

Fundamentos Java



Prof. Dr. João Paulo Lemos Escola
Copyright© 2022

Tópicos da aula

- Exportando projeto para zip e jar;
- Classe principal e método principal;
- Argumentos de linha de comandos;
- Comentários em código Java;
- Tipos de dados primitivos;
- Variáveis e constantes;
- Operadores e Strings;
- Conversões de tipos.

Gerando um .jar

- Como vimos, um arquivo .jar é um programa executável resultante da compilação de um projeto Java;
- Clicando no botão “Build”, será mostrada uma mensagem sobre como executar o .jar resultante:



```
Output - JavaApplication1 (cleanjar) x
Created dir: /home/jpescola/NetBeansProjects/JavaApplication1/build/generated-sources/ap-source-output
Compiling 1 source file to /home/jpescola/NetBeansProjects/JavaApplication1/build/classes
compile:
Created dir: /home/jpescola/NetBeansProjects/JavaApplication1/dist
Copying 1 file to /home/jpescola/NetBeansProjects/JavaApplication1/build
Nothing to copy.
Building jar: /home/jpescola/NetBeansProjects/JavaApplication1/dist/JavaApplication1.jar
To run this application from the command line without Ant, try:
java -jar "/home/jpescola/NetBeansProjects/JavaApplication1/dist/JavaApplication1.jar"
deploy:
jar:
BUILD SUCCESSFUL (total time: 0 seconds)
```

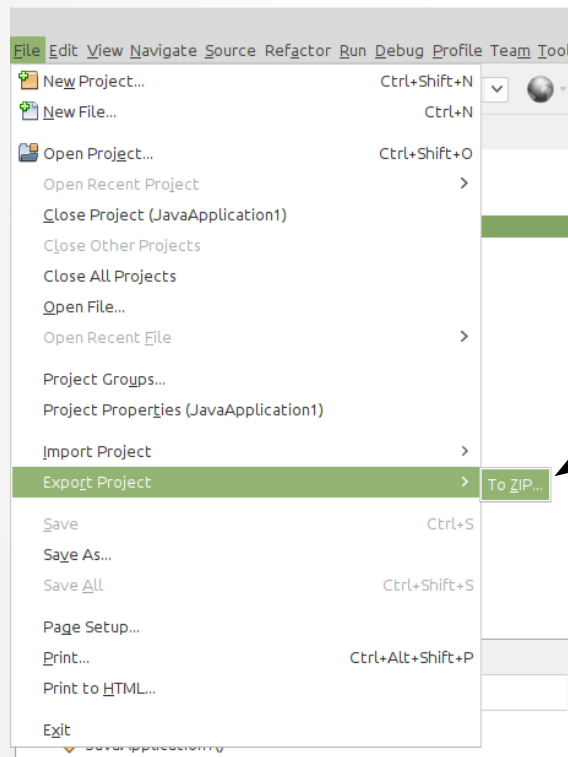
Limpar e Construir

- O botão do martelo com a vassoura serve para limpar o projeto e depois criar o .jar novamente.
- Os arquivos da pasta “dist” são excluídos e criados novamente.



Exportar para .zip

- Para enviar um projeto de tarefa, utilize a opção do menu File > Export Project > to zip:



Classe principal

- Todo projeto necessita de uma classe principal;
- A classe principal possui o método main:

```
public static void main(String[] args){  
    // código do método principal  
}
```

Detalhando o método main

- **public**: método público, pode ser acessado por outras classes do projeto, em arquivos diferentes;
- **static**: método é estático, ou seja, não precisamos instanciar um objeto da classe para poder utilizar este método;
- **void**: método não tem retorno, não pode haver o comando 'return' ao final do método;
- **main**: nome do método. Toda classe principal precisa ter um método com esse nome. Esse método deve receber como parâmetro um vetor de String;
- **String[] args**: o método recebe como parâmetro um vetor de String, que pode ser utilizado no escopo do método pelo nome da variável 'args'. Este nome pode ser qualquer um, exceto palavras reservadas.

