

53 45 52 45 49 20 46 49 45 4c 20
41 4f 53 20 50 52 45 43 45 49 54
4f 53 20 44 41 20 48 4f 4e 52 41
20 45 20 44 41 20 43 49 c3 8a 4e
43 49 41 2c 20 50 52 4f 4d 4f 56
45 4e 44 4f 20 4f 20 55 53 4f 20
45 20 4f 20 44 45 53 45 4e 56 4f
4c 56 49 4d 45 4e 54 4f 20 44 41
20 49 4e 46 4f 52 4d c3 81 54 49
43 41 20 45 4d 20 42 45 4e 45 46
c3 8d 43 49 4f 20 44 4f 20 43 49
44 41 44 c3 83 4f 20 45 20 44 41
20 53 4f 43 49 45 44 41 44 45 2e

RESIDÊNCIA DE SOFTWARE

**CAPACITAR
TREINAR
EMPREGAR**

TRANSFORMAR



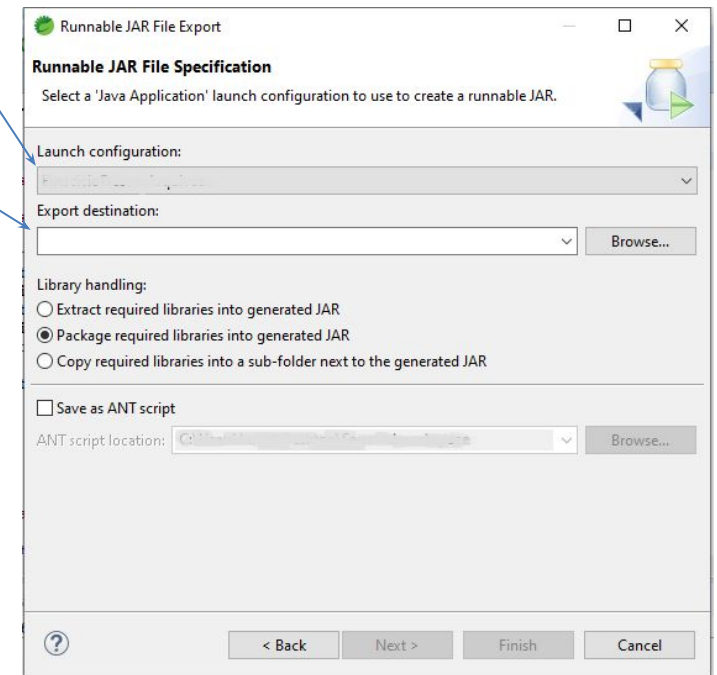
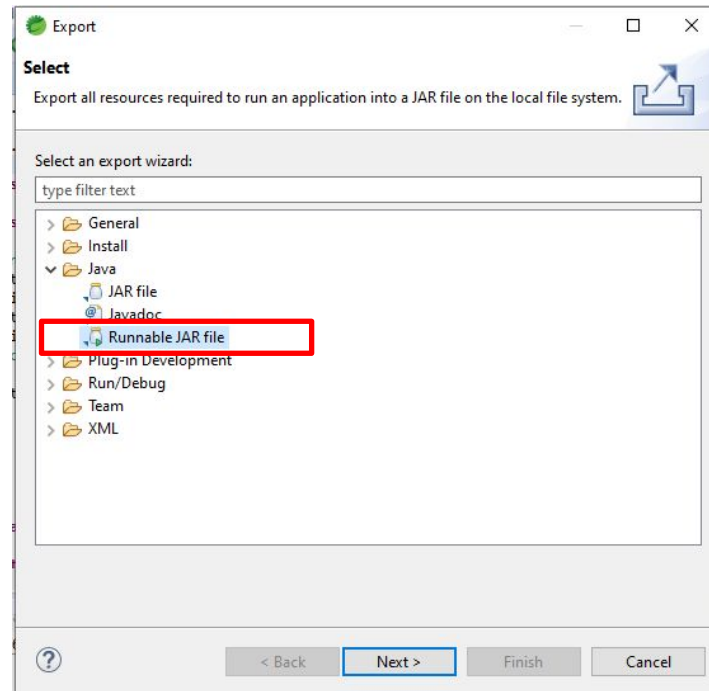
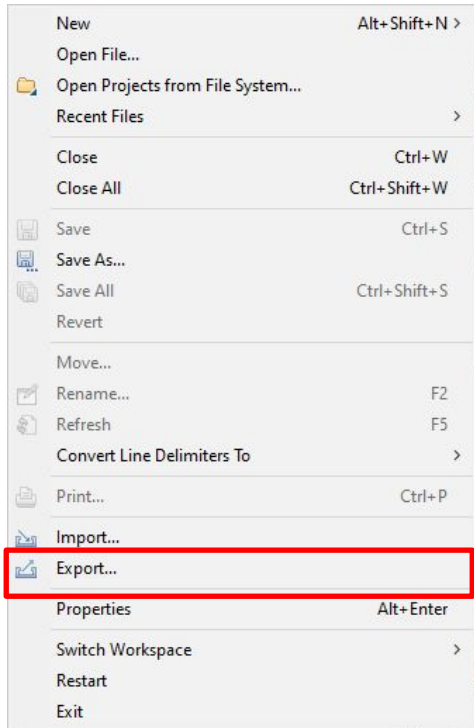
JAVA I
Jar, Javadoc
13/08/2020

