



**iOS**

Criando Aplicações para iPhone e iPad

Classes e Objetos

**SoftBlue**  
www.softblue.com.br

Todos os direitos de cópia reservados. Não é permitida a distribuição física ou eletrônica desta matéria sem a permissão expressa e por escrito do autor.

---

---

---

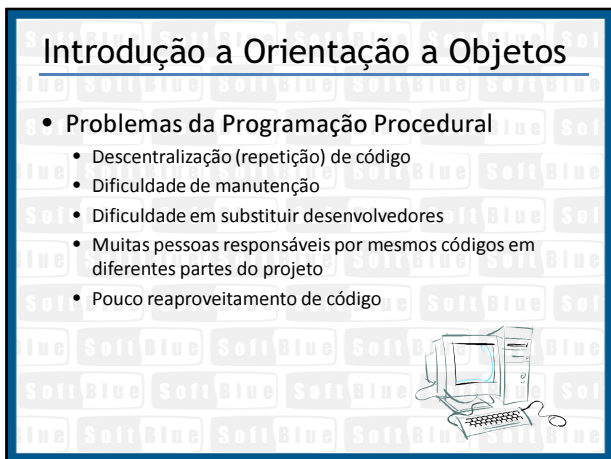
---

---

---


---

---



**Introdução a Orientação a Objetos**

- Problemas da Programação Procedural
  - Descentralização (repetição) de código
  - Dificuldade de manutenção
  - Dificuldade em substituir desenvolvedores
  - Muitas pessoas responsáveis por mesmos códigos em diferentes partes do projeto
  - Pouco reaproveitamento de código




---

---

---

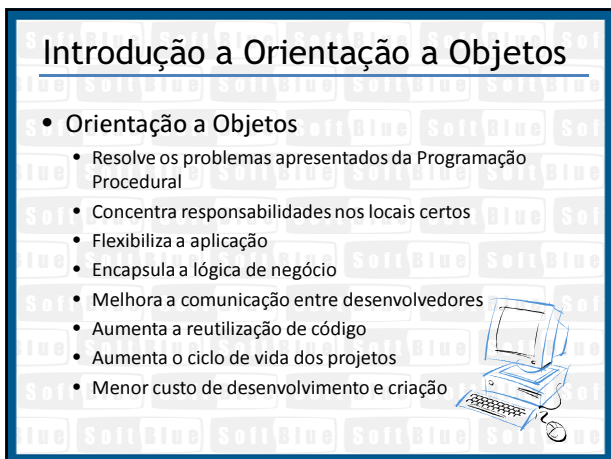
---

---

---


---

---



**Introdução a Orientação a Objetos**

- Orientação a Objetos
  - Resolve os problemas apresentados da Programação Procedural
  - Concentra responsabilidades nos locais certos
  - Flexibiliza a aplicação
  - Encapsula a lógica de negócio
  - Melhora a comunicação entre desenvolvedores
  - Aumenta a reutilização de código
  - Aumenta o ciclo de vida dos projetos
  - Menor custo de desenvolvimento e criação




---

---

---

---

---

---

---

---

## Orientação a Objetos

- Siglas
  - OO: Orientação a Objetos
  - LOO: Linguagem Orientada a Objetos
  - POO: Programação Orientada a Objetos
- Características
  - Códigos pertencem a classes
  - Classes possuem propriedades próprias
  - Classes possuem funcionalidades próprias
  - Classes interagem com outras classes

---

---

---

---

---

---

---

---

## Como funciona (visão geral)

### Modelo Procedural

nome  
sobrenome  
velocidade  
endereço  
anos

INSERT INTO USUARIOS...

INSERT INTO CARROS...

Cad. 1

Cad. 2

Cad. 3

INSERT..

INSERT..

INSERT..

### Modelo OO

[usuario nome]  
[usuario sobrenome]  
[carro velocidade]  
[carro endereço]  
[usuario anos]

[usuario salvar]

[carro salvar]

Cad. 1

Cad. 2

Cad. 3

Classe Usuario  
INSERT..

---

---

---

---

---

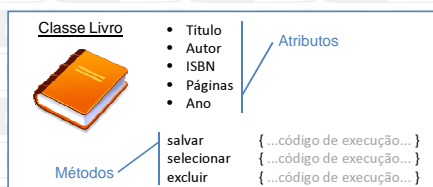
---

---

---

## Classes

- Estrutura de dados
- Representa um tipo de dado
- Comportamento próprio




---

---

---

---

---

---

---

---

## Atributos e Métodos

- **Atributos**
  - Características da classe
  - Tipos primitivos e/ou outras classes
  - Geralmente representados por **substantivos**
  - Exemplos: *nome, idade, endereço, cor, tamanho*
- **Métodos**
  - Ações que a classe pode realizar
  - Podem receber parâmetros e retornar valores
  - Geralmente representados por **verbos**
  - Exemplos: *salvar, ler, abrir, fechar, emprestar, devolver*

