Orientada a Objetos

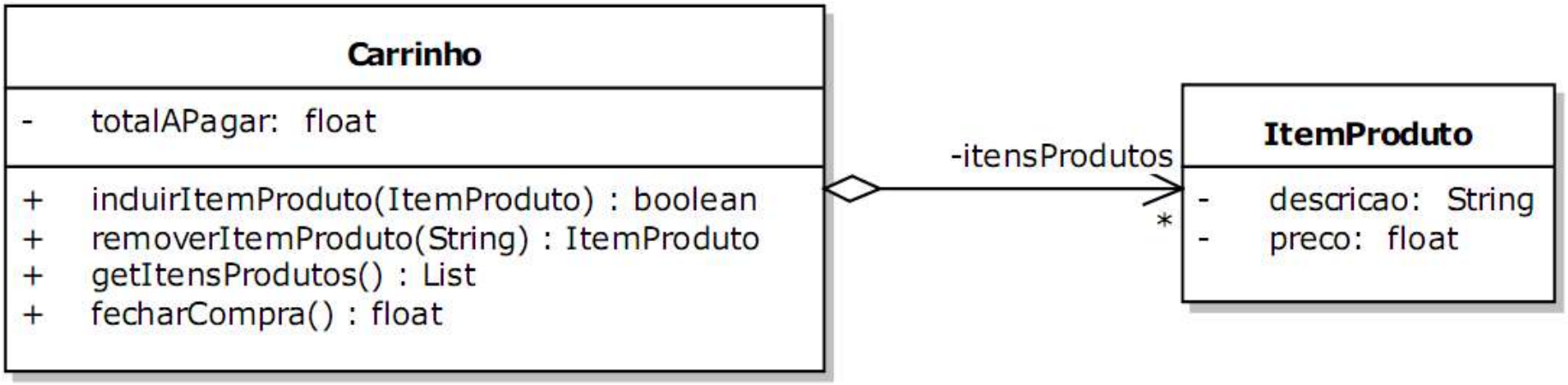
Herança

Profa.: Márcia Sampaio Lima

EST - UEA

Herança

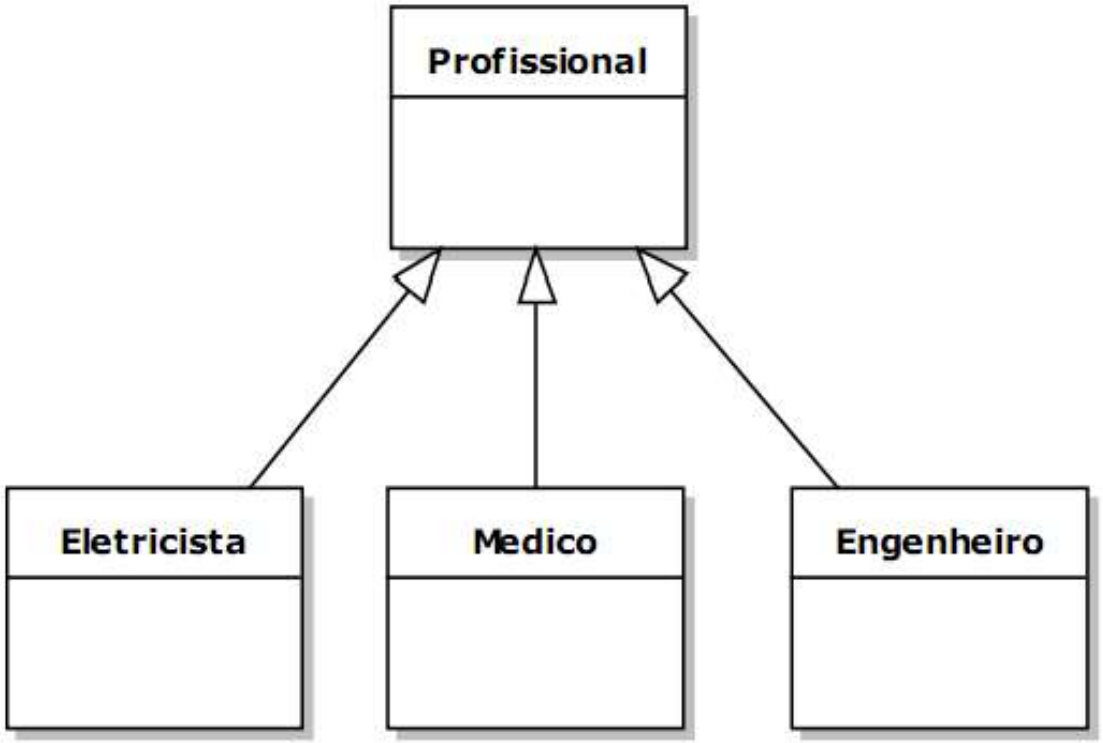
- Vimos relacionamentos em que um objeto contém uma ou mais instâncias de outro. São os relacionamentos chamados "tem-um".
- Exemplo: um carrinho tem muitos itens de produtos.



