03. Distribuição do seu código

Date	@04/10/2022
Categoria	Java
Curso	Java e java.lang: programe com a classe Object e String

Tópicos

- Conhecendo o Javadoc
- Qual comentário?
- Sobre o javadoc
- Para saber mais: Todas as tags
- Criando uma biblioteca com JAR
- O que é um JAR?
- JAR executável (Opcional)
- Mão na massa: javadoc
- Mão na massa: jar
- Para saber mais: Maven
- O que aprendemos?
- Arquivos do projeto atual

Para saber mais: Todas as tags

Já vimos nessa aula algumas tags (ou anotações)
do *javadoc* como `@version` ou `@author`. Segue a lista completa:

- `@author` (usado na classe ou interface)
- `@version` (usado na classe ou interface)
- `@param` (usado no método e construtor)
- `@return` (usado apenas no método)
- `@exception` ou `@throws` (no método ou construtor)
- `@see`
- `@since`
- `@serial`
- `@deprecated`

Importante é que as tags do *javadoc* existem apenas para padronizar alguns dados fundamentais do seu código fonte como o autor e a versão.

Outras anotações

Nos cursos você também já viu uma anotação fora do *javadoc*, a anotação `@Override`. Essa anotação é considerada uma configuração, nesse caso interpretado pelo compilador.

As anotações vão muito além das tags *javadoc* e são muito mais sofisticadas e poderosas. Elas só entraram na plataforma Java a partir da versão 1.5 enquanto o *javadoc* está presente desde o nascimento da plataforma Java. O interessante é que as anotações foram inspirados pelas tags do *javadoc*.

Se você ainda não está seguro sobre o uso das anotações, fique tranquilo pois você verá ainda muitas usadas pelas bibliotecas por ai para definir dados e configurações. Aguarde!

Para saber mais: Maven

Java é uma plataforma de desenvolvimento completa que se destaca com sua grande quantidade de projetos *open source*. Para a maioria dos problemas no dia a dia do desenvolvedor já existem bibliotecas para resolver. Ou seja, se você gostaria de se

conectar com um banco dados, ou trabalhar no desenvolvimento web, na área de data science, criação de serviços ou Android, já existem bibliotecas para tal, muitas vezes mais do que uma.

Aí existe a necessidade de organizar, centralizar e versionar os JARs dessa biblioteca e gerenciar as dependências entre elas. Para resolver isso, foram criadas ferramentas especificas e no mundo Java se destacou o Maven. O Maven organiza os JARs (código compilado, código fonte e documentação) em um repositório central que é publico e pode ser pesquisado:

https://mvnrepository.com/

Lá você pode ver e até baixar os JARs, mas o melhor é que a ferramenta Maven pode fazer isso para você. Se ficou interessado em aprender o Maven que ainda tem outros recursos bem legais, dá uma olhada no nosso curso especifico:

Maven: Build do zero a web

Obs: Se você é usuário Linux, o Maven é bem parecido com os gerenciadores `apt` ou `rpm`. No MacOS existe o `brew` com o mesmo propósito. No mundo .Net temos o `nuget` e a plataforma node.js usa `npm`. Gerenciar dependências é um problema do cotidiano do desenvolvedor, e cada sistema ou plataforma possui a sua solução.

O que aprendemos?

Nessa aula mais leve vimos e aprendemos:

- quais comentários e tags (anotações) a usar para definir o *javadoc*
- como gerar o *javadoc* no Eclipse
- que *javadoc* é uma documentação para desenvolvedores
- que as classes Java padrão também usam *javadoc*
- como criar nossa própria biblioteca através do JAR (**J*ava *AR**chive)
- como importar a biblioteca no novo projeto

- como criar um JAR executável

Na próxima aula vamos conhecer o pacote `java.lang`.