Programação Orientada à Objetos

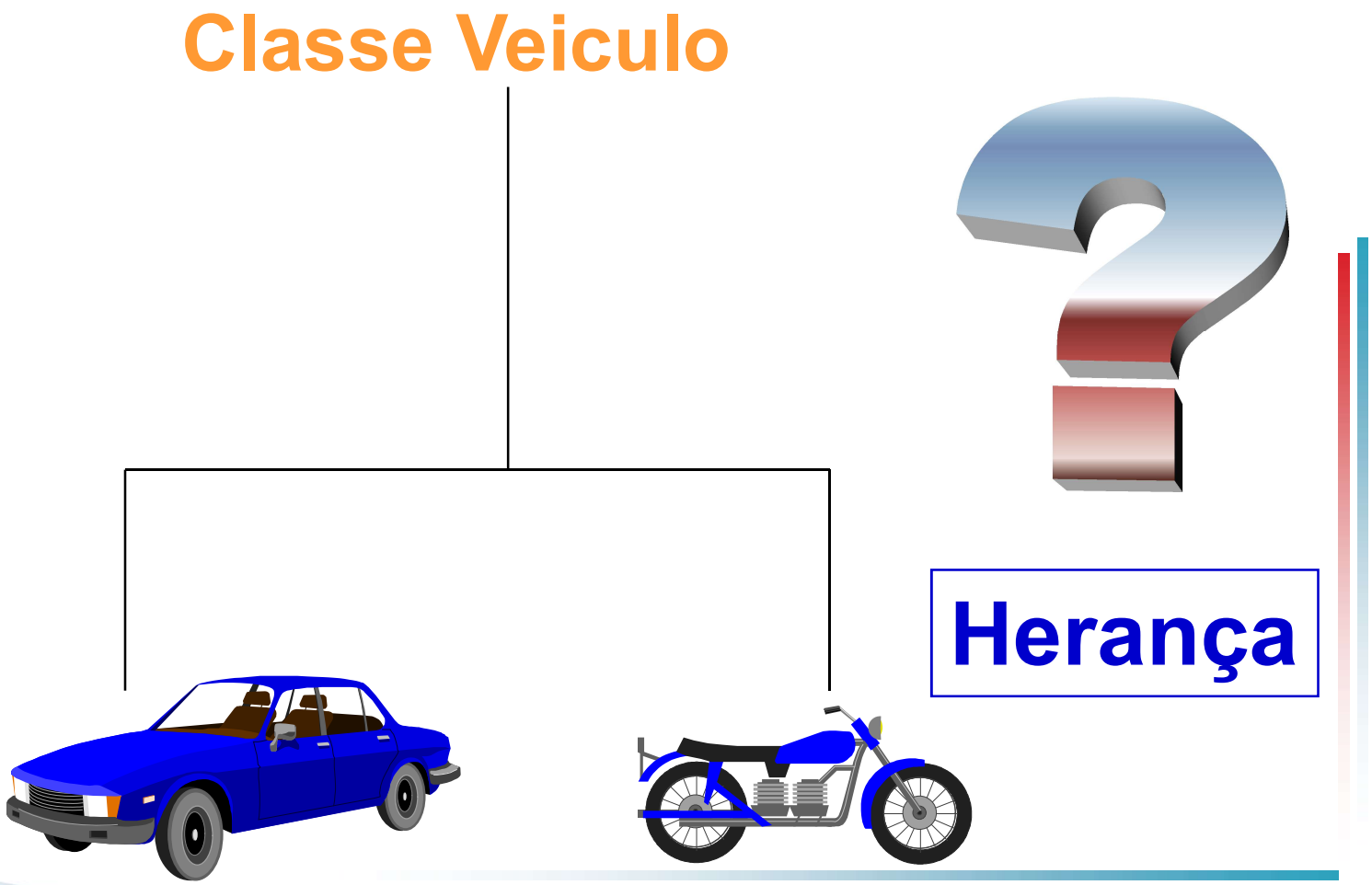
Herança

