Aula 52 — 06 junho

Conteúdos Avançados

Delegate

- Apontador para um método
  - ↳ Todos os overloads do método
    - ↳ Permite abstração dos métodos
- Evento: alterar/eventos na vida de um objeto que ele herda da sua classe base
  - ↳ exemplo: carregar num botão
  - ↳ delegate → digo o método desejado
    → ligo-o ao evento com o delegate
  - } Objetivo: aumentar abstração
    - ↓
    - daí retirar a ligação direta
      - ↓
      - o que aumenta a escalabilidade
        - ↓
        - alterar por trás com o mínimo de alterações possíveis
          - ↳ trabalhar por camadas
- Injeção de dependências: tirar parte do código e mudá-lo
  - ↳ interfaces: via delegates
- 1 Delegate → 1 Método
  - ↳ public delegate void Greeting(string name);
  - ↳ tem a mesma assinatura que os métodos para os quais aponta
    - ↳ Greeting greeting = EnglishGreeting;
      greeting("Ana");
      - ↳ só tenho de mudar isto no futuro
        - ↳ ganho escalabilidade
          - ↳ pouco impacto no código
- O delegate pode ser argumento do método para o tornar mais dinâmico
  - ↳ Substitui ifs e switchs
- Func<T> → para funções ⇒ função anónima → Func<string, int, bool> filter,
  - ↳ Action → para métodos
    - ↳ Usar Func e Action
      - ↳ mais eficazes que delegate

  (string: tipos dos inputs; int: tipo do output; filter: nome)

Eventos

- Publisher: sufixo Handler
  - ↳ public delegate void ZeroFoundHandler(object sender, EventArgs e);

    (object sender, EventArgs e): genéricos
- Delegate deve ser o mais genérico possível
  - ↳ object sender
- Invoke: dispara o evento
- O evento aponta para o método e o evento, o delegate e o método têm a mesma assinatura
- += : subscrever o evento
- Em eventos anónimos posso usar notação lambda

Expressões Lambda

- Alternativa a delegates (melhor método)
  - ↳ Método anónimo que substitui uma instância de um delegate
- Não substitui o delegate, apenas uma instância/definição de um delegate
  - ↳ delegate int Transformer(int x);

```
Transformer sqr = x => x * x;
Console.WriteLine(sqr(4));   // 16  (4 x 4)
```

  - ↳ Func<T> já substitui o delegate
    - ↳ Func<int, int> cube = (int x) => x * x * x;
      Console.WriteLine(cube(3)); // 27 (3 x 3 x 3)