

# Herança

Carlos Arruda Baltazar  
UNIP – Cidade Universitária

Herança é um princípio de orientação a objetos, que permite que classes compartilhem atributos e métodos entre si. Na interação entre as classes, a herança adota um relacionamento esquematizado hierárquico. A partir desse conceito, temos dois tipos de classes:

- **Classe Base:** A classe que concede as características e comportamentos a uma outra classe.
- **Classe Derivada:** A classe que herda as características e comportamentos de uma classe base.

O fato de as classes derivadas herdarem atributos das classes bases assegura que programas orientados a objetos cresçam de forma linear e não geometricamente em complexidade. Cada nova classe derivada não possui interações imprevisíveis em relação ao restante do código do sistema.

Com o uso da herança, uma classe derivada geralmente é uma implementação específica de um caso mais geral. A classe derivada deve apenas definir as características que a tornam única.

Como exemplo pode-se observar as classes 'aluno' e 'professor', onde ambas possuem atributos como nome, endereço e telefone.

Professor	Aluno
- nome:String - endreco:Strinig - telefone:char[11] - funcional:char[9]	- nome:String - endreco:Strinig - telefone:char[11] - ra:char[7]
+ SetNome(nome:String):void + SetEndereco(endereco:String):void + SetTelefone(telefone:char[11]):void + SetFuncional(funcional:char[9]):void + GetNome():String + GetEndereco():String + GetTelefone():char[11] + GetFuncional():char[9]	+ SetNome(nome:String):void + SetEndereco(endereco:String):void + SetTelefone(telefone:char[11]):void + SetRa(ra:char[7]):void + GetNome():String + GetEndereco():String + GetTelefone():char[11] + GetRa():char[7]

```
package Pck_teste;

public class Aluno
{
    private String nome;
    private String endereco;
    private char[] telefone;
    private char[] ra;

    public String getNome()
    {
        return nome;
    }
    public void setNome(String nome)
    {
        this.nome = nome;
    }
    public String getEndereco()
    {
        return endereco;
    }
    public void setEndereco(String endereco)
    {
        this.endereco = endereco;
    }
    public char[] getTelefone()
    {
        return telefone;
    }
    public void setTelefone(char[] telefone)
    {
        this.telefone = telefone;
    }
    public char[] getRa()
    {
        return ra;
    }
    public void setRa(char[] ra)
    {
        this.ra = ra;
    }
}
```

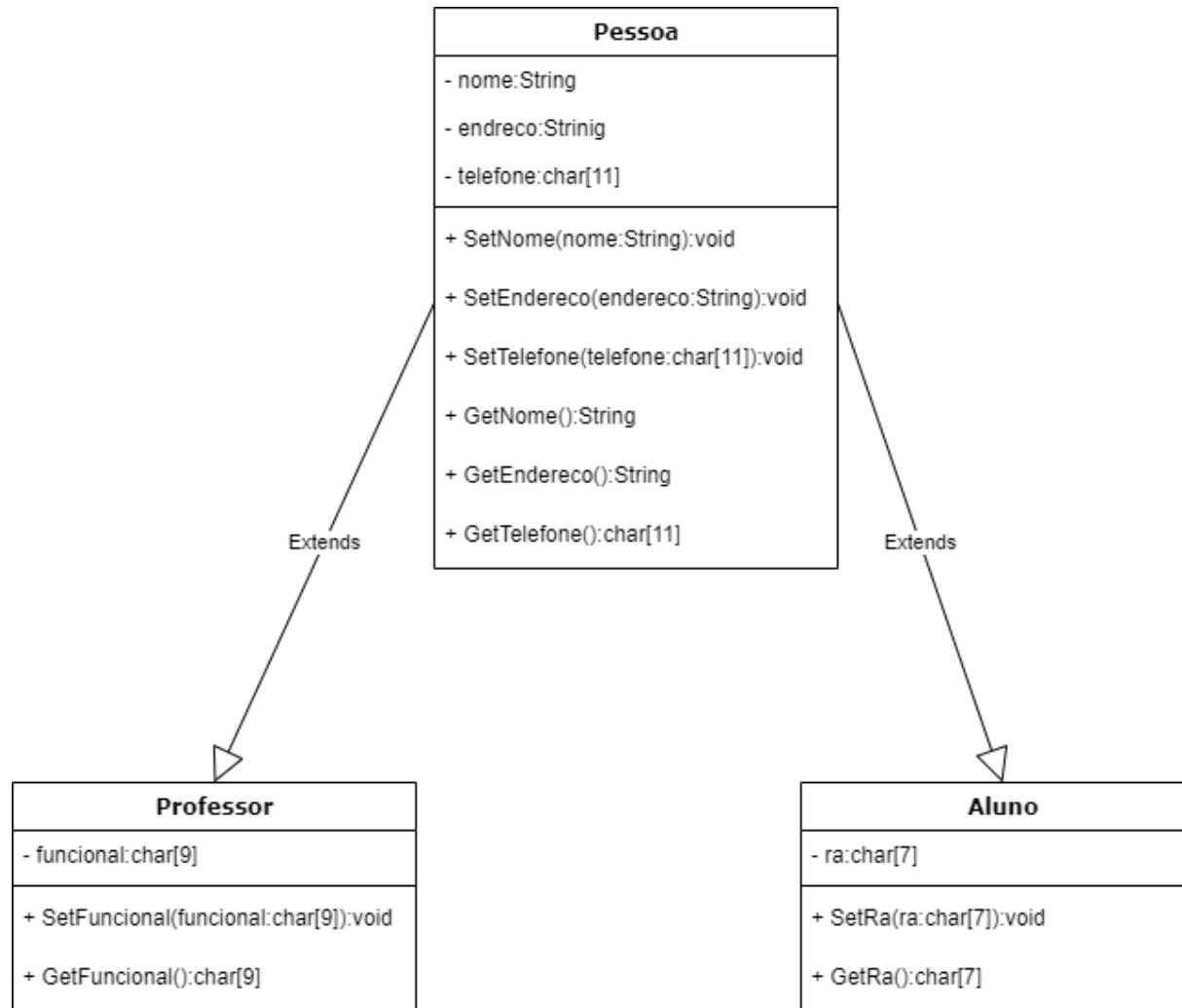
```
package Pck_teste;

public class Professor
{
    private String nome;
    private String endereco;
    private char[] telefone;
    private char[] funcional;

    public String getNome()
    {
        return nome;
    }
    public void setNome(String nome)
    {
        this.nome = nome;
    }
    public String getEndereco()
    {
        return endereco;
    }
    public void setEndereco(String endereco)
    {
        this.endereco = endereco;
    }
    public char[] getTelefone()
    {
        return telefone;
    }
    public void setTelefone(char[] telefone)
    {
        this.telefone = telefone;
    }
    public char[] getFuncional()
    {
        return funcional;
    }
    public void setFuncional(char[] funcional)
    {
        this.funcional = funcional;
    }
}
```

