



# CCM310

# Arquitetura de Software e

# Programação Orientada a Objetos

Profa. Dra. Gabriela Biondi

Prof. Dr. Isaac Jesus

Prof. Dr. Luciano Rossi

GUI  
*Graphical User Interface*

# Eventos

# Eventos

- Um evento ocorre quando o **usuário interage com um objeto gráfico**:
  - Manipular um botão com o mouse
  - Introduzir texto num campo de texto
  - Selecionar um item de menu
- Na programação por eventos, programam-se objetos (*event listeners*) que reagem a alterações no estado de outros objetos (*event sources*)

# Eventos

- Um *event listener* inclui um método que será executado em resposta ao evento gerado:
  - Pressionar do Botão: método *ActionPerformed( ActionEvent evt )*
- Quando um evento é gerado o sistema de execução notifica o objeto de escuta (*listener*) correspondente, invocando o método que trata o evento
- Caso não exista um *listener* registrado no objeto que gera o evento, este não terá qualquer efeito

# Eventos

Exemplos:

- Quando ocorre uma ação ( *ActionEvent* )
- Apertar em uma tecla ( *KeyEvent* )
- Clique do mouse ( *MouseEvent* )
- Fechar uma janela ( *WindowEvent* )

# Eventos com o Teclado

# *KeyEvent*

*event(KeyEvent:*

- Um evento que indica que ocorreu um pressionamento de tecla em um componente.
- Este evento é gerado por um *objeto de componente* quando uma tecla é pressionada, liberada ou digitada



**Como um campo de texto,  
por exemplo**

# KeyEvent

*event(KeyEvent:*

- Pressionar e soltar uma tecla no teclado resulta na geração dos seguintes eventos de teclas (em ordem):
  - **KEY\_PRESSED**
  - **KEY\_TYPED** (*somente se um Unicode válido puder ser gerado*)
  - **KEY\_RELEASED**

# KeyEvent

*event(KeyEvent:*

- Os eventos **KEY\_PRESSED** e **KEY\_RELEASED** são gerados sempre que uma tecla é pressionada e liberada, respectivamente
- A tecla pressionada ou liberada é indicada pelos métodos **getKeyCode** e **getExtendedKeyCode**, que retornam um código de *keyCode*

# KeyEvent

*event(KeyEvent:*

- **Virtual key codes** são usados para relatar qual tecla do teclado foi pressionada, em vez de um caractere gerado pela combinação de uma ou mais teclas (como *A*, que vem de *shift+a*)
- Por exemplo, pressionar a tecla *shift* causará um evento **KEY\_PRESSED** com um *keyCode* **VK\_SHIFT**
- Enquanto pressionar a tecla *a* resultará em um *keyCode* **VK\_A**

# KeyEvent

*event(KeyEvent:*

- Depois que a tecla *a* for liberada, um evento **KEY\_RELEASED** será disparado com *VK\_A*.
- Separadamente, um evento **KEY\_TYPED** com um valor *keyChar* de '*A*' é gerado.

# KeyEvent

*event(KeyEvent:*

- [KeyEvent \(Java SE 11 & JDK 11 \)](#)

Exemplo em Aula

# Eventos com o Mouse

# *MouseEvent*

- Um evento que indica que uma ação do mouse ocorreu em um componente
- Uma ação do mouse é considerada como ocorrendo em um componente específico se o cursor do mouse estiver sobre a parte não obscurecida dos limites do componente quando a ação acontecer

# *MouseEvent*

- *mouseClicked()*
- *mouseEntered()*
- *mouseExit()*
- *mousePressed()*
- *mouseReleased()*
- *mouseDragged()*
- *mouseMoved()*

# *MouseEvent*

- Para capturar as coordenadas do ponteiro do mouse:
  - `evt.getX()`
  - `evt.getY()`

# MouseEvent

- Documentação completa:
  - [MouseEvent \(Java SE 11 & JDK 11\)](#)

Exemplo em Aula

classe Graphics

# Graphics

- A classe **Graphics** (*java.awt.Graphics*) é a classe base para todos os contextos gráficos que permitem que um aplicativo desenhe
- Alguns métodos:
  - *drawLine(int x1, int y1, int x2, int y2)*
  - *drawRect(int x, int y, int width, int height)*
  - *fillOval(int x, int y, int width, int height)*
  - *fillRect(int x, int y, int width, int height)*
  - *setColor(Color c)*

# Graphics

- A classe **Graphics** (*java.awt.Graphics*) é a classe base para todos os contextos gráficos que permitem que um aplicativo desenhe
- Documentação completa:
  - [Graphics \(Java SE 11 & JDK 11\)](#)

# Graphics

- Exemplo 1:
  - Graphics + MouseEvent
  - Desenhando na tela com o mouse

# Graphics

- Exemplo 2:
  - Graphics + MouseEvent
  - Desenhando na tela com o mouse
  - Escolhendo as cores com o componente: *Color Chooser*