## Criando classes (.h)

- Operador **@interface**

```
@interface NomeDaClasse {
    /* Atributos */
}
/* Métodos (assinatura) */
@end
```

```
@interface Usuario {
    /* Atributos */
    NSString *nome;
    int idade;
}
/* Métodos (assinatura) */
- (NSString *) apresentar;
@end
```

## Criando classes (.m)

- Operador **@implements**

```
@implements NomeDaClasse
/* Métodos (implementação) */
@end
```

```
@implements Usuario
/* Métodos (implementação) */
- (NSString *) apresentar {
    return nome;
}
@end
```

## Objetos

- Classe não é um objeto
- Classe é um modelo de objeto (template)
- Objetos são instâncias de classes



## Ponteiros

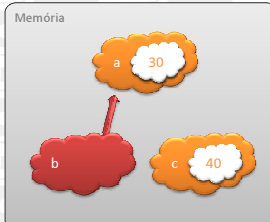
- O que é um ponteiro
- Representado pelo caracter asterisco (\*)

```
int a = 10; // 'a' é uma variável do tipo int
a = 20;    // 'a' agora possui o valor 20

int* b;    // 'b' é um ponteiro, ainda sem uso
b = &a;    // 'b' agora aponta para 'a'
b = 30;    // 'a' agora possui o valor 30

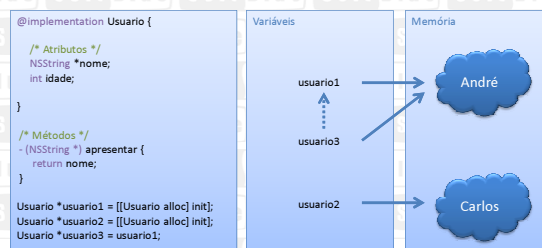
int c;     // 'c' é uma variável do tipo int
c = a;     // 'c' agora possui o valor 30
c = 40;    // 'c' agora possui o valor 40

// Qual o valor de 'a' neste momento?
// Resposta: 30
```



## Criando objetos

- Operador **init**
- Cria uma instância a partir de uma classe



## Construtores

- Método **init** customizado
- Customiza o código de inicialização da classe

```
@implementation Usuario {
    @public
    NSString *nome;
    int idade;
}

- (id) initWithName: (NSString *) n idade: (int) i {
    self->nome = n;
    self->idade = i;
    return self;
}

- (NSString *) apresentar {
    return nome;
}
@end
```

```
// Utilizando o construtor padrão
Usuario *usuario = [[Usuario alloc] init];
usuario->nome = @"André";
usuario->idade = 30;

// Utilizando o construtor customizado
Usuario *usuario = [[Usuario alloc] initWithName:
    @"André" idade:29];
```

## Operador **self**

- Diferencia um atributo do objeto de um atributo do método
- Fornece a referência do próprio objeto para outro método
- Funciona somente dentro do próprio objeto

```
@implementation Usuario {
    int var;
}

- (void) meuMetodo: (int) var {
    var;           // Argumento
    self->var;      // Propriedade
}
@end
```

## Modificadores de acesso

- Definem o acesso aos atributos e métodos
  - **private**: acesso apenas para a própria classe
  - **public**: acesso público

```
@implementation Modificadores {
    @public
    int publicVar;

    @private
    int privateVar;
}
@end
```

```
Modificadores *mod = [[Modificadores alloc] init];

mod->publicVar    // Correto
mod->privateVar   // Erro
```

## Forma de utilização

- Geralmente utilizados da seguinte forma:
  - Atributos são declarados como **private**
  - Métodos são declarados como **public**
  - Acesso realizado via **setters**, **getters** ou outros

## Getters e Setters

- Permitem interagir com os atributos
- Segurança: validações no lugar certo

```
@implementation Carro {
    @private
    int tanqueGasolina;
}

- (void) setGasolina: (int) tanqueGasolina {
    //Validações
    self->tanqueGasolina = tanqueGasolina;
}

- (int) getGasolina {
    return self->tanqueGasolina;
}

@end
```

## Getters e Setters booleanos

- **set** para definir valor
- **is** para consultar, ao invés de get


```
@implementation Carro {
    @private
    bool carroLigado;
}

- (void) setCarroLigado (bool) carroLigado {
    //Validações
    self->carroLigado = carroLigado;
}

- (bool) isCarroLigado {
    return self->carroLigado;
}

@end
```

Aulas práticas e manuais on-line



Assista agora às aulas práticas.

[Clique aqui](#) para visualizar as aulas práticas disponíveis

---

---

---

---

---

---

---