

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL  
INSTITUTO DE INFORMÁTICA  
DEPARTAMENTO DE INFORMÁTICA APLICADA

# **INF01120 - Técnicas de Construção de Programas**

## **Aula 15: Lab 6 – Teste de Software**

Prof. Karina Kohl



# Introdução ao JUnit



# Introdução ao JUnit

- JUnit é um **framework open-source** de **teste** para programadores **Java**
  - Programadores podem criar casos de teste e testar seus próprios códigos
- Testes **unitários** visam mostrar que o programa funciona como o esperado
- O pacote `org.junit` contém diversas interfaces e classes para JUnit testing:
  - `Assert`, `Test`, `Before`, `After`
- Atualmente, o JUnit se encontra na versão 5 (**JUnit 5**)

# Introdução ao JUnit

- IDEs com suporte ao JUnit 5:
  - IntelliJ IDEA
  - Eclipse
  - NetBeans
  - VS Code
- Instalação:
  - Seguir os passos em <https://junit.org/junit5/docs/current/user-guide/> de acordo com a IDE/building tool desejada

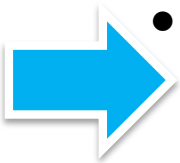
## PARA O LAB:

Faremos por linha de comando...  
Instruções em alguns instantes ;-)

JUNIT

# Introdução ao JUnit

- Tipos de teste (execução):
  - **Manual**: executa testes manualmente (“time consuming and less reliable”)
  - **Automatizado**: executa testes com suporte de ferramenta (“fast and more reliable”)
- Definição de testes:
  - Um teste unitário em JUnit é um **método** em uma **classe** usada **apenas para teste**
  - Classes de teste\* geralmente seguem a convenção **ClassNameTest**



\* Em alguns casos é comum ter **mais de uma classe teste para cada classe**  
(cada classe de teste procura encontrar erros específicos da classe)

# Introdução ao JUnit

- Importação do pacote JUnit:
  - Boa prática é importar os `assertStatements` como `estáticos`
    - Tornam os testes `menores` e `mais fáceis` de serem lidos

```
import static org.junit.Assert.*;  
Assert.assertEquals() -> assertEquals()
```

- Principais notações:
  - `@Test`: define um método como teste
  - `@BeforeEach` e `BeforeAll`: roda o método antes de cada teste / uma vez apenas
  - `@AfterEach` e `AfterAll`: roda o método depois de cada teste
  - `@DisplayName`: define o nome do teste a ser mostrado para o usuário
  - `@RepeatedTest(N)`: define que o método será executado N vezes
  - `@Disabled`: desabilita um método/classe de teste

# Introdução ao JUnit

- Principais assert statements: inclusos pelo pacote `org.junit.Assert.*`;
  - `assertEquals()` e `assertNotEquals()`
  - `assertSame()` e `assertNotSame()`: o esperado e o real se referem ao mesmo objeto.
  - `assertFalse()` e `assertTrue()`
  - `assertNull()`
  - `assertAll()`: agrupa diferentes testes para serem testados
  - `assertThrows()`: afirma que o método vai gerar uma exceção
  - `assertArrayEquals()`: afirma que dois arrays são iguais (mesmos elementos e posições)
  - `assertTimeout(timeout, exec)`: afirma que `exec` completa sua execução antes que o `timeout` seja excedido

<https://junit.org/junit5/docs/5.0.1/api/org/junit/jupiter/api/Assertions.html>

JUNIT

# JUnit

- **Tutoriais JUnit:**

- <https://junit.org/junit5/docs/current/user-guide/>
- <https://www.vogella.com/tutorials/JUnit/article.html>
- <https://www.baeldung.com/junit-5>
- <https://stackabuse.com/unit-testing-in-java-with-junit-5/>
- <https://blog.jetbrains.com/idea/2020/09/writing-tests-with-junit-5/>
- <https://www.jetbrains.com/help/idea/junit.html>
- <https://www.youtube.com/watch?v=2E3WqYupx7c&list=PLqq-6Pg4ITTa4ad5JISViSb2FVG8Vwa4o>
- <https://www.youtube.com/watch?v=ZjZ20BgCL0o&list=PL6Zs6LgrJj3tE9xbgcz16sNbScYkrtce7>
- <https://www.youtube.com/watch?v=W3fSgHrBzek>
- <https://www.youtube.com/watch?v=1siXcUpsldO>



