4) Quais as classes essenciais envolvidas em uma aplicação JavaFX? Qual é o papel de cada uma? Há padrões de projeto envolvidos? Quais?

No JavaFX, para que uma aplicação de interface gráfica seja funcional, é necessário ter as classes que inicializam isso, aqui estão as principais classes de JavaFX:

1. Application Class: É uma classe abstrata que tem o método start(), ela que inicia a aplicação e configura a janela principal do JavaFX.

2. Stage: É a janela principal que contém as "cenas", ou seja, ela exibe a interface do usuário.

3. Scene: Este, é dentro da janela, seria o conteúdo exibido, no caso os elementos, a exemplo de: controles, gráficos, animações e entre outros.

4. FXML: É extremamente importante, é uma linguagem de marcação, isso vai definir a estrutura da interface, é como se fosse um HTML sobre desenvolvimento web, quem carrega ele é o FXMLLoader,

e cria com base nele.

5. Controller: É ele quem vai manipular os eventos e responsabilizar pela lógica da interface, isso tem correlação com os elementos do FXML.

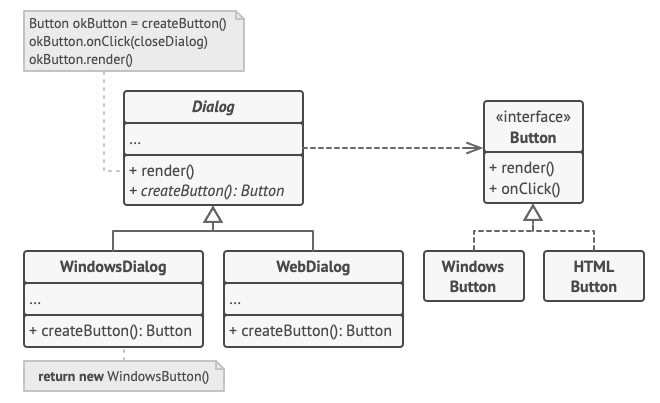
6. CSS: Normalmente, quando cria-se um programa em JavaFX, automaticamente um arquivo chamado application.css é criado, ele é responsável por estilizar a aparência da interface para o usuário,

define-se cores, estilos e formatação...

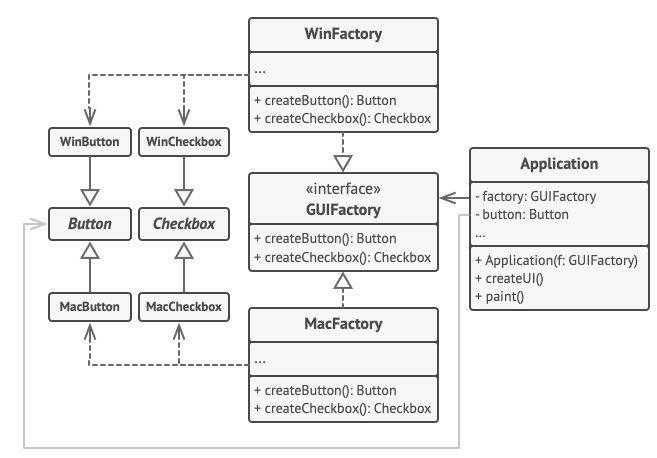
7. Model: São os dados subjacentes da aplicação, no caso, é como se fosse uma lógica que uma aplicação manipula e exibe na interface do usuário.

Em relação aos padrões de projetos, sim, há padrões envolvidos, não necessariamente todos, mas alguns que se aplicam muito bem com interface gráfica, veja:

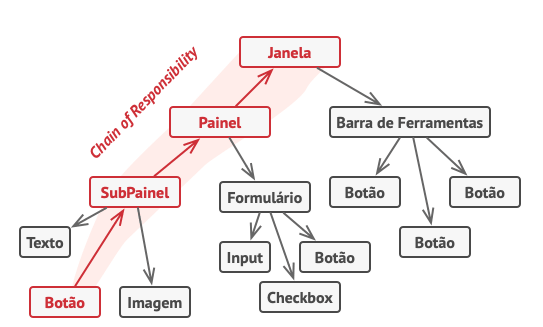
Factory Method: Pode ser comumente utilizado para criar elementos de interface do usuário sem acoplar o código do cliente às classes de UI concretas.



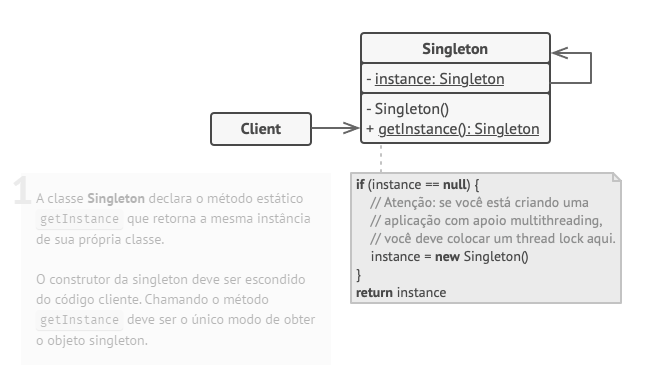
Abstract Factory: pode ser usado para criar elementos UI multiplataforma sem ter que ligar o código do cliente às classes UI concretas, enquanto mantém todos os elementos criados consistentes com um sistema operacional escolhido.



