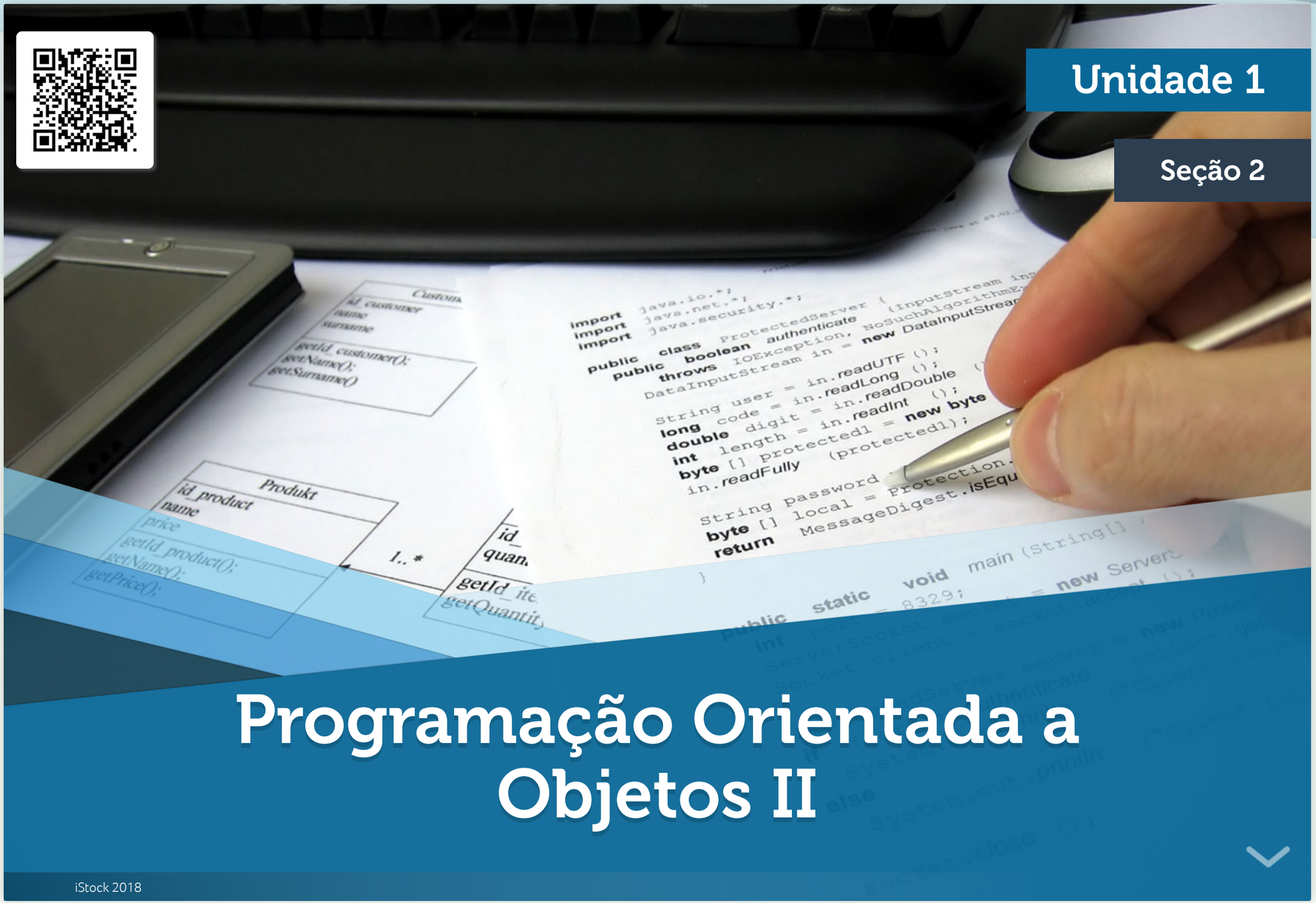
Unidade 1

Seção 2

Programação Orientada a Objetos II



iStock 2018

Webaula 2

Programação em Java usando orientação a eventos

Nesta webaula vamos estudar a programação Java utilizando orientação e eventos, e veremos sobre as interfaces e afins.

iStock 2018

As interfaces gráficas representam como o usuário percebe o sistema, e para se criar essa interface é necessário seguir alguns passos. Clique nas abas e veja quais são:

1° Passo | 2° Passo | 3° Passo | 4° Passo | 5° Passo

Interface gráfica x Interface

Seguindo esses passos é possível criar e posicionar os componentes, todavia, para que essa interface seja interativa é necessário tratar os eventos decorrentes das ações do usuário na janela. No Java Swing existem diversos eventos que podem ser tratados através do uso de interfaces da orientação a objetos.