# Graphics - Exercício 1

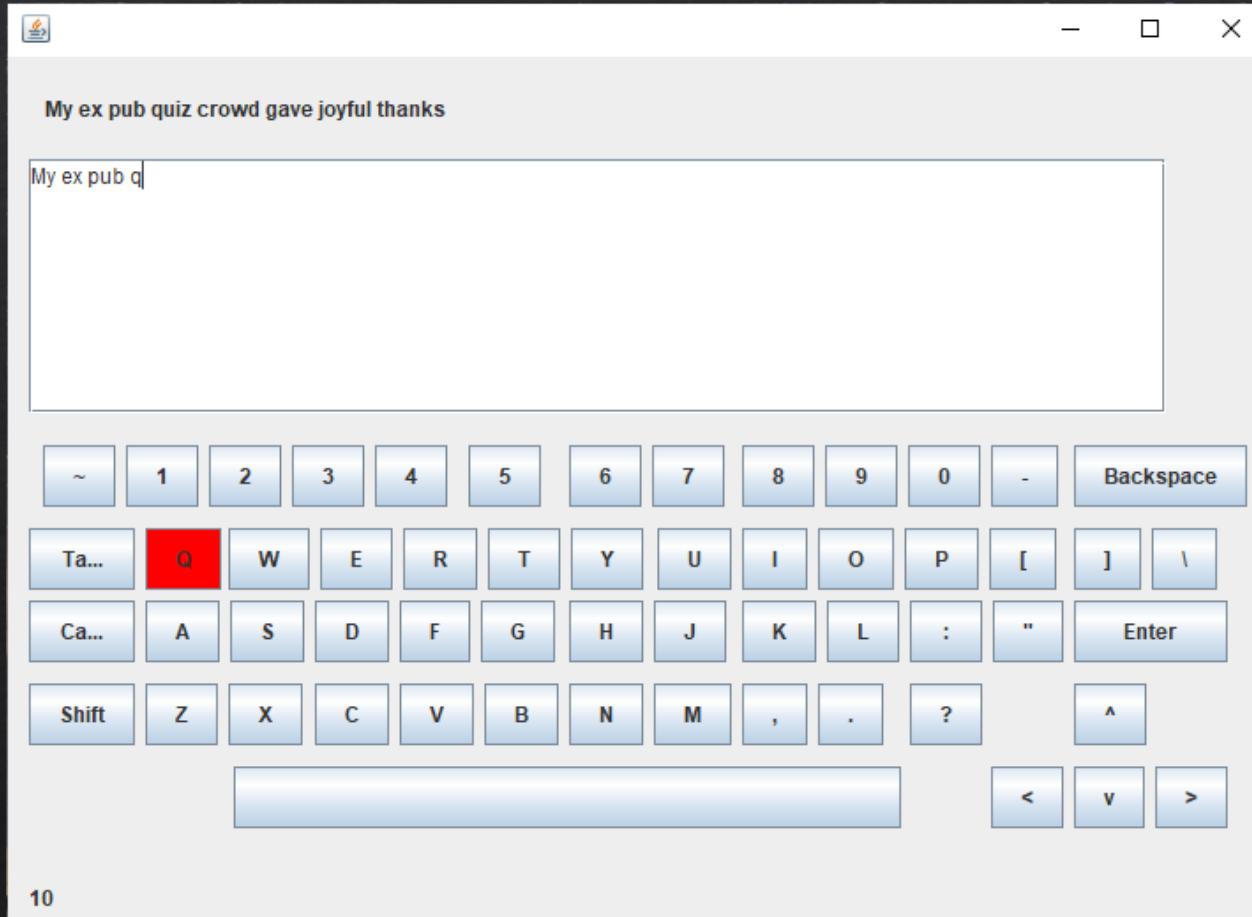
- Construa um aplicativo completo de desenho!
- Inclua as opções:
  - Desenhar retângulo, círculo, linha
  - Alterar tamanho do pincel (com *JSlider*)
  - Incluir Texto
  - Apagar o desenho feito



# Tutor de Datilografia - Exercício 2

*Vamos começar na aula de teoria e deverá ser concluído  
e entregue no laboratório*

# Exercício 2 - Tutor para Datilografia



# Exercício 2 - Tutor para Datilografia

- (*Deitel 14.20*) Digitar de forma rápida e correta é uma habilidade essencial para trabalhar efetivamente com computadores e a Internet. Neste exercício, você vai construir uma aplicação gráfica que ajuda os usuários a aprenderem a “**datilografar**” (ou seja, digitar corretamente sem olhar para o teclado).
- O aplicativo deve exibir um teclado virtual e deve permitir ao usuário que ele veja o que está digitando na tela sem olhar para o teclado real. Use **JButtons** para representar as teclas. À medida que o usuário pressiona cada tecla, o aplicativo destaca o  **JButton** correspondente na GUI e adiciona o caractere a uma **JTextArea** que mostra o que o usuário digitou até o momento.

# Exercício 2 - Tutor para Datilografia

- Para destacar um *JButton*, use seu método *setBackground* para alterar a cor de seu plano de fundo; Quando a tecla for liberada, redefina sua cor de fundo original. Você pode obter as cores originais dos botões *JButton* com o método *getBackground* antes de alterar sua cor.
- Você pode testar seu programa digitando um *pangrama* - uma frase que contém todas as letras do alfabeto pelo menos uma vez - como "**the quick brown fox jumped over a lazy dog**".
- Para tornar o programa mais interessante, você deve monitorar a precisão do usuário. Você deve usar os pangramas exibidos na tela acima do teclado virtual. Então, você acompanha (conta) quantas teclas o usuário digita corretamente e quantas são digitadas incorretamente.

# Exercício 2 - Tutor para Datilografia

- Utilize vários pangramas em um ArrayList e sorteie um para ser exibido na janela
- Para monitorar o desempenho do usuário, conte quantas vezes ele pressiona o backspace. Caso ele conclua a frase de forma incorreta, o programa deve exibir uma caixa de diálogo, *JOptionPane*, informando que a frase precisa ser corrigida.
- A caixa de diálogo deve ser usada também para exibir o desempenho do usuário (erros)

Obrigada pela sua  
participação, nos vemos na  
próxima aula! :)