Usando a Linha de Comando (Shell)

Programação Orientada a Objetos

O objetivo deste tutorial é apresentar os comandos básicos necessários para compilar e executar programas Java usando uma janela de "linha de comando" (ou "shell" https://en.wikipedia.org/wiki/Shell_(computing)) do sistema operacional. Neste tutorial são apresentados os comandos básicos de "Unix" que neste nível são os mesmos da maioria das distribuições de Linux. Se estiveres usando Windows, a sugestão é instalar o WSL (Windows Subsistem for Linux), ou então, procurar documentação sobre o "power shell" do Windows.

Ainda no Windows, além do WSL, sugere-se instalar o "Windows Terminal" que permite abrir facilmente janelas de shell de diferentes tipos (Linux, DOS, Power Shell etc). Para tanto acesse:
https://apps.microsoft.com/store/detail/windows-terminal/9N0DX20HK701?hl=pt-br&gl=BR

Nomes de arquivos e "wildcards"

Por razões históricas os nomes de arquivo são formados por [nome].[extensão de 3 letras]. Atualmente essa regra não vale mais, sendo que a maioria dos sistemas operacionais permite bastante liberdade nos nomes de arquivo. Uma restrição em arquivos Unix é que os arquivos com nome começado por "." é um "arquivo oculto" que só é exibido quando usamos a opção "-a" do comando "ls" (ver na sequência).

Quando queremos nos referenciar a mais de um arquivo por vez podemos usar os seguintes "wildcards":

"?" – substitui uma letra
"*" – substitui várias letras

Exemplos:

"ls A?p.java" → lista todos os arquivos ".java" que começam por "A", tem uma letra qualquer na sequência e se encerram por "p".
"ls A*p.java"→ lista todos os arquivos ".java" começados por "A" e terminados por "p".
"ls *.java" → lista todos os arquivos de terminação ".java"
"ls App.*" → lista todos os arquivos chamados "App" com qualquer terminação.

Os wildcards se aplicam a todos os comandos Unix.

Comandos básicos do Unix

ls – lista diretórios e arquivos (opções: -las → ls -las)

cp – copia arquivos de um local para outro

rm – remove arquivos e diretórios (cuidado com a opção rm -r)

mv – (move) usado para renomear ou trocar os arquivos de lugar

chmod – altera as permissões de um arquivo ou diretório. As permissões são: r, w, x e podem ser atribuídas a outros (others) ao usuário (user) ou ao grupo (group)

Ex: sudo chmod u=rwx,g=rx,o=r Senhas

Neste exemplo o usuário pode tanto ler (r) como escrever (w) no arquivo chamado “Senhas”, bem como executar (x) este arquivo. Já os usuários associados ao grupo podem apenas ler e executar e os demais (outros) apenas ler.

cat – apresenta o conteúdo de um arquivo texto no terminal

more – faz a paginação do texto de entrada

less – opção mais moderna de paginação

head – mostra as primeiras 10 linhas de um texto

tail – mostra as 10 últimas linhas de um texto

grep – procura por padrões em um arquivo texto

cd – muda o diretório

pwd – exibe o diretório corrente

mkdir – cria um diretório

Outros comandos Unix podem ser consultados em https://www.unixtutorial.org/basic-unix-commands

Código Fonte

Para criar um programa em Java primeiro precisamos escrever o seu código fonte. Um arquivo de código fonte Java não é muito diferente de um arquivo de texto comum, basta salvá-lo com a extensão **.java**.

O exemplo a seguir demonstra como criar um arquivo fonte chamado “Main.Java” no sistema operacional Linux.

1. Abra o Terminal de comandos
2. Usando o comando **cd** (*change directory*), caminhe até a pasta desejada
3. Use o comando **touch App.java** para criar tal arquivo (cria o arquivo vazio)
4. Digite **nano App.java** (abre o editor de texto nano)
5. O Editor de Texto invocado pelo comando **nano** se assemelha ao Bloco de Notas do Windows, ou seja, é um editor de texto que não permite formatação, estilos, etc. Usando o **nano**, digite o seguinte código Java:

```
public class Main {
  public static void main(String[] args) {
    System.out.println("Hello World!");
  }
}
```

Vale ressaltar que o nome dado à classe deve ser o mesmo que o nome do arquivo, ou seja, não podemos criar a classe *Entrada* no arquivo *Main.java*. Agora, basta salvar o arquivo.