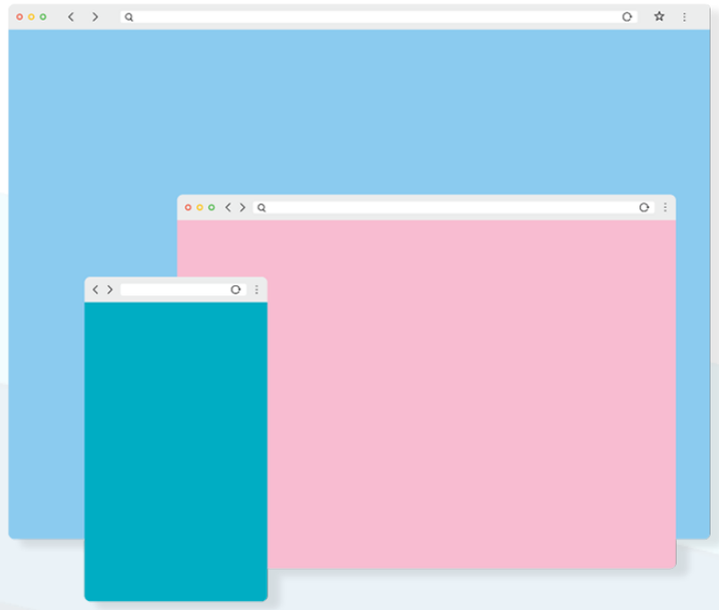
É importante perceber que esse novo elemento possui conceito e implementação distintos de uma interface gráfica.

Para criar uma interface clique com o botão direito no projeto, selecione *New >> Interface*, escolha a pasta e o nome. O código no quadro a seguir apresenta um exemplo desse item da orientação a objetos. Repare que nesse tipo de código não temos nenhuma implementação de código e sim um conjunto de métodos. Essas descrições de implementação representam uma forma de padronizar o método que será o responsável por tratar um evento na interface gráfica.

```
1. public interface AcessoElementos {
2.         public int getElemento (int index);
3.         public void setElemento (int index);
4. }
```

Fonte: elaborado pelo autor

Para evitar confusões na definição dos termos, chamaremos de **interface** os elementos da orientação a objetos que descrevem um conjunto de métodos, e **interface gráfica** serão os elementos como botões, janelas, campos e outros. Ao se construir uma interface gráfica usando Java *Swing*, acrescentamos as interfaces para definir qual método será utilizado no momento de chamada que um evento é iniciado.



Tratamento de eventos

A criação da interface gráfica representa o mecanismo de interação do usuário com o sistema. Cada ação do usuário gerará uma "reação" no e do sistema. Esse conjunto de ações (interações) é o que se chama de **evento**.

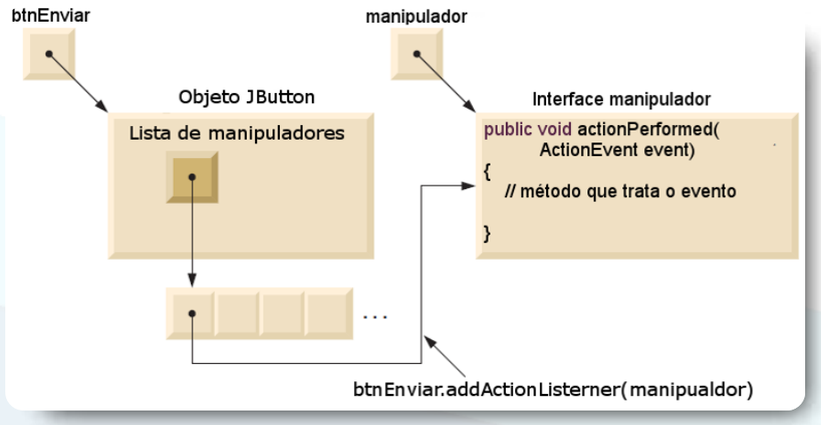
O quadro a seguir traz alguns eventos suportados pela linguagem Java.