Prof. Alexandre Marcelino

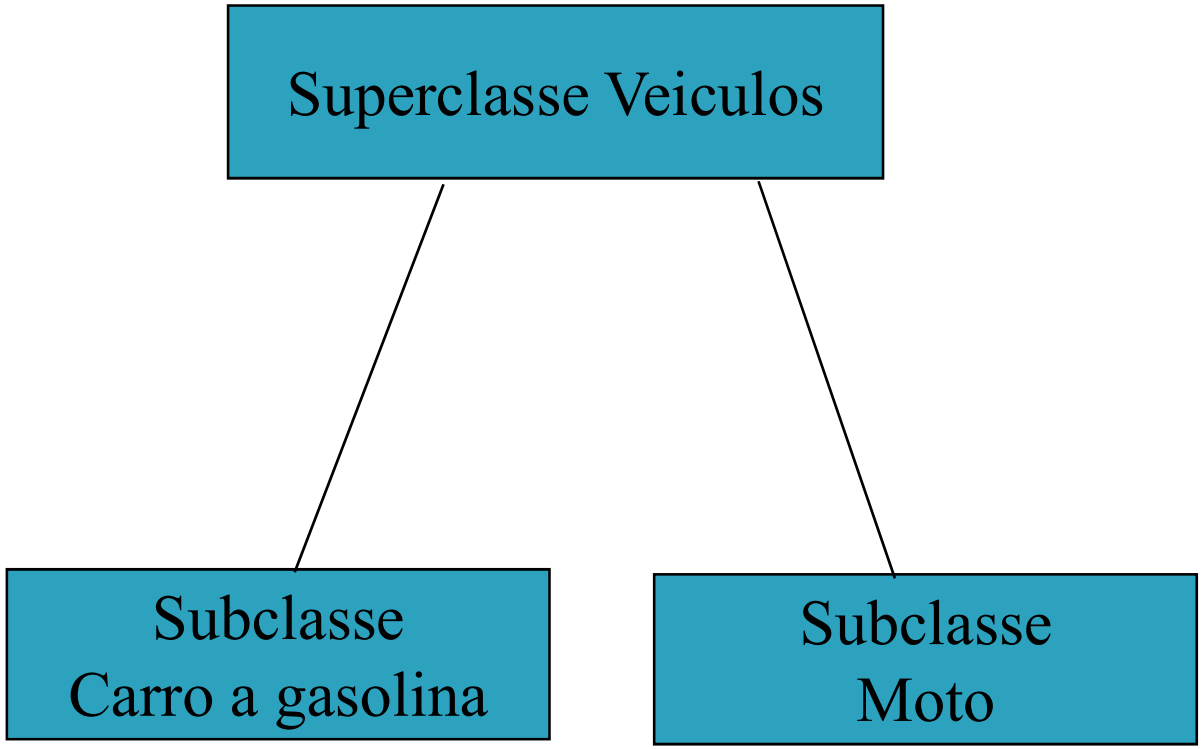
alexandre@vidadura.org (Skype/Facebbok)

