EXERCÍCIO 1:

Criar em Eclipse, um novo Java Project com uma classe chamada RedesController.java no package controller e uma classe Main.java no package view.

A classe RedesController.java deve ter 3 métodos.

1) O primeiro, chamado os, que identifica e retorna o nome do Sistema Operacional (Fazê-lo privado)

2) O segundo, chamado ip, que verifica o Sistema Operacional e, de acordo com o S.O., faz a chamada de configuração de IP.

A leitura do processo chamado deve verificar cada linha e, imprimir, apenas, o nome do adaptador de rede e o IPv4, portanto, adaptadores sem IPv4 não devem ser mostrados

3) O terceiro, chamado ping, que verifica o Sistema Operacional e, de acordo com o S.O. e, de acordo com o S.O., faz a chamada de ping em IPv4 com 10 iterações.

A leitura do processo chamado deve verificar as linhas de saída e exibir, apenas, o tempo médio do ping. O teste de ping deve ser feito com a URL www.google.com.br

A Classe Main.java deve dar as opções de chamadas do método ip ou do método ping com JOptionPane e, dependendo da escolha, instanciar a Classe RedesController.java e chamar o método escolhido. A opção de finalizar a aplicação também deve estar disponível.

Dicas:

1) Para validar o Sistema Operacional, utilizar a operação contains;

2) Para validar as saídas e executar o que foi pedido, utiizar a operação Split;

3) Processo de chamada de configuração de IP:

Windows: IPCONFIG

Linux: IFCONFIG

4) Processo de chamada de PING com 10 iterações, em IPv4 para www.google.com.br

Windows: PING -4 -n 10 www.google.com.br

Linux: PING -4 -c 10 www.google.com.br

EXERCÍCIO 2

Fazer, em java, uma aplicação que liste os processos ativos, permita ao usuário entrar com o nome ou o PID do processo e o mate.

A aplicação deverá funcionar, minimamente em Windows e Linux (Alunos com Mac podem fazer para os 3 SO).

É notório que cada SO tem comandos diferentes para as ações supracitadas, portanto:

Criar em Eclipse, um novo Java Project com uma classe chamada KillController.java no package controller e uma classe Main.java no package view.

A classe KillController.java deve ter 4 métodos.

1) O primeiro, chamado os, que identifica e retorna o nome do Sistema Operacional (Fazê-lo privado)

2) O segundo, chamado listaProcessos, que verifica o SO e, de acordo com SO, selecione o comando para listar os processos ativos.

O método deve receber todas as linhas de saída do processo de listagem e exibi-las em console

3) O terceiro, chamado mataPid, que recebe um PID como parâmetro de entrada, verifica o SO e, de acordo com SO, selecione o comando para matar o processo e o finalize

4) O quarto, chamado mataNome, que recebe um nome de processo como parâmetro de entrada, verifica o SO e, de acordo com SO, selecione o comando para matar o processo e o finalize

Dicas:

1) Chamada de processo para listagem da tabela de processos:
Windows: TASKLIST /FO TABLE
Linux: ps -ef

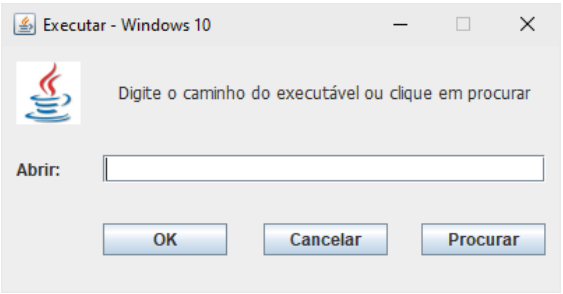
2) Chamada de processo que mata processo por PID:
Windows: TASKKILL /PID pid_do_processo
Linux: kill -9 pid_do_processo

3) Chamada de processo que mata processo por Nome:
Windows: TASKKILL /IM pid_do_processo
Linux: pkill -f nome_do_processo

EXERCÍCIO DESAFIO (Para S.O. Windows):

Utilizando o Framework Window Builder, criar, em Eclipse, um projeto Java que simula o Executar (Run) do Windows.
No package view, deve ser criado, com auxílio do framework, conforme figura abaixo, uma tela com um JTextField e 3 botões (OK, Cancelar e Procurar).

No package controller, devemos ter :
1) Uma classe, chamada SearchController, que receba o JTextField pelo construtor, implementa um ActionListener para executar a ação do botão Procurar. No método ActionPerformed, deve ter uma busca de arquivos executáveis Windows, via JFileChooser, e seleciona o arquivo a ser executado e escreve seu caminho completo no JTextField.
2) Uma classe, chamada RunController, que receba o JTextField e o próprio JFrame da tela pelo construtor, implementa um ActionListener para executar a ação do
botão OK. No método ActionPerformed, deve tentar executar o que está escrito no JTextField (O usuário pode digitar o caminho por conta própria, ao invés de procurar). Caso o arquivo seja inválido, dar uma mensagem de erro. Uma vez executado, sem erro, a tela deverá ser finalizada pelo método dispose().
3) Uma classe, chamada CancelController, que receba o próprio JFrame da tela pelo construtor, implementa um ActionListener para executar a ação do botão Cancelar. O método actionPerformed deve proceder um dispose() da tela.



Dicas:
Assistir, no site do Professor, os vídeos:
1) Eclipse Window Builder Aplicação com ActionListener implementado
2) Introdução ao JFileChooser