CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA
DO ESTADO DE MINAS GERAIS
CAMPUS DIVINÓPOLIS

ENGENHARIA DE COMPUTAÇÃO

Disciplina: Linguagens de Programação

ESTUDO DIRIGIDO: PROGRAMAÇÃO ORIENTADA À EVENTOS

Conceitos e Implementação de uma Calculadora

Aluno:
Júlia de Moura Souza

06 de dezembro, 2025

Júlia de Moura Souza

ESTUDO DIRIGIDO: PROGRAMAÇÃO ORIENTADA À EVENTOS

Conceitos e Implementação de uma Calculadora

06 de dezembro, 2025

Resumo

Este documento apresenta o estudo aprofundado dos conceitos fundamentais da Programação Orientada a Eventos. A pesquisa combina fundamentos teóricos com exemplos práticos na linguagem Python utilizando a biblioteca Tkinter para construção de interfaces gráficas. Como atividade prática verificadora, implementa-se uma calculadora interativa que demonstra na prática a aplicação desses conceitos, ilustrando o funcionamento de sistemas reativos baseados em eventos.

1 Introdução

A Programação Orientada a Eventos (POE) é um paradigma no qual o fluxo de execução do programa depende da ocorrência de eventos, como ações do usuário, mudanças no sistema ou sinais externos. Em vez de seguir um fluxo rígido e linear de instruções, aplicações orientadas a eventos permanecem em um estado de espera e reagem dinamicamente às interações ou mudanças que ocorrem.

Esse paradigma é amplamente utilizado em aplicações interativas, interfaces gráficas (GUI), sistemas distribuídos, jogos, sistemas embarcados e aplicações que dependem de resposta imediata a inputs externos. O estudo a seguir aborda os três pilares fundamentais da POE — fontes de eventos, ouvintes e manipuladores — aplicados tanto na teoria quanto na prática. A linguagem escolhida para exemplificação e implementação é Python com Tkinter, pela facilidade em manipular eventos e criar interfaces gráficas simples e eficientes.

Ao final do estudo, aplica-se o paradigma construindo uma calculadora básica que demonstra, na prática, o uso integrado dos componentes da POE.

2 Fundamentação

2.1 Produtores (Fontes) de Eventos

O que é um evento? Um evento é uma ocorrência ou mudança significativa no estado de um sistema que pode ser detectada e tratada durante a execução do programa, como interações do usuário, mudanças no sistema ou comunicações externas.

Exemplos de eventos:

- Clique do mouse em um botão (interação do usuário)
- Pressionamento de tecla no teclado (interação do usuário)
- Temporizador expirado (sistema operacional)
- Recebimento de dados via rede (comunicação)
- Redimensionamento da janela (sistema)
- Mudança no valor de um widget (interface gráfica)

O que é uma "fonte de eventos"? Uma fonte de eventos é um componente que pode gerar eventos. Sempre que ocorre uma ação relevante, a fonte é responsável por emitir o evento que poderá ser capturado por *listeners*. Em sistemas gráficos, widgets como botões, campos de texto e janelas são comuns produtores de eventos.

Diferença entre programa sequencial e orientado a eventos:

- **Programa sequencial tradicional:** Segue um fluxo linear e predefinido de execução, onde o controle do programa está centralizado e as ações ocorrem em uma ordem específica determinada pelo código.
- **Programa orientado a eventos:** Baseia-se em um modelo de "espera e reação". O programa fica em um loop principal aguardando eventos ocorrerem e, quando um evento acontece, executa as funções apropriadas para lidar com ele. O controle é descentralizado e dirigido pelos eventos.

Exemplos de fontes de eventos em Python/Tkinter:

- `<Button-1>` - Clique do mouse
  - **Fonte:** Widgets Tkinter (Button, Canvas, Frame, etc.)
  - **Produtor:** Objeto widget (ex: `Button(master, text="Clique")`)
- `<KeyPress>` - Pressionamento de tecla
  - **Fonte:** Widgets que podem receber foco ou a janela principal
  - **Produtor:** Objeto `Tk()` (janela) ou widgets como `Entry`, `Text`
- `<Motion>` - Movimento do mouse
  - **Fonte:** Widgets que podem detectar movimento do mouse
  - **Produtor:** Widgets como `Canvas`, `Frame`, janela principal
- `<<ListboxSelect>>` - Mudança de valor
  - **Fonte:** Widgets como Listbox, Combobox
  - **Produtor:** Widget que sofreu alteração no valor selecionado

2.2 Event Listeners (Ouvintes de Eventos)

O que é um event listener? Um *event listener* (ouvinte de eventos) é uma função ou objeto que "escuta" ou aguarda por eventos específicos de uma determinada fonte. Quando o evento ocorre, o *listener* é notificado e pode executar uma ação em resposta.