alexandre@salesiano-ata.br

Herança

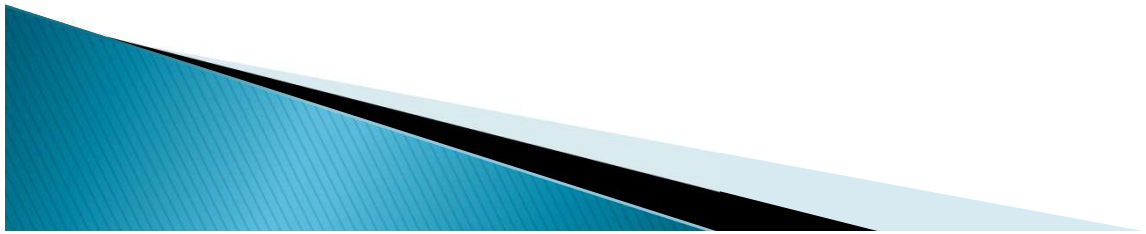


Herança


Superclasse Veiculos
Subclasse
Carro a gasolina
Subclasse
Moto

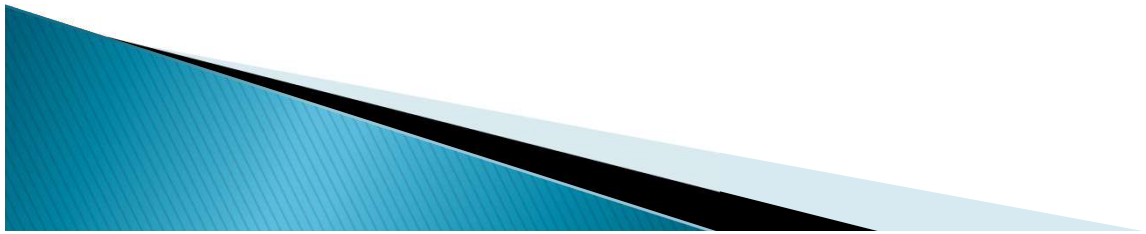
Herança

- Herança é um mecanismo da OO que permite criar novas classes a partir de classes já existentes, aproveitando-se das características já definidas na classe;
- Esta característica permite mecanismo o reuso e reaproveitamento de código existente, diminuindo o tamanho do código e tornando a programação mais eficiente;
- As novas classes são chamadas de classes derivadas (subclasses) e herdam as características da classes bases (superclasses);
- Uma subclasse pode herdar: Atributos, Métodos e Relacionamentos

