Programação Orientada a Objetos

Prof. Márcio Miguel Gomes

POO

- Tem como objetivo diminuir a distância entre o mundo real e o implementado em software
- Orientação a Objetos
  - O mundo é composto por objetos, físicos ou lógicos
  - Os objetos combinam atributos e funcionalidades
  - Problemas são modelados como objetos associados que interagem entre si

Atributos

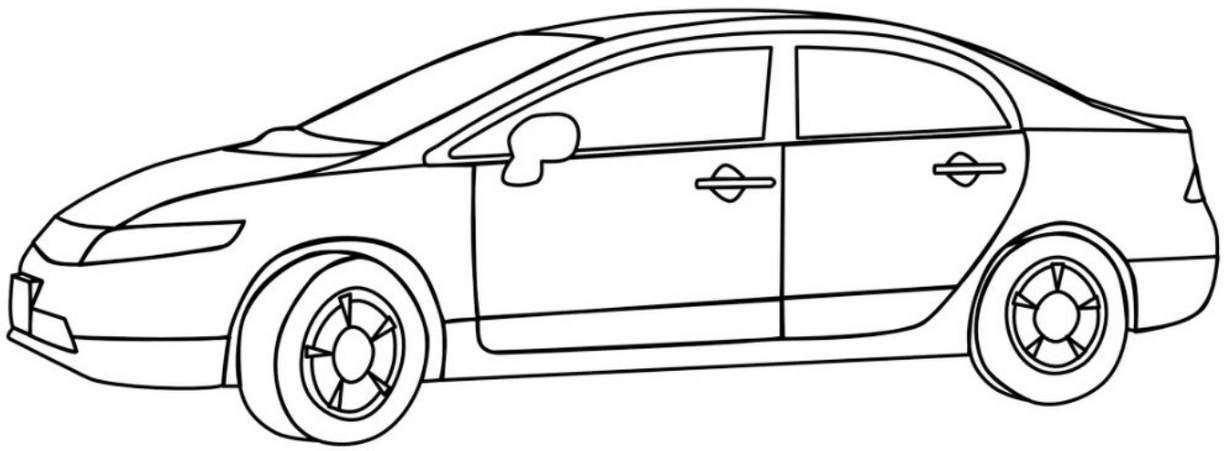
- São as características de um objeto
- Os dados que compõem uma classe
- classe Pessoa:
  - Nome
  - Idade
  - Sexo

Funcionalidades

- São a implementação das habilidades da classe
- Semelhantes a funções ou métodos, no entanto, estão vinculados a uma classe
- classe Pessoa:
  - Falar
  - Andar
  - Correr
  - Comer

Objeto Físico

- Classe “Veículo”
- Atributos
  - Marca, modelo, cor, potência, peso, combustível, etc
- Funcionalidades
  - Ligar, desligar, acelerar, frear, estacionar, abastecer, etc



Objeto Lógico

- Classe “Conta Bancária”
- Atributos
  - Tipo, correntista, saldo, limite de crédito, etc
- Funcionalidades
  - Depositar, sacar, transferir, pagar conta, tirar extrato, etc

EXTRATO DE CONTA CORRENTE E INVESTIMENTO
18/10/06 Conta: 9999 99999-9 12:03:45
ANTONIO COUTINHO 5 ESTRELAS

Data		Histórico	Valor
10/10		SALDO ANTERIOR	100,00
11/10	C	CEI 998554 DEP CHQ	250,00
11/10		SALDO	350,00
11/10		(-) SALDO A LIBERAR	250,00
11/10		SALDO FINAL DISPONIVEL	100,00
13/10	*	CEI SAQUE	50,00 -
13/10		SALDO	300,00
16/10	P	CREDITO SALARIO	1.000,00
16/10		SDO CTA/APL AUTOMATICAS	1.300,00
17/10	*	CEI SAQUE	300,00 -
17/10	*	CH COMPENSADO 325700	700,00 -

POSIÇÃO EM 18/10/2006.....

(+)	SDO PROV CTA/APL AUTOM	300,00
(=)	SALDO DISPONIVEL P/ SAQUE	300,00
(=)	VALOR TOTAL DISP P/ SAQUE	300,00

SDO DISP P/ APLIC HOJE C/ CPMF 300,00

Lançamentos Futuros

Data	Histórico	Valor
20/10	ELETROPAULO 054474809 5500	50,00 -

Débitos automáticos não efetuados

Data	Histórico	Valor
11/10	TIM CELULAR	400,00

Fundamentos de POO

- Auxiliam a administrar a complexidade de problemas
- Guiam toda a tarefa de modelagem computacional
- São eles:
  - Abstração
  - Encapsulamento
  - Modularidade
  - Hierarquia

Abstração

- Modelo computacional
- Visão simplificada, concentrada nos aspectos principais do problema a ser resolvido
- Podem ser objetos reais ou lógicos
- Um objeto “veículo” precisa de poucas informações para uma revenda, mas muitas para uma montadora. Você consegue identificá-los?

