jogo do Aquário** ou **Jogo da Velha** em relação aos critérios estruturais disponíveis nas ferramentas de análise de cobertura EclEmma (critérios baseados em fluxo de controle) e Baduíno (critérios baseados em fluxo de dados). Se defeitos forem identificados, deve-se apresentá-los no relatório. Corrija os defeitos no código fonte e reteste o programa. Importante: Nesta subfase, não adicione outros casos de teste, somente aqueles gerados pelo teste funcional.

### Parte II - B: Aplicação do Teste Estrutural

Por meio do uso das ferramentas de análise de cobertura de Teste Estrutural EclEmma e Baduíno, execute os casos de teste gerados anteriormente (casos de teste adequados ao Teste Funcional) e avalie a cobertura para os critérios disponíveis na ferramenta (consulte os slides com informações sobre EclEmma e Baduíno). Em seguida, adicione novos casos de teste de modo a melhorar a cobertura do Teste Estrutural, gerando o conjunto de teste **TestSet-Estr**.

O objetivo é definir casos de teste para obter 100% de cobertura para os critérios estruturais baseados em **Fluxo de Controle** e **Fluxo de Dados**. Gerar relatórios das ferramentas **EclEmma** e **Baduíno** com os resultados obtidos para entregar juntamente com a atividade. Caso defeitos sejam identificados nesta subfase, apresente-os no relatório. Corrija os defeitos e reteste o programa, considerando todos os casos de testes inseridos.

### Parte II - C: Elaboração de Relatório de Execução dos Testes

Elabore um relatório contendo os resultados da aplicação de cada Técnica de Teste (**Funcional** e **Estrutural**), incluindo coberturas de teste obtidas, relatórios gerados pelas ferramentas de análise de cobertura **EclEmma** e **Baduíno** e outras informações relevantes. Inclua uma análise pessoal sobre a eficiência das técnicas de Teste Funcional e Estrutural para encontrar defeitos no programa. Pergunta direcionada ao Grupo: *Ao desenvolver o programa, vocês sentiram que foram influenciados por terem criado os casos de teste funcionais primeiro (ou seja, aplicando o desenvolvimento dirigido por teste – do inglês – Test-Driven Development - TDD)? Se sim, como isso refletiu no código?*

**Data de Entrega:** 19/12/2025 até às 23:59.

## Parte III: Teste Baseado em Defeitos: Teste de Mutação

### Parte III - A: Avaliação da Qualidade do Conjunto de Casos de Teste Funcional e Estrutural

Utilizando os casos de teste gerados nas Partes II – A e II - B (adequados ao Teste Funcional e Estrutural), faça:

1. Execute o programa **Jogo do Aquário** ou **Jogo da Velha** na ferramenta **PITest** com o conjunto de casos de teste, aplicando todos os operadores de mutação (Full Mutation).
2. Gere o relatório sobre a qualidade desse conjunto de casos de teste: escore (score) de mutação e número de (programas) mutantes que permaneceram vivos (Mutation Coverage).
3. Caso defeitos sejam identificados nesta fase, apresente-os no relatório. Corrija os defeitos e reteste o programa, considerando todos os casos de testes inseridos.

### Parte III - B: Aplicação do Teste de Mutação

Com base nas informações anteriores, aplique o processo de execução do Teste de Mutação, criando novos casos de teste e analisando os programas mutantes que permaneceram vivos. Aplique todos os operadores de mutação. Utilize a ferramenta PITest.

1. Novos casos de teste deverão ser adicionados até todos os programas mutantes estarem mortos ou só restarem programas mutantes equivalentes (ou seja, **versões mutantes do programa original que já possuem casos de teste**).
2. Anote no relatório a **quantidade final** de casos de teste e o **total de programas mutantes equivalentes** gerados.
3. Escolha três programas mutantes equivalentes e explique no relatório a razão pelas quais eles são considerados equivalentes.
4. Caso defeitos sejam identificados nesta fase, apresente-os no relatório. Corrija os defeitos e reteste o programa, considerando todos os testes inseridos.
5. Gere o relatório da ferramenta PITest com os resultados obtidos para entregar juntamente como resultado da Parte III do trabalho.

### Parte III - C: Continuação do Relatório da Execução dos Testes

Complete o relatório da Parte II com os resultados da aplicação do critério de **Teste de Mutação**, incluindo o escore de mutação antes e no final, os relatórios gerados e outras informações que forem relevantes. Inclua uma análise pessoal sobre a eficiência da técnica de Teste de Mutação em encontrar defeitos no programa. Pergunta direcionada ao grupo: *Se os programas mutantes que têm comportamento diferente do programa original são mortos, como eu sei que o mutante que morreu não é a versão correta do programa? Como resolver isso?*

**Data de Entrega:** 21/01/2026 até às 23:59.

## Sobre a Entrega:

O relatório deverá ser elaborado e armazenado no Google Drive. O link de acesso ao documento deverá ser disponibilizado na entrega da atividade. Lembre-se de deixar permissão para que o professor possa ter acesso ao relatório no Google Drive (permissão para o email: [andre.oliveira@ufjf.br](mailto:andre.oliveira@ufjf.br)).

A entrega do trabalho deverá ser realizada via Google Classroom em arquivo compactado (.zip) com todos os artefatos requeridos em cada parte do trabalho.

Os arquivos .zip deverão conter: *i)* o **Projeto Java/Eclipse** do código fonte do programa e os **casos de teste JUnit**, *ii)* o **Relatório** gerado pelas ferramentas de análise de cobertura **EclEmma** e **Baduíno** para cada parte do trabalho, *iii)* o **Relatório** gerado pela ferramenta **PITest** (screenshots da **Eclipse View - PIT Summary**) e *iv)* o **Relatório Final do Projeto** (conforme descrito nas Partes II e III). Apenas um membro do grupo deverá realizar o envio. Coloque o nome de cada integrante do grupo.

## Referências

BADUINO. BA-DUA Plug-in for Eclipse. Disponível em: <https://github.com/saeg/baduino>. Acesso em: 27 de Outubro de 2025.

ECLEMMMA. EclEmma – Java Code Coverage for Eclipse. Disponível em: <https://www.eclEmma.org/>. Acesso em 27 de Outubro de 2025.

ECLIPSE. Instalação do Eclipse. Disponível em: <https://drive.google.com/file/d/1r5uwgoJw0MrSchqfKmHdOEm0tLD9hrnK/view?usp=sharing>. Acesso em 27 de Outubro de 2025.

JUNIT. JUnit download and install. Disponível em: <https://github.com/junit-team/junit4/wiki/Download-and-Install>. Acesso em 27 de Outubro de 2025.

PITEST. PIT Mutation Testing. Disponível em: <https://pitest.org/>. Acesso em: 27 de Outubro de 2025.

ZASLAVSKY, C. (1982). [Tic Tac Toe: And Other Three-In-A Row Games from Ancient Egypt to the Modern Computer](#). [S.l.]: Crowell. ISBN 0-690-04316-3.