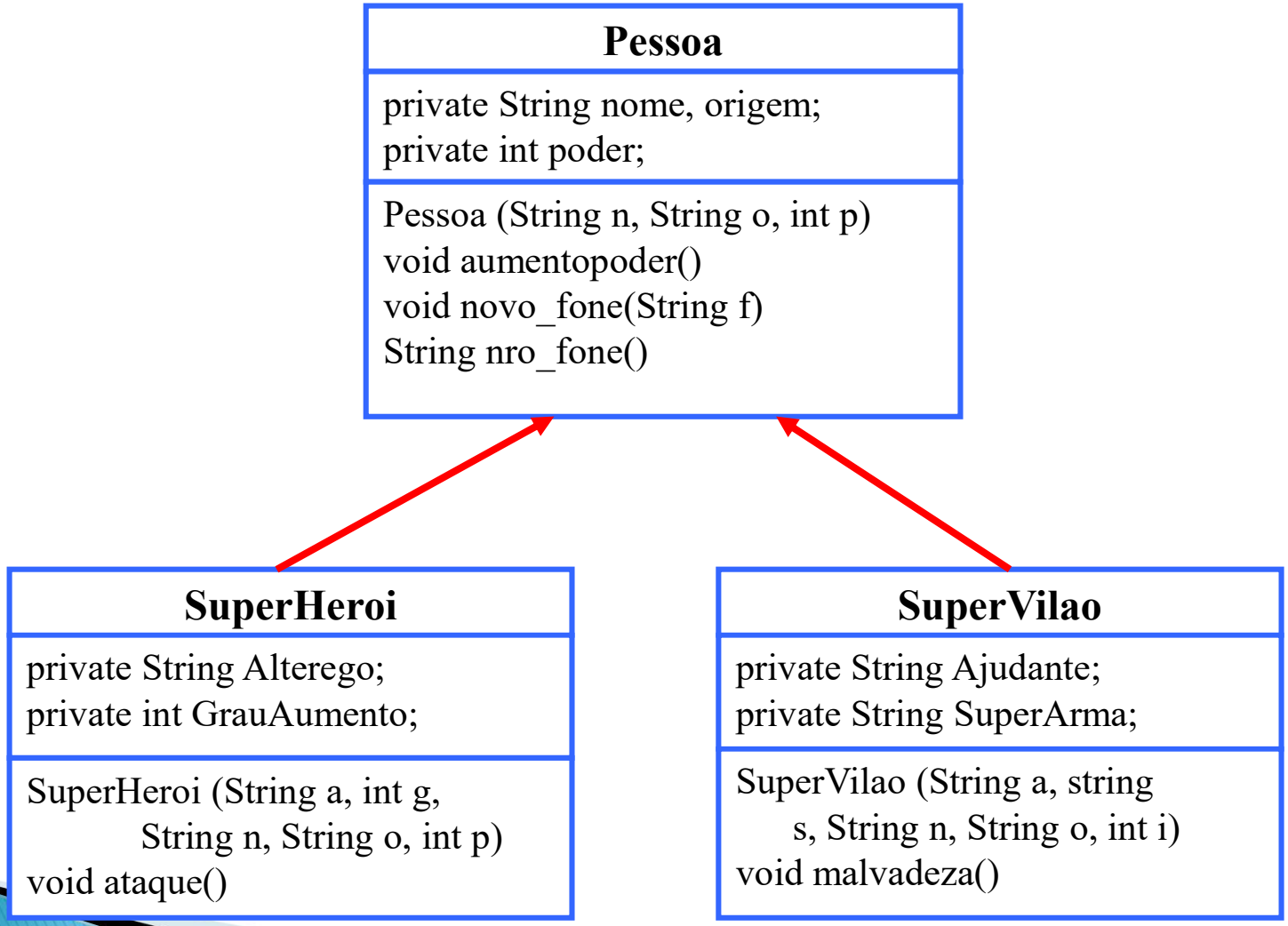


Herança

Na herança as classes filhas passam a atender pelos mesmos métodos e atributos public da classe pai, as classes filhas podem acrescentar métodos, atributos e até redefinir métodos herdados. Os atributos encapsulados (private) da classe pai não são acessíveis diretamente na classe filha a não ser que sejam qualificados como protected ou public.