Herança

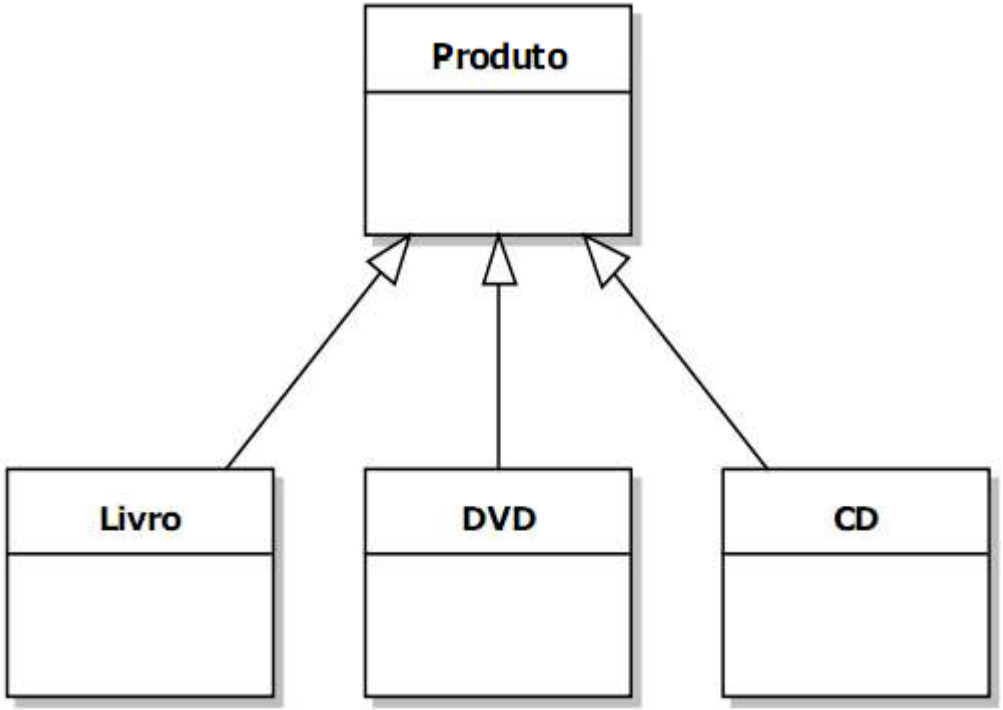
- Há relacionamentos que indicam que um objeto é um tipo mais específico de outro objeto.
  - Um engenheiro é um tipo de profissional.
  - Um médico é um tipo de profissional.
  - Profissional é uma generalização de engenheiro, médico, etc.