Compilando

Como já sabemos, para executar o programa, primeiro precisamos compilar o código fonte. Para isso, usamos o comando **javac**. No caso: **javac Main.java**.

A execução desse comando pode resultar em duas situações:

1. O código foi compilado com sucesso, ou
2. Existem erros de sintaxe no código fonte

Existindo erros de sintaxe eles serão listados na tela. Conserte os erros e compile o programa novamente.

Quando a execução do comando **javac** for concluída (sem erros) você notará a presença de um novo arquivo junto com o código fonte: **App.class** (código java compilado, pronto para ser executado pela JVM). Para cada arquivo **.java** será criado um arquivo **.class** correspondente.

Executando

Agora que o programa foi compilado com sucesso, podemos executá-lo. Para isso, usaremos o comando **java** seguido do nome da classe inicial do nosso programa. A classe inicial de um programa é aquela que contêm o método *static void main(String[] args)*. Vale ressaltar que, quando executamos o programa, não devemos incluir a extensão do arquivo. Para executar o programa recém-criado digite: **java App.**

Criando um arquivo .jar

Programas java mais elaborados são formados por mais de um arquivo **.class** (basicamente um arquivo **.class** por classe). Arquivos **.jar** são arquivos que empacotam todos os .class e outros recursos necessários a execução de programas mais complexos facilitando a distribuição dos programas. Para criar um arquivo **.jar** usamos o seguinte comando:

```
jar cfe [jarFile] [mainClass] [inputFile(s)]
```

- **jar** : indica que queremos criar um arquivo Java executável

- **cf** : o **c** (create) indica que deve ser criado um executável novo, o **f** indica que o mesmo deve ser armazenado em um arquivo (ao invés de imprimi-lo na tela, por exemplo) e **e** indica que vamos especificar a classe inicial do programa.
- **jarFile** : é o nome que queremos dar ao arquivo criado. Por convenção, executáveis Java tem a extensão **.jar**, mas ela não é obrigatória
- **mainClass** : o nome da classe inicial do programa (não o arquivo, apenas o nome da classe)
- **inputFile(s)** : lista dos arquivos que compõem o executável podendo ser arquivos de código fonte, imagens, etc.

Se quisermos criar um **.jar** da classe *App* devemos usar o comando como segue: **jar cfe App.jar App App.class**.

Para executar um arquivo **.jar** (seu ou de outro desenvolvedor) usamos o comando **java** seguido da *flag* **-jar** seguida do caminho para o arquivo, por exemplo, **java -jar App.jar**.

Exercício:

1) Crie as duas classes abaixo e, em seguida, crie um ".jar" que empacote as duas. Execute o programa a partir do ".jar".

```
public class Contas{
    public int soma(int a,int b){
        return a+b;
    }
}
```

```
public class App{
    public static void main(String args[]){
        Contas contas = new Contas();
        System.out.println("Teste: "+contas.soma(5,8));
    }
}
```

2) Pesquise como se faz para programas Java receberem parâmetros pela linha de comando. Escreva um exemplo.

3) Pesquise como se faz redirecionamento de entrada e saída no Linux.

    a. Use este recurso para fazer um programa que cria um arquivo com os inteiros de 0 a 10000.
    b. Use este recurso para fazer um programa que recebe um arquivo de números e gera outro com os números somados de um valor recebido como parâmetro na linha de comando.

Outros recursos de compilação

Por vezes precisamos organizar os arquivos em pastas. Então é necessário dizer ao "javac" onde estão os arquivos fonte e onde ele deve colocar os ".class".

Exemplo: javac ./src/*.java -d ./target

Neste caso estamos dizendo que os arquivos fonte estão na pasta “src”. Da mesma forma estamos dizendo que os arquivos “.class” devem ser criados dentro da pasta “target”.

Também é possível usar a opção “-cp” para dizer onde mais encontrar arquivos fonte.

java –cp target App

Criando arquivo com lotes de comandos

Para criar um arquivo executável com um conjunto (lote) de comandos de shell é necessário criar um arquivo texto iniciado por:

#!/bin/bash

Na sequência coloca-se um comando de shell por linha.

Depois é só alterar as permissões do arquivo para “x”. Para tanto use “chmod +x <nome do arquivo>". Para mais detalhes consulte https://www.pluralsight.com/blog/it-ops/linux-file-permissions