ARQUIVO JAR

- Java é uma linguagem portátil, ou seja, é multi-plataforma
- O compilador Java não cria nenhum arquivo executável (.exe)
- Para os sistemas operacionais os arquivos .java são vistos da mesma forma
- Para a JVM, os arquivos .class são visto da mesma fora
- Um arquivo .jar (**J**ava **AR**chive), tem o mesmo funcionamento de um arquivo executável

CRIANDO ARQUIVO JAR

- A criação do arquivo .jar pode ser feita através do terminal utilizando o comando “jar”
- As IDEs dão suporte para a criação
- Um jar pode ser uma aplicação, uma biblioteca, uma instalação...

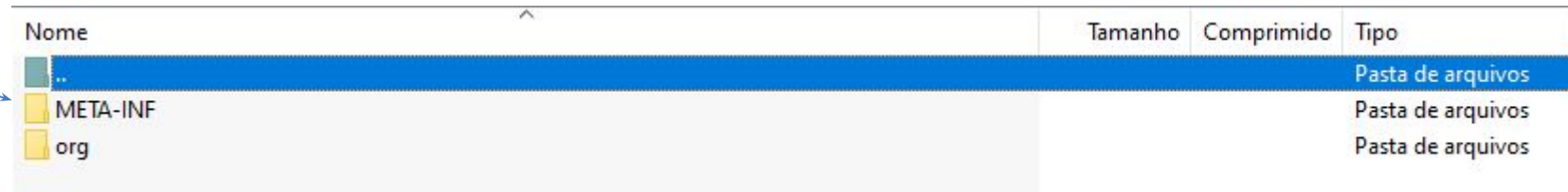


CRIANDO ARQUIVO JAR

- Um arquivo jar possui um arquivo manifesto localizado META-INF/MANIFEST.MF. As entradas do arquivo manifesto determinam como o arquivo jar será usado. Arquivos jar para serem executáveis (como os “.exe” do Windows) terão uma de suas classes especificadas como a classe "principal", a classe que contém a “main”
- Comando para executar o jar: “java -jar <nome_do_arquivo.jar>”

Jar pode ser aberto com programas de compactação

Manifesto



Nome	Tamanho	Comprimido	Tipo
..			Pasta de arquivos
META-INF			Pasta de arquivos
org			Pasta de arquivos

- Permite escrever comentários no código que serão usados para gerar documentação textual que pode ser entregue ao usuário

The screenshot shows a code editor with the following snippet:

```
public static void main(String[] args) {  
    System.out.
```

A tooltip is displayed for the `append` method. The tooltip is divided into two main sections:

- Method List:** A list of methods available for `System.out`, with `append(char c) : PrintStream - PrintStream` selected.
- Method Details:** A detailed description of the `append` method, including its purpose, usage, and exceptions.