Encapsulamento

- Dados “protegidos” e acessados somente através de métodos do próprio objeto
- O programa executa trocando mensagens entre objetos
- Para acessá-los, primeiro identifica-se o objeto e depois o dado que se deseja acessar
- O mesmo ocorre com os métodos (funções)
- Um objeto “usuário” de um sistema não pode permitir acesso direto a sua senha. Então, como autenticar um login ou trocar a senha?

Encapsulamento

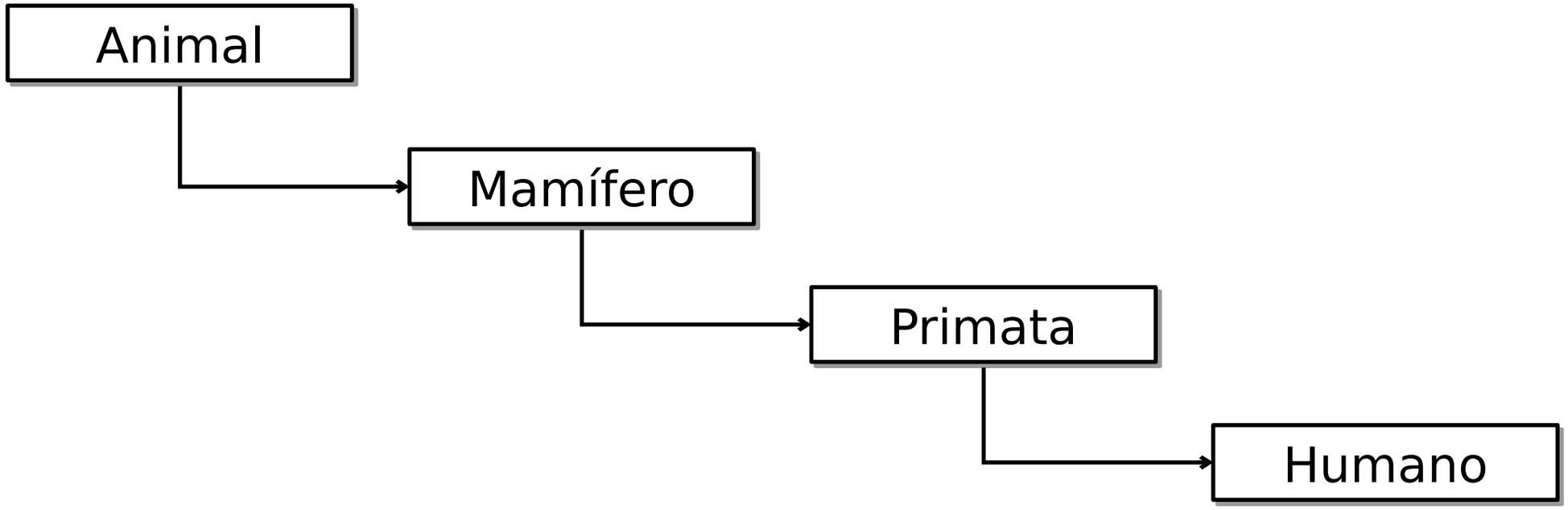
- **Público (public)**
  - Pode ser acessado por qualquer classe/objeto
- **Privado (private)**
  - Pode ser acessado somente pela própria classe/objeto
- **Protegido (protected)**
  - Pode ser acessado somente pela própria classe/objeto ou por classes/objetos “filhos”

Modularidade

- Segmenta as classes compostas em classes mais simples
- As classes podem ser utilizadas por outras classes e até por outros programas
- Classes são coesas, têm baixo acoplamento, não dependem de outras classes
- Interface simples e coerente
- Como poderíamos segmentar uma classe “bicicleta”?

Herança

- Classes mais específicas herdam recursos de suas classes pai
- Quais atributos e funções são definidos em cada nível?