Empregando argumentos

```
// mostra um texto
```

```
System.out.print("primeiro argumento: ");
```

```
// mostra o primeiro argumento
```

```
System.out.print(args[0]);
```

```
// mostra um texto e o segundo argumento concatenados
```

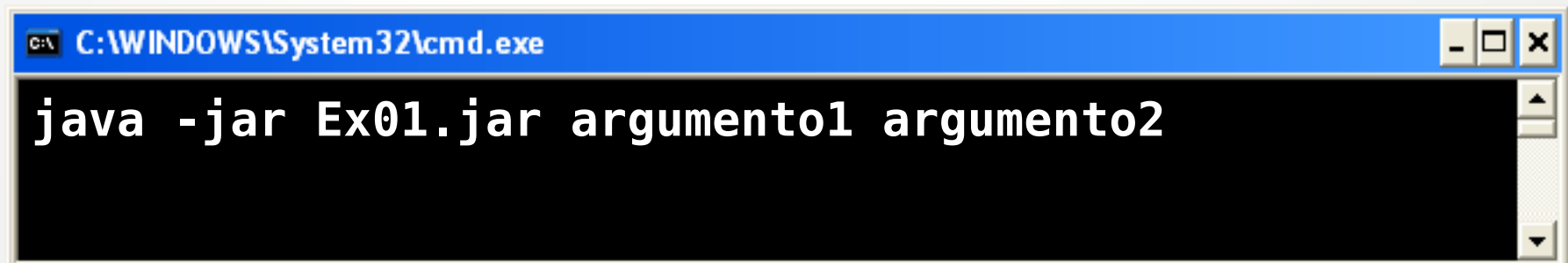
```
System.out.println("segundo argumento: "+args[1]);
```

```
// mostra um texto e o segundo argumento concatenados
```

```
System.out.printf("segundo argumento: %s \n", args[1]);
```


Como executar um jar

- Crie um projeto com o nome “Ex01”;
- Adicione o comando para imprimir os primeiros dois argumentos;
- Clique no ícone de build (F11) para criar o jar;
- Entre na pasta do projeto;
- Abra o prompt de comandos e execute o comando abaixo:

A screenshot of a Windows command prompt window. The title bar is blue and contains the text "C:\WINDOWS\System32\cmd.exe" and standard window control buttons (minimize, maximize, close). The main area of the window is black with white text. The text displayed is the command "java -jar Ex01.jar argumento1 argumento2".

```
C:\WINDOWS\System32\cmd.exe  
java -jar Ex01.jar argumento1 argumento2
```

Build no Netbeans

- Vimos que o build resulta no seguinte:



```
Output - JavaApplication1 (clean.jar) x
Created dir: /home/jpescola/NetBeansProjects/JavaApplication1/build/generated-sources/ap-source-output
Compiling 1 source file to /home/jpescola/NetBeansProjects/JavaApplication1/build/classes
compile:
Created dir: /home/jpescola/NetBeansProjects/JavaApplication1/dist
Copying 1 file to /home/jpescola/NetBeansProjects/JavaApplication1/build
Nothing to copy
Building jar: /home/jpescola/NetBeansProjects/JavaApplication1/dist/JavaApplication1.jar
To run this application from the command line without Ant, try:
java -jar "/home/jpescola/NetBeansProjects/JavaApplication1/dist/JavaApplication1.jar"
deploy:
jar:
BUILD SUCCESSFUL (total time: 0 seconds)
```

- Basta copiar e colar o comando no prompt.

A02ex01

- Crie um programa que receba um argumento na linha de comandos e retorne a mensagem conforme o exemplo:
 - Se a linha de comandos for **java -jar A02ex01.jar 1**, o resultado deve ser “O argumento passado foi: 1”
- Sugestão:
 - Crie uma pasta na raiz da unidade “C:” ou utilize uma pasta já criada no ambiente;
 - Crie os projetos dentro dessa pasta;
 - Entre na pasta e execute o comando com os argumentos.