Nesse caso pode-se criar uma nova classe chamada por exemplo, 'pessoa', que contenha as semelhanças entre as duas classes, fazendo com que aluno e professor herdem as características de pessoa, desta maneira pode-se dizer que aluno e professor são subclasses de pessoa.





```
package Pck_teste;

public class Pessoa
{
    private String nome;
    private String endereco;
    private char[] telefone;

    public String getNome()
    {
        return nome;
    }
    public void setNome(String nome)
    {
        this.nome = nome;
    }
    public String getEndereco()
    {
        return endereco;
    }
    public void setEndereco(String endereco)
    {
        this.endereco = endereco;
    }
    public char[] getTelefone()
    {
        return telefone;
    }
    public void setTelefone(char[] telefone)
    {
        this.telefone = telefone;
    }
}
```

```
package Pck_teste;

import Pck_teste.Pessoa;

public class Aluno extends Pessoa
{
    private char[] ra;

    public void setRa(char[] ra)
    {
        this.ra = ra;
    }
}

package Pck_teste;

import Pck_teste.Pessoa;

public class Professor extends Pessoa
{
    private char[] funcional;

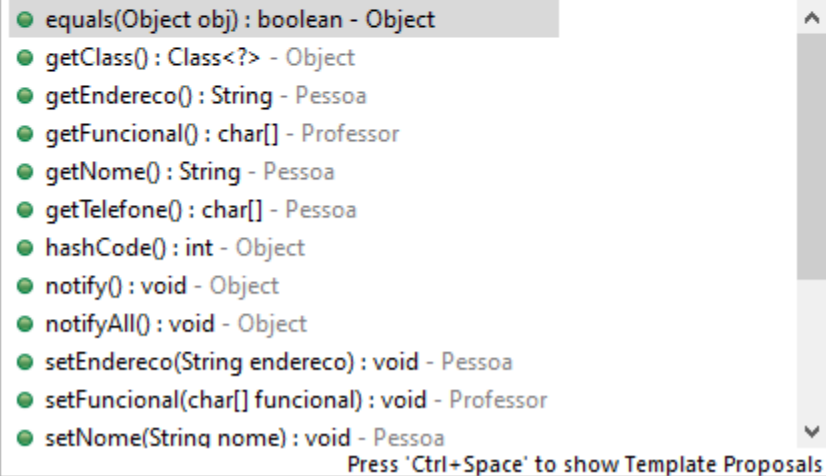
    public char[] getFuncional()
    {
        return funcional;
    }
    public void setFuncional(char[] funcional)
    {
        this.funcional = funcional;
    }
}
```

```
package Pck_teste;

import Pck_teste.Aluno;
import Pck_teste.Professor;

public class Main
{
    public static void main(String[] args)
    {
        Professor carlos = new Professor();
        Aluno objAluno = new Aluno();

        carlos.
    }
}
```



Autocomplete suggestions for `carlos.`:

- `equals(Object obj) : boolean - Object`
- `getClass() : Class<?> - Object`
- `getEndereco() : String - Pessoa`
- `getFuncional() : char[] - Professor`
- `getNome() : String - Pessoa`
- `getTelefone() : char[] - Pessoa`
- `hashCode() : int - Object`
- `notify() : void - Object`
- `notifyAll() : void - Object`
- `setEndereco(String endereco) : void - Pessoa`
- `setFuncional(char[] funcional) : void - Professor`
- `setNome(String nome) : void - Pessoa`

Press 'Ctrl+Space' to show Template Proposals

```
package Pck_teste;

import Pck_teste.Aluno;
import Pck_teste.Professor;

public class Main
{
    public static void main(String[] args)
    {
        Professor carlos = new Professor();
        Aluno objAluno = new Aluno();

        objAluno.
    }
}
```

- equals(Object obj) : boolean - Object
  - getClass() : Class<?> - Object
  - getEndereco() : String - Pessoa
  - getNome() : String - Pessoa
  - getTelefone() : char[] - Pessoa
  - hashCode() : int - Object
  - notify() : void - Object
  - notifyAll() : void - Object
  - setEndereco(String endereco) : void - Pessoa
  - setNome(String nome) : void - Pessoa
  - setRa(char[] ra) : void - Aluno
  - setTelefone(char[] telefone) : void - Pessoa
- Press 'Ctrl+Space' to show Template Proposals

# OBRIGADO