Herança



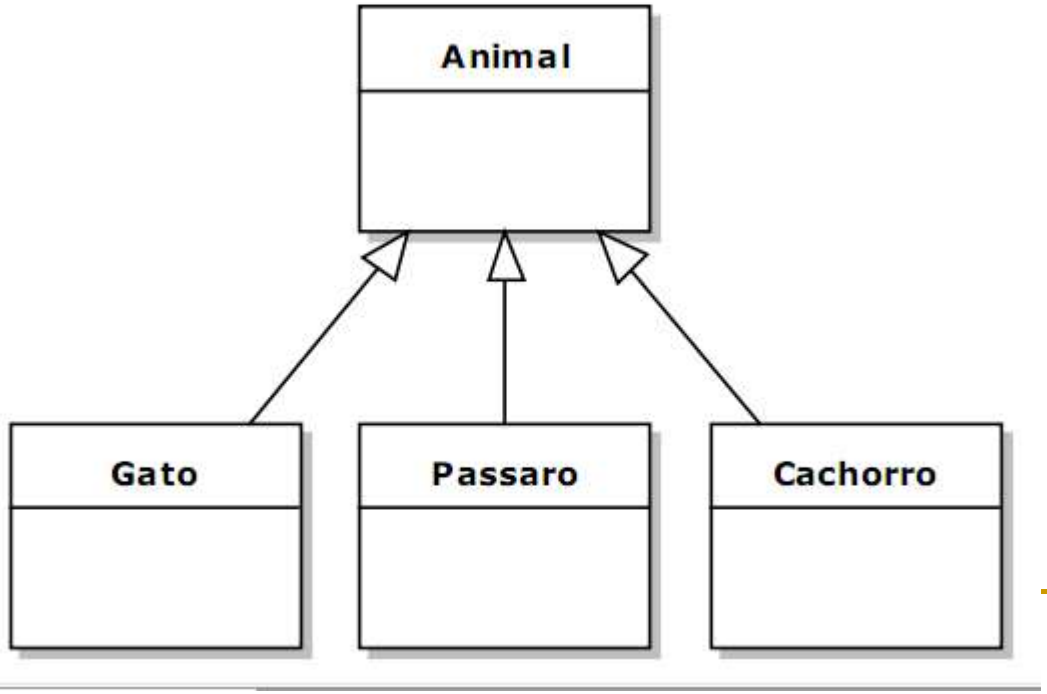
Herança

- Um livro (ou DVD, CD) é um tipo de produto.
  - Produto é uma generalização de livro, CD, DVD, etc.



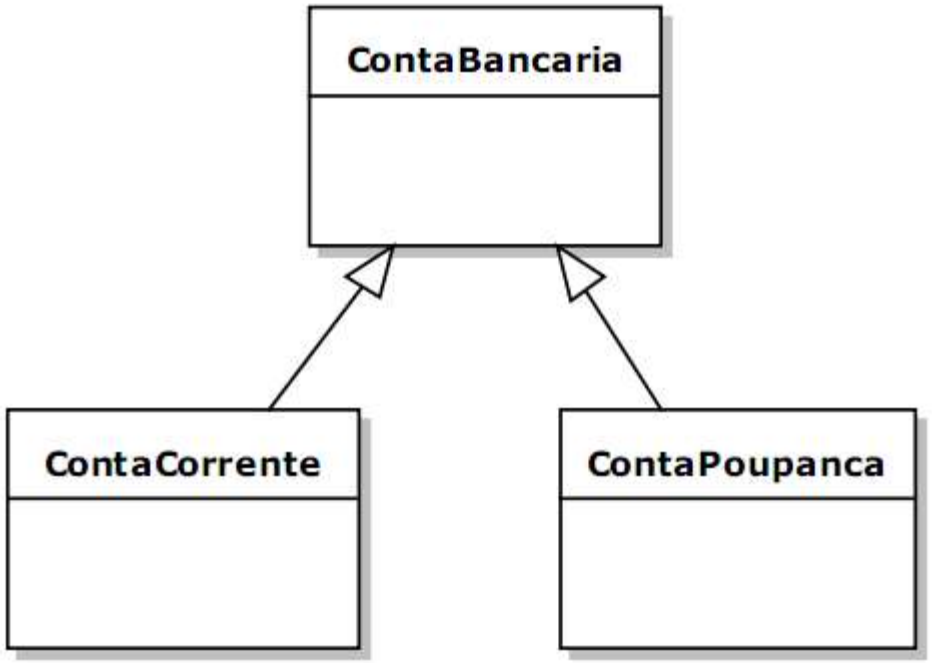
Herança

- Um gato (ou cachorro, etc.) é um tipo de animal.
  - Animal é uma generalização de gato, cachorro, etc.