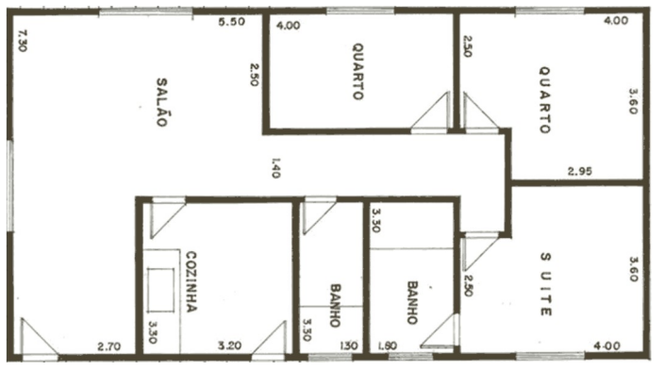


Classe

- Definição de um conjunto de características (atributos e funcionalidades) que representam o estado e o comportamento dos objetos definidos por essa classe
- Um objeto é uma instancia de uma classe
  - Classe = conceitual
  - Objeto = implementação

Classe vs Objeto

- Uma planta baixa é uma “classe”, um modelo conceitual
- Podemos construir inúmeras casas idênticas a partir de um mesmo projeto, mas cada casa será única
- Cada casa possui um número único
- Cada casa pode ter uma cor diferente



Exemplo de Classe - Python

```
class Pessoa:
    def tamanho_do_nome(self):
        return len(self.nome)

    def tamanho_do_sobrenome(self):
        return len(self.sobrenome)

    def set_nome(self, nome):
        self.nome = nome

    def get_nome(self):
        return self.nome

    def set_sobrenome(self, sobrenome):
        self.sobrenome = sobrenome

    def get_sobrenome(self):
        return self.sobrenome

    def set_sexo(self, sexo):
        self.sexo = sexo

    def get_sexo(self):
        return self.sexo

    def set_idade(self, idade):
        self.idade = idade

    def get_idade(self):
        return self.idade

    def tamanho_do_nome_completo(self):
        return self.tamanho_do_nome() +
               self.tamanho_do_sobrenome()
```

Uso da Classe - Python

```
nome = input('Qual o seu nome? ')
sobrenome = input('Qual o seu sobrenome? ')
sexo = input('Qual o seu sexo? ')
idade = int(input('Qual a sua idade? '))

p = Pessoa()
p.set_nome(nome)
p.set_sobrenome(sobrenome)
p.set_sexo(sexo)
p.set_idade(idade)

print('Nome:', p.get_nome())
print('Sobrenome:', p.get_sobrenome())
print('Sexo:', p.get_sexo())
print('Idade:', p.get_idade())
print('Tamanho do nome:', p.tamanho_do_nome_completo())
```

Exemplo de Classe - C++

```
class Pessoa {
    string nome, sobrenome, sexo; // Define os atributos da classe
    int idade;
private: // Define os métodos privados
    int tamanhoDoNome(){ return (int)this->nome.length(); }
    int tamanhoDoSobrenome(){ return (int)this->sobrenome.length(); }
public: // Define os métodos públicos
    Pessoa() { }
    void setNome(string nome) { this->nome = nome; }
    string getNome() { return this->nome; }
    void setSobrenome(string sobrenome) { this->sobrenome = sobrenome; }
    string getSobrenome() { return this->sobrenome; }
    void setSexo(string sexo) { this->sexo = sexo; }
    string getSexo() { return this->sexo; }
    void setIdade(int idade){ this->idade = idade; }
    int getIdade() { return this->idade; }
    int tamanhoDoNomeCompleto(){ return this->tamanhoDoNome() + this->tamanhoDoSobrenome(); }
};
```

Uso da Classe - C++

```
int main()
{
    string nome, sobrenome, sexo;
    int idade;
    cout << "Qual o seu nome?"<< endl;
    getline(cin, nome);
    cout << "Qual o seu sobrenome?"<< endl;
    cin >> sobrenome;
    cout << "Qual o seu sexo?"<< endl;
    cin >> sexo;
    cout << "Qual a sua idade?"<< endl;
    cin >> idade;

    Pessoa p;
    p.setNome(nome);
    p.setSobrenome(sobrenome);
    p.setSexo(sexo);
    p.setIdade(idade);

    cout << "Nome:" << p.getNome() << endl;
    cout << "Sobrenome:" << p.getSobrenome() << endl;
    cout << "Sexo:" << p.getSexo() << endl;
    cout << "Idade:" << p.getIdade() << endl;
    cout << "Tamanho do nome:" << p.tamanhoDoNomeCompleto() << endl;

    return 0;
}
```

Praticando

Classe Retangulo:

Atributos:
- base
- altura

Métodos:
- set_base(base)
- get_base()
- set_altura(altura)
- get_altura()
- area()

Classe Retangulo - Python

```python
class Retangulo:
  def set_base(self, base):
    self.base = base

  def get_base(self):
    return self.base

  def set_altura(self, altura):
    self.altura = altura

  def get_altura(self):
    return self.altura

  def area(self):
    return self.base * self.altura
```

```python
r1 = Retangulo()
r1.set_base(10.0)
r1.set_altura(8.5)
print(f'{r1.get_base()} x {r1.get_altura()} = {r1.area()}')

r2 = Retangulo()
r2.set_base(12.2)
r2.set_altura(6.8)
print(f'{r2.get_base()} x {r2.get_altura()} = {r2.area()}')
```