Diferença entre evento e listener:

- **Evento:** É o acontecimento em si (ex: "usuário clicou no botão")
- **Listener:** É o código que está registrado para ser executado quando o evento ocorrer

Para que serve um event listener? O *event listener* atua como um intermediário entre o produtor de eventos e o *handler*. Ele fica registrado em um produtor específico, aguardando por um tipo específico de evento, e quando esse evento ocorre, ele dispara a execução do handler associado.

Por que programas orientados a eventos são descritos como "ouvintes"? Porque o programa não executa um fluxo linear contínuo, mas sim fica em um estado de espera (normalmente em um loop principal).

Como registrar um listener em Tkinter: A.1

2.3 Event Handlers e Callback Functions

O que é um event handler? Um event handler (manipulador de eventos) é a função ou método que contém o código a ser executado quando um evento específico ocorre. Ele define como o programa deve reagir ao evento.

O que é uma função callback? Uma função callback é uma função que é passada como argumento para outra função, com a expectativa de que será chamada ("devolvida") em algum momento futuro, geralmente após a conclusão de uma operação assíncrona ou quando um evento ocorre.

Relação entre event handlers e callbacks:

- Um event handler é um tipo específico de callback, que é invocado quando um evento ocorre
- Nem toda callback está ligada a um evento (ex: callbacks para funções de ordenação com `key=` em Python)
- Todo event handler é uma callback, pois é passado para um listener e chamado posteriormente

Sequência evento → listener → handler:

1. Um evento ocorre (ex: usuário clica em um botão)
2. O sistema detecta o evento e notifica todos os listeners registrados para aquele evento naquela fonte
3. Cada listener executa seu handler (callback) associado
4. Os handlers realizam as ações apropriadas (ex: atualizar interface, processar dados)

Exemplo de event handler como callback: A.2

3 Paradigma Orientado à Eventos

No paradigma orientado a eventos, um sistema funciona assim: Componentes do sistema (produtores de eventos) ficam ativos e podem gerar eventos quando algo significativo acontece. Ouvintes de eventos (*listeners*) estão registrados para escutar eventos específicos de produtores específicos. Quando um evento ocorre, todos os *listeners* registrados para aquele evento naquela fonte são notificados, e seus *handlers* correspondentes (funções callback) são executados para processar o evento. Após a execução dos *handlers*, o sistema retorna ao loop principal aguardando novos eventos.

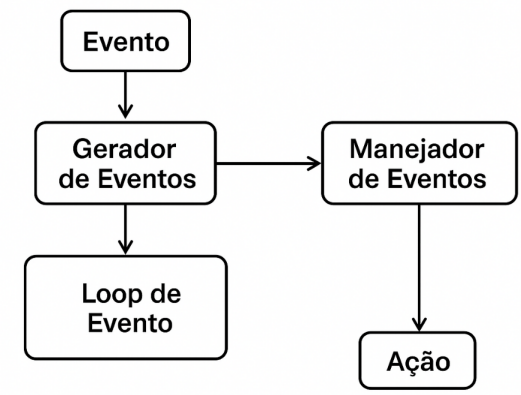

Figure 1: Diagrama conceitual da Programação Orientada a Eventos

4 Implementação Prática - Calculadora Básica

Enunciado: Implementar uma calculadora básica em Python com Tkinter que funcione inteiramente baseada em eventos de interação do usuário (cliques nos botões e entrada por teclado).

Requisitos Mínimos:

- **Interface (Tkinter):**
  - Display para mostrar números e resultados
  - Botões para dígitos 0-9
  - Botões para operações básicas: +, -, *, /
  - Botão C (clear) para limpar
  - Botão = para calcular
  - Layout organizado com grid ou pack
- **Lógica (Python):**
  - Cada botão deve ser um produtor de eventos

  - Registrar event listeners para eventos de clique
  - Handlers devem atualizar o display e realizar operações
  - Suporte a números de vários dígitos
  - Realizar operações básicas
- **Tratamento de erros:**
  - Prevenir divisão por zero
  - Lidar com expressões inválidas

A calculadora implementada e publicada no GitHub [1] aplica diretamente os princípios de programação orientada a eventos. Cada botão da interface gráfica é um produtor de eventos e utiliza o parâmetro *command* do Tkinter para registrar um *listener* associado a uma função *callback*.