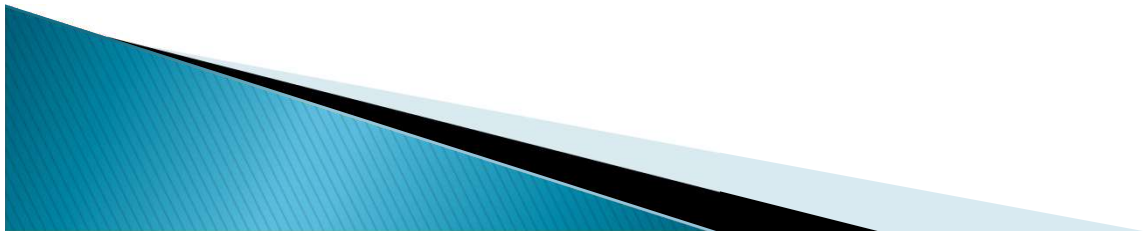


Herança

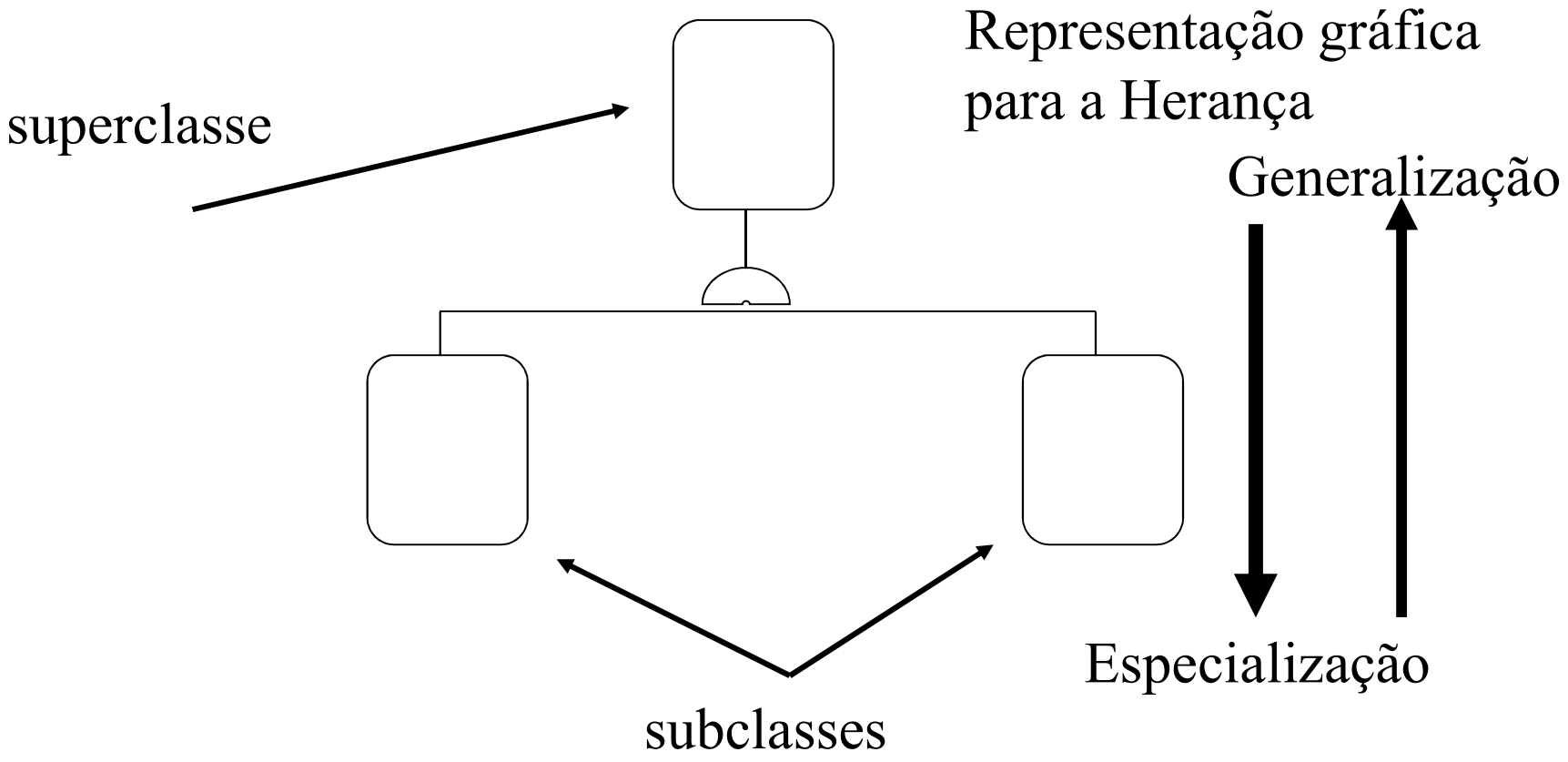

Pessoa
private String nome, origem;
private int poder;
Pessoa (String n, String o, int p)
void aumentopoder()
void novo_fone(String f)
String nro_fone()
SuperHeroi
private String Alterego;
private int GrauAumento;
SuperHeroi (String a, int g,
String n, String o, int p)
void ataque()
SuperVilao
private String Ajudante;
private String SuperArma;
SuperVilao (String a, string
s, String n, String o, int i)
void malvadeza()

