Interface Pessoa-Máquina

Licenciatura em Engenharia Informática

Ficha Prática #07

José Creissac Campos
jose.campos@di.uminho.pt

(v. 1.0)

Conteúdo

1 Objectivos

1. Praticar construção de interfaces com JavaFX.

2 JavaFX

2.1 Tratamento de eventos

O tratamento de eventos é um aspecto importante da programação de interfaces gráficas, permitindo que as aplicações respondam às acções do utilizador.
Em JavaFX, um evento é gerado em resposta a uma dada acção ou a uma alteração de estado. Exemplos de eventos incluem cliques no rato, premir teclas, activação de botões e redimensionamento de janelas. Os eventos são normalmente processados por *event handlers*, que são responsáveis por definir as acções a serem executadas quando ocorre o evento.
As **fontes de eventos** (*event sources*) são os objectos que geram eventos. Em JavaFX, as fontes de eventos comuns incluem controlos de IU, como botões, campos de texto e menus. Cada fonte de eventos tem tipos de eventos associados que definem os eventos específicos que pode gerar. Por exemplo, um botão pode gerar um `ActionEvent` quando clicado.
Os ***event handlers*** são responsáveis pelo tratamento dos eventos gerados pelas fontes de eventos e definem as ações a serem executadas quando os eventos a que estão associados ocorrem. Em JavaFX, um *event handler* para um qualquer tipo de evento `T`, é uma implementação da interface `EventHandler<T>`.
Para associar um *event handler* a uma fonte de eventos, é necessário registrar o *event handler* na fonte de eventos apropriada. Isto pode ser feito de forma declarativa em FXML, ou de forma imperativa utilizando o método `addEventHandler()` da fonte de eventos. No código fornecido com este tutorial é dada preferência à abordagem declarativa.

2.2 Propriedades

As propriedades encapsulam um valor e expõem métodos para aceder e modificar esse valor[1]. Oferecem funcionalidades adicionais, como a notificação de alterações, permitindo ao programa reagir a alterações nos valores das propriedades. Por exemplo, a classe `SimpleStringProperty`[2] representa uma propriedade que con-

[1] `https://docs.oracle.com/javafx/2/api/javafx/beans/property/Property.html`
[2] `https://docs.oracle.com/javafx/2/api/javafx/beans/property/SimpleStringProperty.html`

tém uma string e a classe `SimpleIntegerProperty`[3] representa uma propriedade que contém um inteiro.
Estabelecer *bindings* (ligações) entre propriedades permite que estas propriedades sejam *ligadas* entre si, fazendo com que alterações numa propriedade se propaguem automaticamente para outras propriedades dependentes. Isto elimina a necessidade de sincronização manual e garante que a interface se mantém actualizada em relação aos dados. As ligações podem ser estabelecidas utilizando métodos como `bind` e `bindBidirectional`.
Por exemplo, considere duas instâncias de `SimpleIntegerProperty`: `prop1` e `prop2`. É possível criar uma ligação entre elas usando o método `bind`: `prop1.bind(prop2)`. Agora, sempre que o valor de `prop2` for alterado, a propriedade `prop1` será automaticamente actualizada para reflectir o novo valor.
O JavaFX também suporta ligações bidireccionais, permitindo que as alterações se propaguem em ambas as direcções. Com uma ligação bidireccional, sempre que a propriedade `prop1` ou a propriedade `prop2` sejam alteradas, a outra será actualizada em conformidade. A ligação bidireccional pode ser obtida utilizando o método `bindBidirectional`: `prop1.bindBidirectional(prop2)`.
Em geral, a utilização de propriedades e *bindings* em JavaFX simplifica a gestão do estado da interface e assegura uma interface de utilizador consistente e reactiva, ao mesmo tempo que reduz a quantidade de código necessária no controlador.

3 Exercícios

1. Considere o projecto fornecido como exemplo. O tratamento dos graus Celsius e do título estão completos, mas falta terminar o tratamento dos graus Fahrenheit e Kelvin. Complete o código de modo a que toda a interface fique funcional. Para isso deverá: adicionar as propriedades necessárias no *Model* e os *bindings* e *event handlers* no *Controller*, bem como actualizar a *View* para usar esses *handlers*.

2. ★Trabalhando a partir dos os écrans do seu protótipo anteriormente implementados, estabeleça a ligação com a lógica de negócio desenvolvendo quer o *Model* e as suas propriedades, quer os *event handlers* e *bindings* do *Controller* e *View*.

[3] `https://docs.oracle.com/javafx/2/api/javafx/beans/property/SimpleIntegerProperty.html`