Chain of Responsibility: é responsável por mostrar informação de ajuda contextual para elementos de GUI ativos.



Singleton: Ele garante que haja uma única instância de objetos importantes, como o Stage (janela principal).



Esses são apenas alguns que podem ter mais relevância em comparação aos outros padrões de projetos, já que, no site refactoring guru, há alguns exemplos que se relacionam diretamente com a interface gráfica, mas

podemos também tentar alguma implementação com outros padrões, dependendo do tipo e do que se trata.

-------------------------------

5) Para os “controles” selecionados: quais as classes envolvidas em cada controle? Quais são seus atributos e métodos mais relevantes? O que o controle faz e quando utilizá-lo?

Para Label, Button e Checkbox são as seguintes classes:

Label: javafx.scene.control.Label

Button: javafx.scene.control.Button

Checkbox: javafx.scene.control.CheckBox

Atributos e métodos mais relevantes separados por "/":

Label: textProperty() (definir o texto exibido no rótulo), setFont(Font font) (o tipo de fonte a ser exibido) / setText(String texto): Define uma área de texto para UI, setGraphic(Node graphic): Define o nó gráfico associado ao rótulo.

Finalidade: O Label é utilizado para exibir texto ou informações para a UI, São rótulos para outros elementos, campo de entrada, botões...

Button: textProperty() (O texto exibido no botão, pode definir ou obter o texto por meio desse atributo), onActionProperty() (Cria um evento de ação a ser executado quando o botão é clicado) / setText(String texto) (Definir texto para o botão), setOnAction(EventHandler<ActionEvent> handler) (Cria um evento de ação (Action) para ser executado ao clicá-lo).

Finalidade: Cria botões na UI para associar a um evento para executá-lo, são bons para transitar telas, formulários, comandos...

CheckBox: textProperty() (Texto ao lado da caixa de seleção que é exibida), selectedProperty() (Estado da caixa de seleção) / setText(String texto) (Definir texto ao lado da caixa de seleção), isSelected() (Verifica se a caixa está selecionada ou não).

Finalidade: O usuário pode selecionar várias caixas de seleção ou somente uma ou outra, é recomendável para várias, já que a radiobutton já faz o trabalho binário. Utilizado bastante para opiniões, ativar e desativar alguma configuração.

6) Para os “contêineres” e “painéis” selecionados: quais são as classes envolvidas e quais são suas aplicações? Quais são seus atributos e métodos mais relevantes? Há padrões de projeto envolvidos? Quais?

Para ScrollBar e Menu são as seguintes classes:

ScrollBar: javafx.scene.control.ScrollBar

Menu: javafx.scene.control.Menu

Atributos e métodos mais relevantes:

ScrollBar:

Atributos:

valueProperty(): Valor atual da barra de rolagem.

minProperty(): Valor mínimo da barra de rolagem.

maxProperty(): Valor máximo da barra de rolagem.

Métodos:

setValue(double value): Define o valor atual da barra de rolagem.

setMin(double min): Define o valor mínimo da barra de rolagem.

setMax(double max): Define o valor máximo da barra de rolagem.

Finalidade: Muitas vezes, o conteúdo que é exibido, não mostra tudo pelo seu tamanho, então, temos a scrollbar que é uma barra de rolagem para averiguar o conteúdo restante que não foi exibido anteriormente.

Menu:

Atributos:

textProperty(): Representa o texto exibido no menu.

items: Representa a lista de itens de menu contidos dentro do menu.

Métodos:

setText(String text): Define o texto exibido no menu.

getItems(): Obtém a lista de itens de menu contidos no menu.

addItem(MenuItem item): Adiciona um item de menu ao menu.

Finalidade: Menus são criados para fornecer opções de ação ao usuário, ativar, navegar em diferentes lugares, um exemplo do código foi fornecido, mostrando que ao clicar no menu JavaFX, ele mostra as opções e, após alguma selecionada, emite um alert.

Referências bibliográficas:

<https://refactoring.guru/pt-br/design-patterns>

<https://docs.oracle.com/javase/8/javafx/api/javafx/scene/control/Menu.html>

<https://docs.oracle.com/javase/8/javafx/api/javafx/scene/control/MenuItem.html>

<https://docs.oracle.com/javase/8/javafx/api/javafx/scene/control/Label.html?external_link=true>

<https://docs.oracle.com/javase/8/javafx/api/javafx/scene/control/Button.html>

<https://docs.oracle.com/javase/8/javafx/api/javafx/scene/control/CheckBox.html>

<https://docs.oracle.com/javase/8/javafx/api/javafx/scene/control/ScrollBar.html>

<https://stackoverflow.com/questions/28037818/how-to-add-a-scrollbar-in-javafx>