Herança

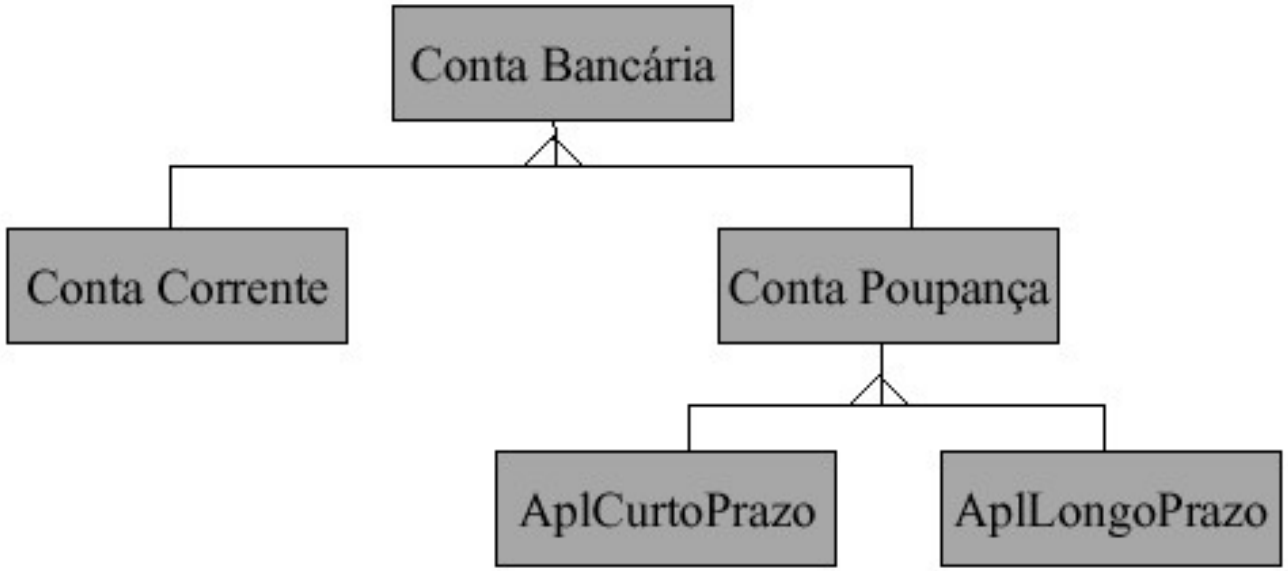
- As subclasses são mais especializadas do que as suas superclasses que são mais genéricas.
- Dizemos também que a subclasse é uma extensão da superclasse;
- Razões para estender classes:
  - alterar classes já existentes e adicionar propriedades ou comportamentos para representar outra classe de objetos;
  - criar uma hierarquia de classes que “herdam” propriedades e comportamentos de outra classe e definem novas propriedades e comportamentos.



Herança

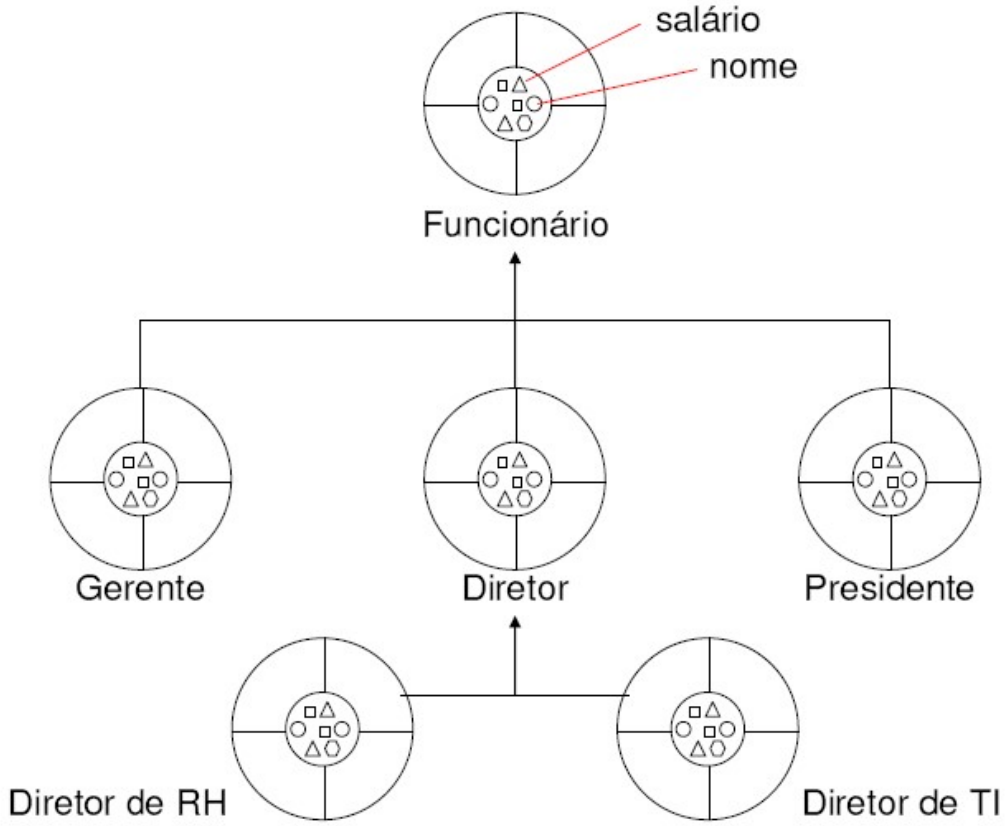

Representação gráfica
para a Herança
superclasse
Generalização
Especialização
subclasses

