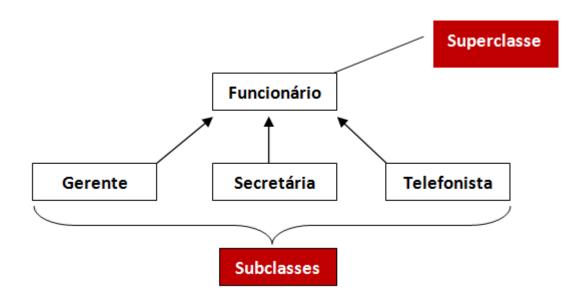
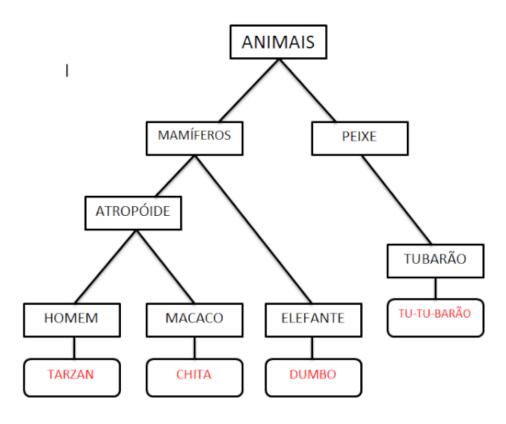
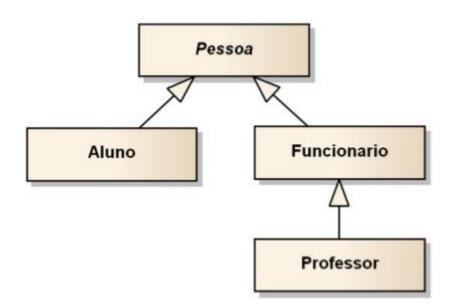
PROF. ME. HÉLIO ESPERIDIÃO

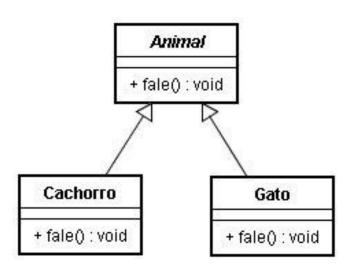
Permite que programadores aproveitem código de classes previamente construídas para criar novas classes.

As classes novas herdam todas as características das classes já criadas









#### Pessoa

nome : Stringendereco : Stringcidade : Stringcep : String

- estado : int - telefone : int

+ getNome(): String

+ setNome(nome : String) : void

+ getEndereco() : String

+ setEndereco(endereco : String) : void

+ getCidade() : String

+ setCidade(cidade : String) : void

+ getCep() : String

+ setCep(cep : String) : void

+ getEstado() : String

+ setEstado(estado : String) : void

+ getTelefone() : String

+ setTelefone(telefone : String) : void

#### PessoaFisica

- cpf : String

+ getCpf() : String

+ setCpf(cpf : String) : void

#### PessoaJuridica

- cnpj : String

+ getCnpj() : String

+ setCnpj(cnpj : String) : void

# Classe pai

```
public class Pai {
    public String nome;
    public int idade;
}
```

### Classe filha

#### **Extends pai**

Herda os atributos e métodos da classe pai

```
public class Filho extends Pai {
    public String propriedade1;
}
```

### Modo de uso

Note que o atributo nome e o atributo idade são da classe pai

```
public static void main(String[] args) {
  Filho f = new Filho();
  f.nome="Hélio";
  f.idade=30;
  f.propriedade1="test";
}
```

# Herança e construtores

```
public class Pai {
   public String nome;
   public int idade;

Pai() {
      this.nome="Sem nome";
      this.idade=0;
   }
}
```

```
public class Filho extends Pai {
   public String propriedade1;
}
```

```
public static void main(String[] args) {
  Filho f = new Filho();
  System.out.println(f.nome);
}
```

Qual valor será mostrado?

# Herança e construtores

```
public class Pai {
   public String nome;
   public int idade;
   Pai(String nome) {
      this.nome=nome;
      this.idade=0;
   }
}
```

```
public class Filho extends Pai {
   public String propriedade1;

   public Filho(String nome) {
      super(nome);
   }
```

```
public static void main(String[] args) {
  Filho f = new Filho("Hélio");
  System.out.println(f.nome);
}
```

Qual valor será mostrado?

# Herança e construtores

```
public class Pai {
   public String nome;
   public int idade;

Pai() {
      this.nome="Sem nome";
      this.idade=0;
   }
}
```

```
public class Filho extends Pai {
   public String propriedade1;

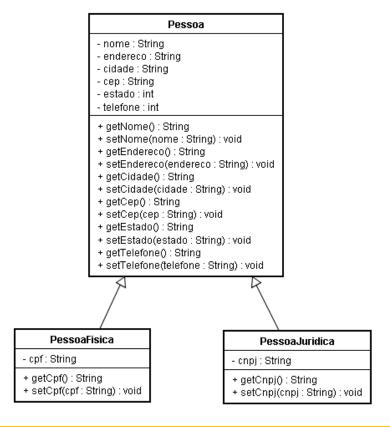
public Filho(String nome) {
      super(nome);
      this.propriedade1="novo valor";
   }
}
```

```
public static void main(String[] args) {
  Filho f = new Filho("Hélio");
  System.out.println(f.propriedade1)4;
}
```

Qual valor será mostrado?

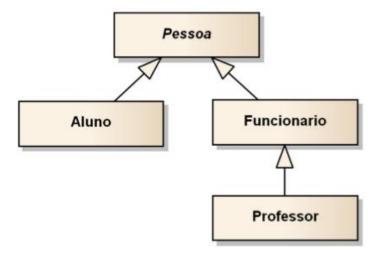
### Exercícios.

Crie uma classe pai Pessoa e duas classes filhas PessoaFisica e PessoaJuridica.



# Exercícios

Análise o diagrama abaixo e construa as classes necessárias para sua implementação.



### Exercícios.

Abra os arquivos do exercício do prontuário eletrônico e implemente os conceitos de orientação a objetos para facilitar o entendimento do código.