# Lab 5



## **LAB 6**

# **Exercício 1 – Parte 0**

- **Familiarização com JUnit5**
  - **Pegue o link do TCP-Lab-6 no Moodle**
  - **Faça o download para sua máquina**

# Exercício 1 – Parte 0

- Familiarização com JUnit5
  - <https://junit.org/junit5/docs/current/user-guide/#running-tests-console-launcher>
  - O **ConsoleLauncher** é um aplicativo Java de linha de comando que permite que você inicie o JUnit a partir do terminal. Pode ser usado para executar testes JUnit e imprimir resultados de execução de teste no console.
  - Um Fat JAR executável que contém o conteúdo de todas as suas dependências é publicado no repositório Maven Central sob o diretório junit-platform-console-standalone. Para o nosso lab, ele já está no projeto, na pasta lib.
    - **lib/junit-platform-console-standalone-1.11.0.jar**

# Exercício 1 – Parte 0

- **Familiarização com JUnit5**

- **Vá até o diretório onde você fez o download do projeto template do github**

- **Compile as classes (incluindo a de teste) com o seguinte comando**

- `javac -cp "lib/junit-platform-console-standalone-1.11.0.jar" src/*.java`

- **As classes serão criadas**

- **Rode os testes usando o Console Launcher**

- `java -jar lib/junit-platform-console-standalone-1.11.0.jar execute --class-path src --scan-class-path`

- **A saída no terminal será semelhante a seguinte:**

## LAB 6

# Exercício 1 – Parte 0

```
192:TCP-Lab-6 karina.kohl$ javac -cp "lib/junit-platform-console-standalone-1.11.0.jar" src/*.java
192:TCP-Lab-6 karina.kohl$ java -jar lib/junit-platform-console-standalone-1.11.0.jar execute --class-path src --scan-class-path

♥ Thanks for using JUnit! Support its development at https://junit.org/sponsoring

└─ JUnit Platform Suite ✓
   └─ JUnit Jupiter ✓
      └─ BibliotecaTest ✓
         └─ testAdicionarLivro() ✓
            └─ testAdicionarLivroNulo() ✓
   └─ JUnit Vintage ✓

Test run finished after 134 ms
[ 4 containers found ]
[ 0 containers skipped ]
[ 4 containers started ]
[ 0 containers aborted ]
[ 4 containers successful ]
[ 0 containers failed ]
[ 2 tests found ]
[ 0 tests skipped ]
[ 2 tests started ]
[ 0 tests aborted ]
[ 2 tests successful ]
[ 0 tests failed ]
```

# Exercício 1 – Parte 0

- **Familiarização com JUnit5**
  - Se familiarize com os códigos antes de ir para a parte 1.
    - Estrutura básica de teste com anotação @Test
    - Nomes de exibição para melhor relatório de teste
    - Diferentes tipos de asserções
    - Teste de exceção
    - Teste de um caso extremo (divisão por zero)
  - Como faria um teste falhar?

# Exercício 1 – Parte 1

Na classe `BibliotecaTest.java`, escreva testes adicionais cobrindo os seguintes cenários:

- ✓ Teste a adição de livros (já está no exemplo)
- Teste a remoção de livros
- Teste a busca de livros por ISBN e por título
- Teste o registro de usuários
- Teste o empréstimo e devolução de livros
- Teste a listagem de livros disponíveis
- Teste a listagem de empréstimos ativos
- Teste os casos de erro (ex: empréstimo de livro não disponível, devolução de livro não emprestado, etc.)

## Exercício 1 – Parte 2

- **Compile e rode os testes na linha de comando**
  - **Faça um print do resultado e faça o upload para o seu repositório juntamente com o Código que você implementou dos testes (commit e push do BibliotecaTest.java)**



**UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL  
INSTITUTO DE INFORMÁTICA  
DEPARTAMENTO DE INFORMÁTICA APLICADA**

**Dúvidas?**

**Prof. Karina Kohl**