Herança



Hierarquia de Classes de Contas Bancárias

Herança



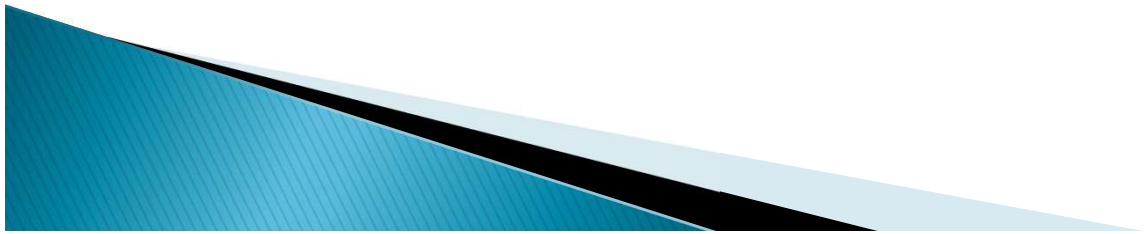
Na Orientação a Objetos as palavras classe base, supertipo, superclasse, classe pai e classe mãe são sinônimos, bem como as palavras classe derivada, subtipo, subclasse e classe filha também são sinônimos.

Herança

- Alterações feitas na superclasse afetam diretamente todas as classes descendentes;
- Termos como palavras classe base, supertipo, superclasse, classe pai e classe mãe são sinônimos, assim como palavras classe derivada, subtipo, subclasse e classe filha também são sinônimos;
- Herança simples X herança múltipla:
  - Herança simples é quando uma classe só herda características de uma única classe mãe;
  - Herança múltipla é quando uma classe herda características de mais de uma classe;
  - Algumas linguagens permitem herança múltipla como C++, outras não permitem herança múltipla que é o caso do Java;
  - Java contém um mecanismo para “aproximar” a herança múltipla que é o uso de interfaces.

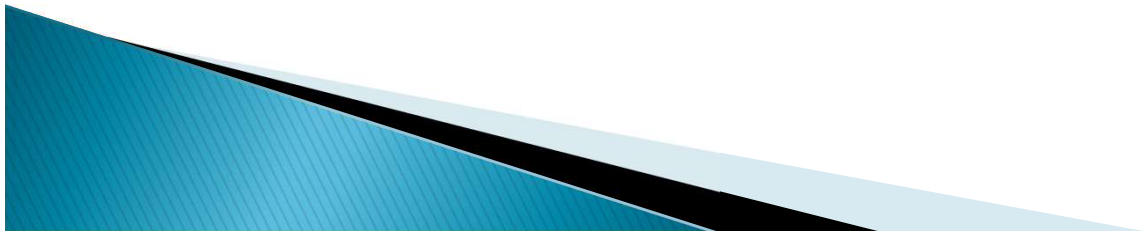
Herança

```
class Funcionario {    //  super classe
   public:       void setNome(std::string nome);
                 string getNome( );
   protected:    string nome; //protected: classe derivada pode acessar
                 string cpf;
                 double salario;
                 double getBonificação ( );
   };
   void Funcionario::setNome(std::string _nome) {
                                          nome = _nome;  }
    double getBonificação ( )    { return this.salario *0.10;  }
    string Funcionario:: getNome( )  { return  nome  }
         //idem para cpf e salario
```