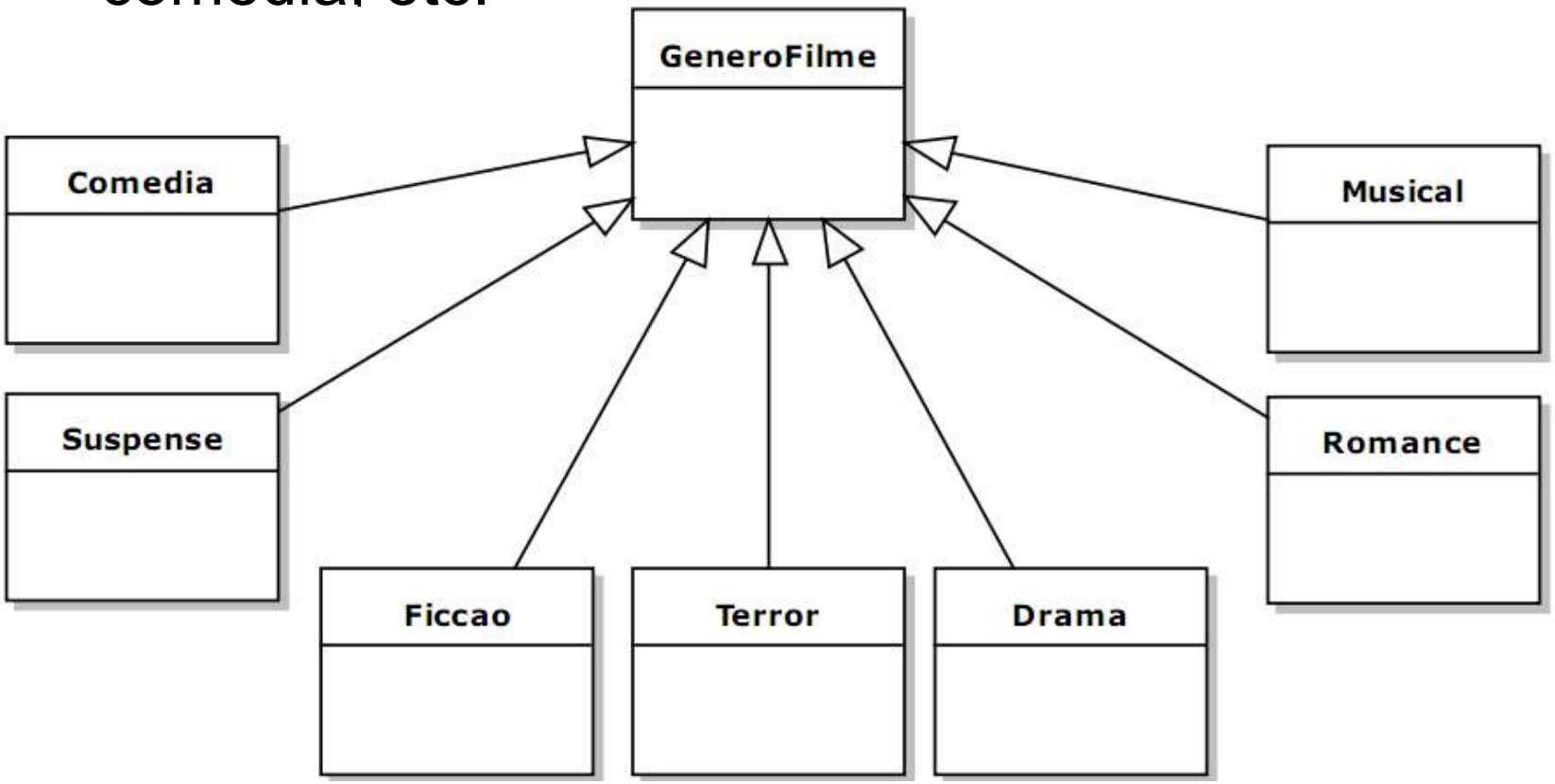
Herança

- Uma conta corrente ou uma poupança é um tipo de conta bancária.
  - Conta bancária é uma generalização de conta corrente e conta poupança.