A02ex02

- Crie um novo projeto, que mostre os três argumentos da linha de comandos.

Comentários

- Comentário de uma linha:

```
// isto é um comentário
```

- Comentário de mais de uma linha:

```
/* isto é  
um comentário  
de várias  
linhas */
```

Tipos de dados

- 8 bits:
 - boolean e byte;
- 16 bits:
 - char e short;
- 32 bits:
 - int e float;
- 64 bits:
 - long e double.

Declaração de variáveis

(A02ex03.jar)

```
public static void main(String[] args){  
    char sexo = 'f'; // char ou String  
    double nota = 5.5; // float ou double  
    int alunos = 100, classes = 10; // int, short ou long  
    boolean ativado = false;  
    System.out.println("sexo: "+sexo);  
    System.out.println("nota: "+nota);  
    System.out.println("alunos: "+alunos);  
    System.out.println("classes: "+classes+" situação? "+ativado);  
}
```

Constantes

- A diferença entre as variáveis e as constantes é que uma variável pode ter seu valor alterado durante o código, já uma constante deve ter um valor fixo do início ao fim do programa:

```
int idade=10; // declaração de variável
```

```
final int IDADE=10; // declaração de constante
```


Operadores Aritméticos

- Adição:

$x = 1+1;$

- Subtração:

$x = 1-1;$

- Multiplicação:

$x = 1*1;$

- Divisão:

$x = 1/1;$

Operadores Aritméticos (cont.)

- Resto da divisão:

`x = y%z;`

- Sinal negativo:

`x = -x;`

- Sinal positivo:

`x = +x;`

- Incremento unitário:

`x = ++x; ou`

`x = x++;`

- Decremento unitário:

`x = --x; ou`

`x = x--;`

Exemplo de operadores aritméticos

(A02ex04.jar)

```
public static void main(String[] args){  
    int x = 10;  
    int y = 3;  
    System.out.println("x="+x);  
    System.out.println("y="+y);  
    System.out.println("-x="+(-x));  
    System.out.println("x/y="+(x/y));  
    System.out.println("Resto de x por y="+(x%y));  
    System.out.println("Incremento de x="+(++x));  
}
```

String

- Podemos criar variáveis para armazenar conjuntos de letras, ou seja, palavras ou frases, utilizando o tipo de dados String;
- String na verdade é uma classe Java;
- É como um vetor de caracteres (char), mas muito mais fácil de manipular.

Conversão de tipos

- Para converter valores usamos o tipo de dado entre parênteses (casting) ou então um método próprio:

```
// converte o valor de x para float e armazena em y;  
y = (float)x;  
// método de conversão de String para inteiro (int)  
y = Integer.parseInt(x);  
// método de conversão de String para número real (float)  
y = Float.parseFloat(x);  
// método de conversão de String para número real (double)  
y = Double.parseDouble(x);  
// método de conversão de número para String  
y = String.valueOf(x); ou  
y = ""+x;
```

Exemplo de conversão de tipos

(A02ex05.jar)

```
public static void main(String[] args){  
    int x = 10;  
    double y = 3.5;  
    char z = 'a';  
    String v = "3.5"  
  
    n = (int)y;  
    System.out.println("n = "+n);  
    n = Float.parseFloat(v);  
    System.out.println("n = "+n);  
    float a = (float)x;  
    String idade = ""+x;  
    System.out.println("Valor de a: "+a+", idade="+idade);  
}
```

A02ex06

- Vamos criar um programa que receba dois inteiros como argumento:
 - O resultado deve ser a soma dos dois argumentos.

O que aprendemos?

- Exportando projeto para zip e jar;
- Classe principal e método principal;
- Argumentos de linha de comandos;
- Comentários em código Java;
- Tipos de dados primitivos;
- Variáveis e constantes;
- Operadores e Strings;
- Conversões de tipos.

Na próxima aula...

- Modelos de código;
- Janelas gráficas;
- Sobrecarga de métodos;
- Estruturas de decisão.