Evento	Descrição
ActionListener	Evento gerado pelo clique com o mouse em um botão.
ItemListener	Evento gerado quando um item em uma lista é selecionado.
FocusListener	Evento gerado quando um elemento ganha ou perde foco.
WindowListener	Evento gerado quando ocorre uma mudança na janela.

Fonte: elaborado pelo autor

Para tratar a ação de clique em um botão em uma janela gráfica feita em Java *Swing*, utiliza-se a interface *ActionListener*. Na figura a seguir temos a representação de um objeto da classe *JButton*, cuja instância pode tratar diversos eventos. A instância "btnEnviar" registra uma interface do manipulador, nesse caso o *ActionListener*, para tratar um de seus possível eventos.

Clique na imagem para melhor visualizá-la.



Fonte: adaptada de Deitel e Deitel (2016, p. 494).

Novamente, repare que não há implementação nesses métodos, apenas a descrição (assinatura).

Para implementar o tratamento de eventos é necessário que a classe que está especializando o *JFrame* tenha alguma implementação dos métodos da interface que trate os eventos e associe essa interface aos componentes que enviarão os eventos (HORSTMANN, 2016).

Para demonstrar, utilizaremos a tela da figura. Essa interface gráfica possui um *JButton*, um *JLabel* e um *JTextField*. O objetivo é verificar se o botão foi clicado pelo usuário, e para isso é possível utilizar a interface *ActionListener* de três maneiras distintas (MANZANO, 2014):

1. Implementar a interface *ActionListener* diretamente na classe que especializa o *JFrame*.

2. Criar uma classe que implementa a interface e trata todos os eventos dos componentes gráficos.

3. Fazer com que cada componente crie uma classe anônima que implemente a interface *ActionListener*.

Existem muitos outros eventos para serem apresentados; se um componente recebeu ou perdeu o foco, por exemplo. A forma de tratá-los segue o mesmo conceito visto nos outros casos. O quadro a seguir apresenta um código para verificar se um componente gráfico recebeu ou perdeu o foco. Na linha 1 cria-se uma classe anônima para programar a interface *FocusListener*, e essa criação força a criação dos métodos *focusLost*() e *focusGained*().

```
1.  txtNome.addFocusListener(new FocusListener() {
2.      public void focusLost(FocusEvent e) {
        }
3.      public void focusGained(FocusEvent e) {
        }
    });
```

Fonte: elaborado pelo autor.

O tratamento de eventos em Java *Swing* deve sempre seguir os elementos da orientação a objetos, e esses métodos devem ter alta coesão, sendo que cada uma dessas ações deve se limitar a funções específicas.

Nesse sentido, a implementação das interfaces para tratar os eventos deve apenas recebê-los e chamar outros métodos para que cada ação seja tratada de forma independente.

Você já conhece o Saber?

Aqui você tem na palma da sua mão a **biblioteca digital** para sua **formação profissional**.

Estude no celular, tablet ou PC em qualquer hora e lugar sem pagar mais nada por isso.

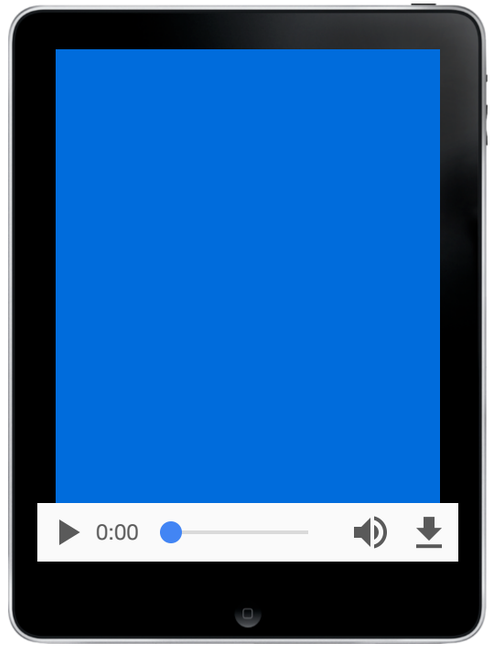
Mais de 450 livros com interatividade, vídeos, animações e jogos para você.

DISPONÍVEL NA Google play

Android:
https://goo.gl/yAL2Mv

Baixar na App Store

iPhone e iPad - IOS:
https://goo.gl/OFWqcq



Bons estudos!