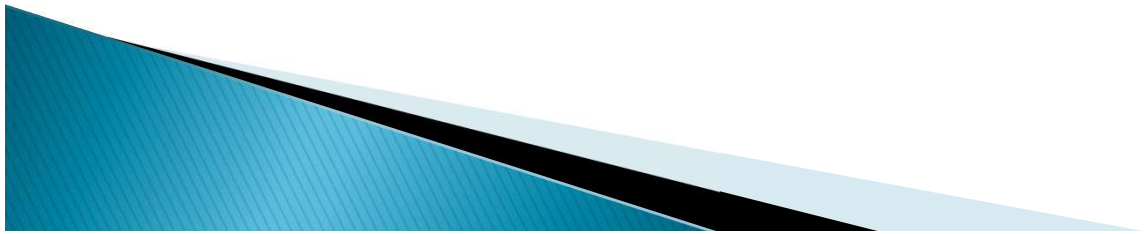
Herança

```
class Gerente :  public Funcionario {   //   gerente é
subclasse
        private:              int senha;
        public:               int numFuncGerenciados;
                        void setSenha(int senha);
                         boolean autentica (int senha);
 };
void   Gerente::setSenha(int _senha)  {   senha = _senha;  }
boolean Gerente::autentica (int _senha) {
                            if   (senha == _senha) return true
                            else return false;
                         }
```



Herança

- Atributos e métodos privados são herdados, mas não podem ser acessados diretamente;
- Além dos especificadores public e private, existe um outro especificador relacionado ao conceito de herança: protected;
- protected:
  - permite a visibilidade ao pacote e as subclasses de outros pacotes;
  - funciona como private sob ponto de vista externo a classe;
  - atributos protected são visíveis pelas classes derivadas, enquanto os private não o são;



Herança

Membros de Classe Protected

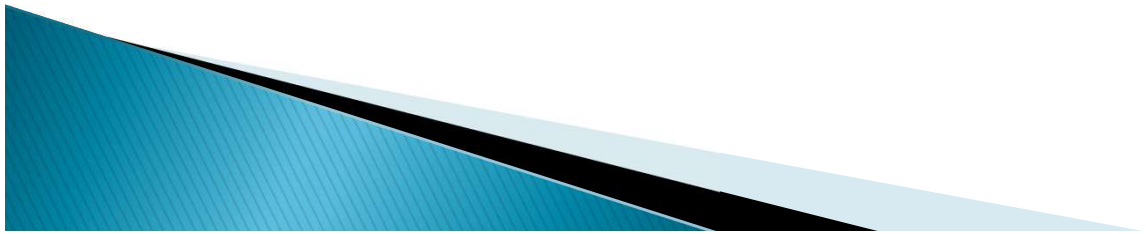
```
class A {
 private:
   int a;
 protected:
   int b;
 public:
   int c;
};
```

```
void main()
{
   A ca;
   B cb;

   ca.a = 1; // ERRO! a não é visível (private)
   ca.b = 2; // ERRO! b não é visível de fora (protected)
   ca.c = 3; // válido (c é public)

   cb.a = 4; // ERRO! a não é visível nem internamente em B
   cb.b = 5; // ERRO! b continua protected em B
   cb.c = 6; // válido (c continua public em B)
}
```

```
class B : public A {
 public:
   int geta() { return a; }  // ERRO!! a não é visível
   int getb() { return b; }  // válido (b protected)
   int getc() { return c; }  // válido (c public)
};
```

Herança

- Reescrita de Métodos:
  - Um método herdado de uma classe pode ser reescrito (sobreescrito / override), alterando seu comportamento;
- Um método herdado de uma classe pode ser reescrito (sobreescrito, sobreposto, override), alterando assim seu comportamento;
- As assinaturas (parâmetros) devem ser identicas;



Herança

```
class Gerente :  public Funcionario {   //   gerente é subclasse
        private:              int senha;
        public:               int numFuncGerenciados;
                        void setSenha(int senha);
                         boolean autentica (int senha);
 };
void   Gerente::setSenha(int _senha)  {   senha = _senha;  }
double Gerente::getBonificação ( )  { return this.salario *0.15;  }        //  sobreescrevendo o método da classe pai
boolean Gerente::autentica (int _senha) {
                        if   (senha == _senha) return true
                        else return false;
                    }
```