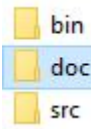
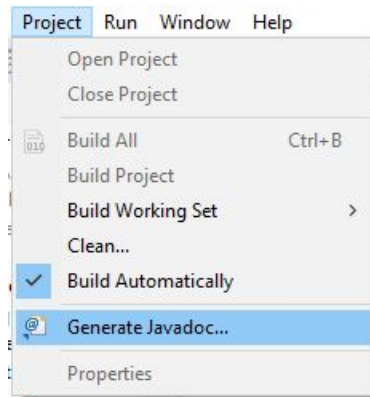
Annotations on the image:

- A blue box labeled "Método" points to the `append` method in the list.
- A blue box labeled "Documentação sobre o método" points to the detailed description of the `append` method.

Method Details:

- Append:** Appends the specified character to this output stream.
- Usage:** An invocation of this method of the form `out.append(c)` behaves in exactly the same way as the invocation `out.print(c)`.
- Specified by:** `append(...)` in `Appendable`
- Parameters:**
 - `c` The 16-bit character to append
- Returns:** This output stream
- Throws:** `IOException` - If an I/O error occurs
- Since:**

- Permite escrever comentários no código que serão usados para gerar documentação textual que pode ser entregue ao usuário
- Observar que tag html podem ser utilizadas
- Utilizamos anotações próprias para descrever parâmetros retornos, autor, exceções entre outros



Pasta com documentação

```
/**
 * Appends the specified character sequence to this output stream.
 *
 * <p> An invocation of this method of the form <tt>out.append(csq)</tt>
 * behaves in exactly the same way as the invocation
 *
 * <pre>
 *     out.print(csq.toString()) </pre>
 *
 * <p> Depending on the specification of <tt>toString</tt> for the
 * character sequence <tt>csq</tt>, the entire sequence may not be
 * appended. For instance, invoking then <tt>toString</tt> method of a
 * character buffer will return a subsequence whose content depends upon
 * the buffer's position and limit.
 *
 * @param csq
 *     The character sequence to append. If <tt>csq</tt> is
 *     <tt>null</tt>, then the four characters <tt>"null"</tt> are
 *     appended to this output stream.
 *
 * @return This output stream
 *
 * @since 1.5
 */
public PrintStream append(CharSequence csq) {
```

- Os trechos de javadoc são blocos com as seguintes delimitações `/** */`
- As tags são inseridas dentro do bloco de comentários, antecedidas pelo caracter `@` e após o nome da tag, segue o conteúdo que deseja descrever

Tag	Significado
@author	Especifica o autor da classe ou do método em questão.
@deprecated	Identifica classes ou métodos obsoletos. É interessante informar nessa tag, quais métodos ou classes podem ser usadas como alternativa ao método obsoleto.
@link	Possibilita a definição de um link para um outro documento local ou remoto através de um URL.
@param	Mostra um parâmetro que será passado a um método.
@return	Mostra qual o tipo de retorno de um método.
@see	Possibilita a definição referências de classes ou métodos, que podem ser consultadas para melhor compreender idéia daquilo que está sendo comentada.
@since	Indica desde quando uma classe ou métodos foi adicionado na aplicação.
@throws	Indica os tipos de exceções que podem ser lançadas por um método.
@version	Informa a versão da classe.

EXERCÍCIO

- Criar uma classe funcionário com método aumentarSalario(double aumento) e adicionar os javadoc para a classe e método

```
/**
 * Classe que representa um funcionario
 * @author Serratec
 *
 */

public class Funcionario {

    /**
     * Aplica um aumento ao salario do funcionario
     *
     * @param aumento - aumento a ser aplicado no salario do funcionario
     *
     * @throws IllegalArgumentException
     * Se o aumento < 0
     *
     */
    public void aumentarSalario(double aumento) throws IllegalArgumentException{
```