Quando o usuário clica em um botão, o evento dispara a execução de *on_button_click()*, que recebe o caractere correspondente e decide a ação a ser executada. Se o caractere for "=", o *handler* tenta avaliar a expressão presente no campo de entrada usando *eval()*, exibindo o resultado ou uma mensagem de erro em caso de operação inválida. Para qualquer outro caractere (números, operadores ou ponto), o *callback* insere o valor no display da calculadora. Assim, toda a lógica da aplicação depende de eventos gerados pela interação do usuário, tratados dentro de funções responsáveis por atualizar o estado e o comportamento da interface.

5 Conclusão

Este estudo dirigido permitiu compreender os conceitos fundamentais da Programação Orientada a Eventos. Através da pesquisa teórica e da implementação prática da calculadora em Python/Tkinter, foi possível observar como os conceitos de produtores de eventos, *listeners* e *handlers* se integram para criar aplicações interativas e responsivas.

References

[1] Implementação da Calculadora. Disponível em: `https://github.com/msjujubr/Python_Calculator`. Acesso em: 06 dez. 2025.

[2] Documentação Oficial do Tkinter. Disponível em: `https://docs.python.org/3/library/tkinter.html`. Acesso em: 06 dez. 2025.

[3] LUTZ, Mark; ASCHER, David. Aprendendo Python. 2. ed. Rio de Janeiro: Alta Books, 2007.

A Códigos de Exemplo em Python/Tkinter

A.1 Código 1: Exemplo Básico de Eventos em Tkinter

```
import tkinter as tk

def ao_clicar(event):
    print("Botão clicado! Evento capturado com bind().")

root = tk.Tk()

botao = tk.Button(root, text="Clique aqui")
botao.pack()

# Registro do listener usando bind()
botao.bind('<Button-1>', ao_clicar)

root.mainloop()
```

A.2 Código-Exemplo 2:

```
import tkinter as tk

def processar_evento(event):
    # Este é o event handler (callback)
    tecla = event.char
    print(f"Tecla pressionada: {tecla}")

root = tk.Tk()

label = tk.Label(root, text="Pressione qualquer tecla")
label.pack()

# Registrando o event handler como callback
root.bind("<Key>", processar_evento)

root.mainloop()
```

B Código - Implementação da Calculadora Básica

```
import tkinter as tk
from tkinter import messagebox

class Calculadora:
    def __init__(self, master):
        self.master = master
        master.title("Calculadora Rudoilfo")

        self.resultado = tk.Entry(master, width=16, font=('Arial', 24),
                                  borderwidth=2, relief='ridge')
        self.resultado.grid(row=0, column=0, columnspan=4)

        self.criar_botoes()

        # Listener para teclado
        master.bind("<Key>", self.on_key_press)

    def criar_botoes(self):
        botoes = [
            ('7', 1, 0), ('8', 1, 1), ('9', 1, 2), ('/', 1, 3),
            ('4', 2, 0), ('5', 2, 1), ('6', 2, 2), ('*', 2, 3),
            ('1', 3, 0), ('2', 3, 1), ('3', 3, 2), ('-', 3, 3),
            ('0', 4, 0), ('.', 4, 1), ('=', 4, 2), ('+', 4, 3),
        ]

        for (text, row, col) in botoes:
            botao = tk.Button(self.master, text=text,
            width=5, height=2,
                              command=lambda t=text: self.on_button_click(t))
            botao.grid(row=row, column=col)

    # callback para botões e teclado
    def on_button_click(self, char):
        if char == '=':
            try:
                resultado = eval(self.resultado.get())
                self.resultado.delete(0, tk.END)
                self.resultado.insert(0, str(resultado))
            except Exception:
                messagebox.showerror("Erro", "Operação inválida")
                self.resultado.delete(0, tk.END)
        else:
            self.resultado.insert(tk.END, char)

    # listener para teclado
    def on_key_press(self, event):
        char = event.char

        # Se pressionar Enter
        if event.keysym == "Return":
            self.on_button_click("=")
            return

        # Se pressionar '='
        if char == "=":
            self.on_button_click("=")
            return

        # Se for número, operador ou ponto
        if char in "0123456789+-*/.":
            self.on_button_click(char)

if __name__ == "__main__":
    interface = tk.Tk()
    calculadora = Calculadora(interface)
    interface.mainloop()
```