Classe Retangulo - C++

```
class Retangulo {
    float base, altura;
public:
    void setBase(float base) {
        this->base = base;
    }
    float getBase() {
        return this->base;
    }
    void setAltura(float altura){
        this->altura = altura;
    }
    float getAltura() {
        return this->altura;
    }
    float area() {
        return this->base * this->altura;
    }
};
```

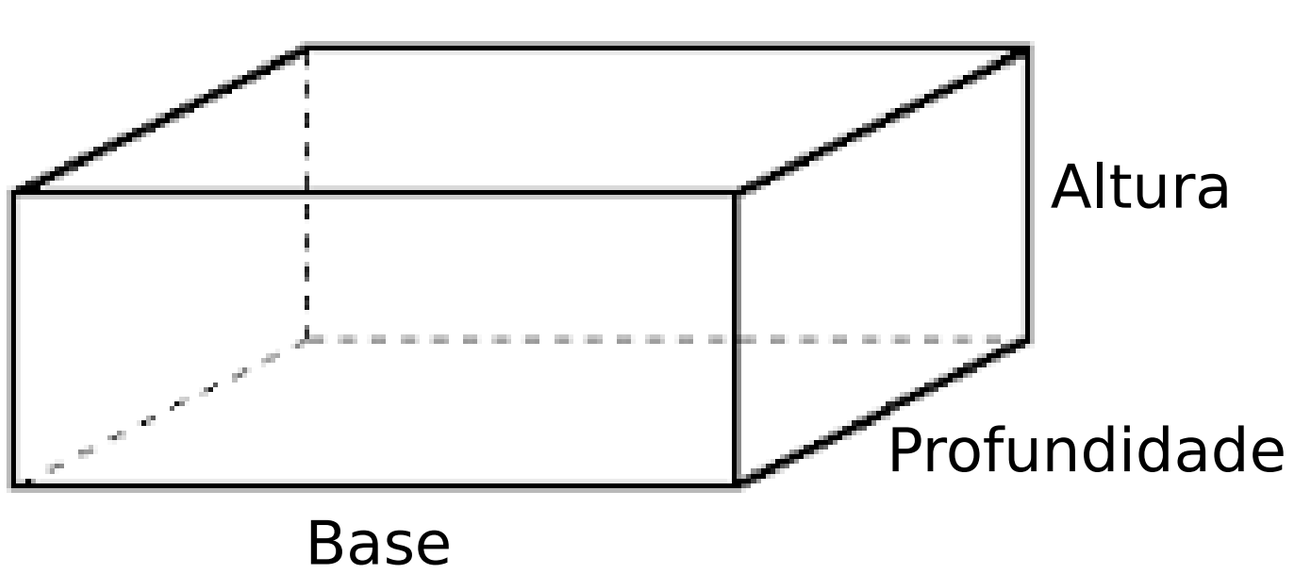
```
int main()
{
    Retangulo r1;
    r1.setBase(10.0);
    r1.setAltura(8.5);
    cout << r1.getBase() << " x " << r1.getAltura()
         << " = " << r1.area() << endl;

    Retangulo *r2 = new Retangulo();
    r2->setBase(12.2);
    r2->setAltura(6.8);
    cout << r2->getBase() << " x " << r2->getAltura()
         << " = " << r2->area() << endl;

    return 0;
}
```

Praticando

- Com base no exemplo anterior, crie a classe Paralelepipedo com a seguinte estrutura:
- Atributos:
  - Base
  - Altura
  - Profundidade
- Métodos:
  - Getter e Setter
  - Volume



Atividade

- Criar a classe “Contribuinte” para cálculo do imposto de renda simplificado
- Deve possuir os atributos: Nome do contribuinte, ano de nascimento, renda mensal e número de dependentes
- Deve possuir os métodos: Idade, renda anual, renda per capita mensal, base de cálculo, alíquota IR, alíquota IR efetiva, valor IR devido, imprimir
- Cada dependente reduz o valor base de cálculo do imposto em R$189,59 por mês
- Criar um programa que instancie um objeto e solicite que o usuário informe todos os atributos. Em seguida, deve fazer o cálculo do IR e imprimir todos os dados na tela
- Base = Renda Mensal – Dependentes * Desconto
- IR = (Base x Alíquota Correspondente) – Dedução

De	Até	Alíquota	Dedução
R$ 0,00	R$ 1.903,98	0,00%	R$ 0,00
R$ 1.903,99	R$ 2.826,65	7,50%	R$ 142,80
R$ 2.826,66	R$ 3.751,05	15,00%	R$ 354,80
R$ 3.751,06	R$ 4.664,68	22,50%	R$ 636,13
R$ 4.664,69	Sem Limite	27,50%	R$ 869,36