Ex. 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 2 métodos.
O primeiro, chamado ip, que recebe o nome do Sistema Operacional como parâmetro e, de acordo com o S.O., faz a chamada de configuração de IP e filtra a saída do processo, retornando um String com o nome do Adaptador Ethernet e o IPv4 apenas (Não importa o número de adaptadores ethernet, devem aparecer todos). Os adaptadores que não tiverem IPv4 não devem ser mostrados.
O segundo, chamado ping, que recebe o nome do Sistema Operacional como parâmetro e, de acordo com o S.O., faz a chamada de ping com 10 iterações, filtra a saída, pegando apenas o tempo e dá a saída, em ms, do tempo médio do ping. (O endereço para ping, pode ser o www.google.com.br)
A Classe Main.java deve ter a possibilidade de o usuário escolher a ação que quer fazer e, dependendo da escolha, instanciar a Classe RedesController.java e chamar o método escolhido. A opção de finalizar a aplicação também deve estar disponível.

* Para filtrar a saída String, considere utilizar contains, Split, substring, lastIndexOf, trim;

Ex. 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. Pesquisar os comandos para cada SO.

A aplicação deverá ter, no package view, uma classe que tenha um método main que dê ao usuário a possibilidade de ver os processos ativos ou matar os processos (Por Nome ou PID).

No package controller, deverá ter:

1) Uma classe que tenha um método que identifique o SO;
2) Um método que, recebendo o SO, no qual está rodando, como parâmetro de entrada, selecione o comando para listar os processos ativos;
3) Um método que, recebendo o SO, no qual está rodando, e o PID do processo, como parâmetros de entrada, selecione o comando para matar o processo e o finalize;
4) Um método que, recebendo o SO, no qual está rodando, e o Nome do processo, como parâmetros de entrada, selecione o comando para matar o processo e o finalize;

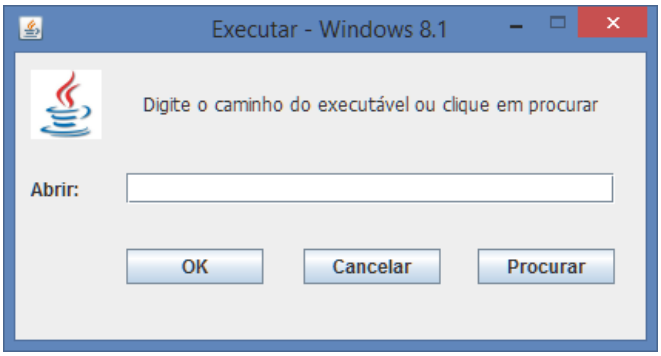
Ex. 3

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 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 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 finalizado pelo método dispose().
3) Uma classe 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.



Assistir, no site do Professor, os vídeos:

1) Eclipse Window Builder Aplicação com ActionListener implementado

2) Introdução ao JFileChooser