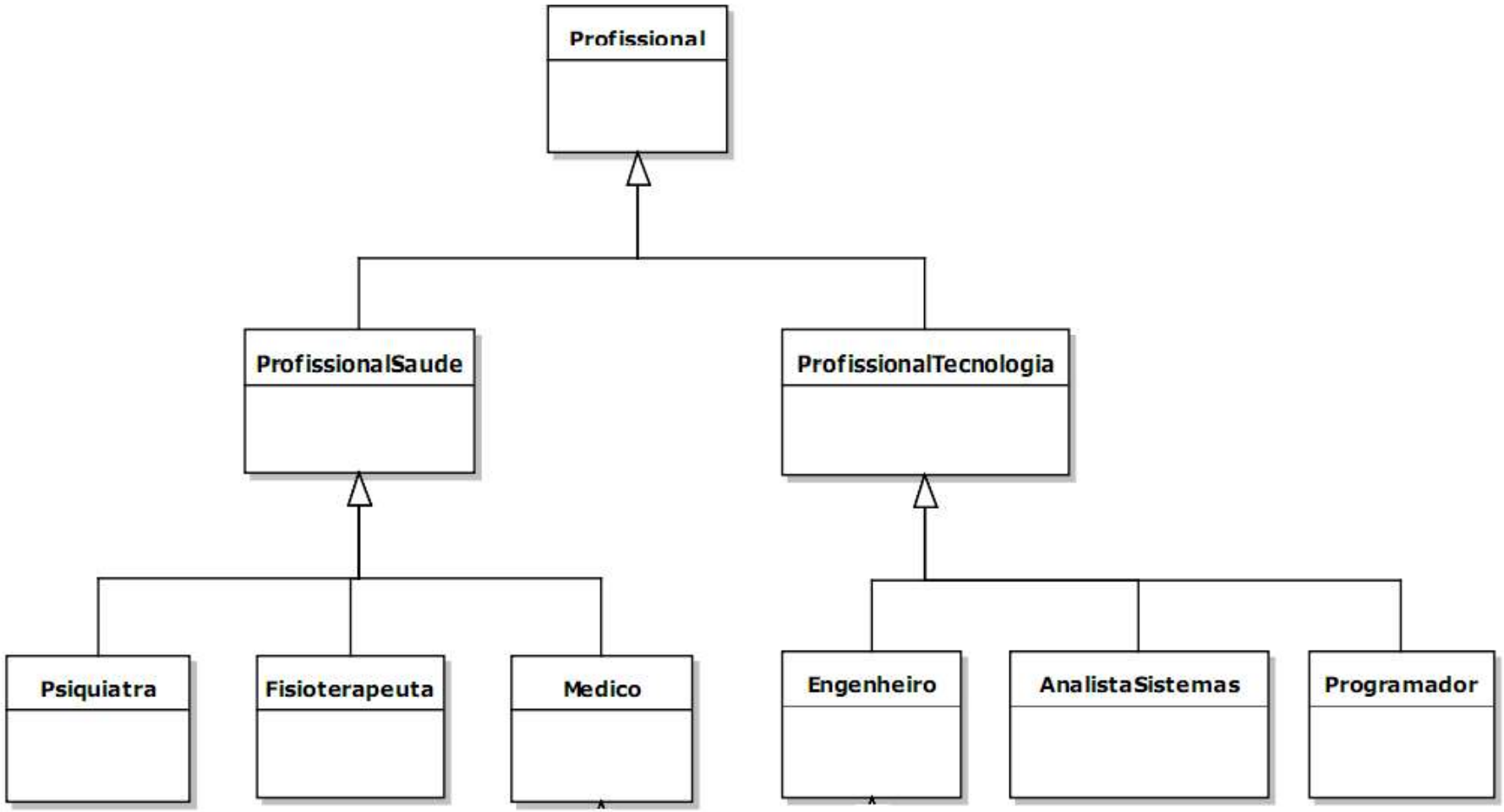


Herança

- Um musical é um tipo de gênero de filme.
  - Gênero de filme é uma generalização de musical, comédia. etc.



Herança


Profissional
ProfissionalSaude
ProfissionalTecnologia
Psiquiatra
Fisioterapeuta
Medico
Engenheiro
AnalistaSistemas
Programador

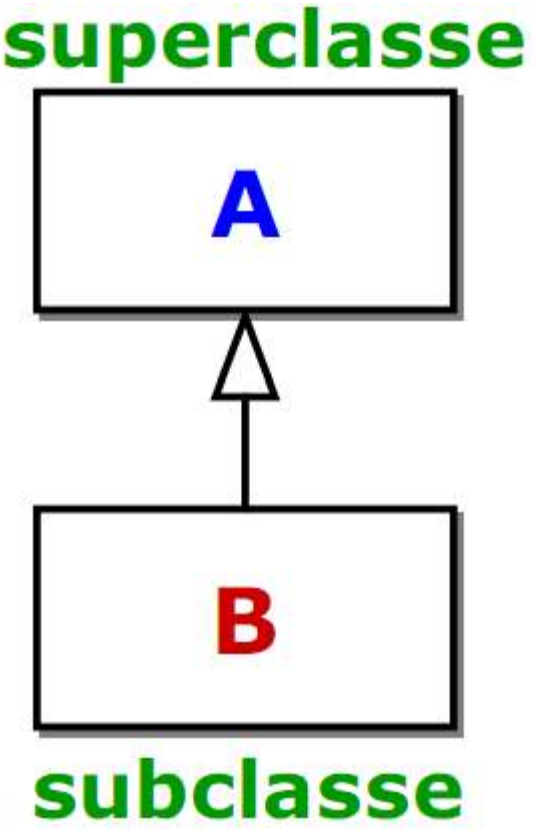
Herança

- Generalização:
  - Tipo de relacionamento entre dois objetos a e b em que:
    - a é mais geral que b;
    - b é um tipo específico de objeto a.
- Generalização também é chamada de relacionamento "é-um" ou "é-um-tipo-de".
- Exemplos:
  - um oftalmologista é um médico.
  - um oboé é um tipo de instrumento musical.

Herança

- Sejam A e B classes ligadas por uma generalização.

- Se A é a classe mais geral, e B é a classe específica, então:
  - A é chamada superclasse de B;
  - B é chamada subclasse de A.

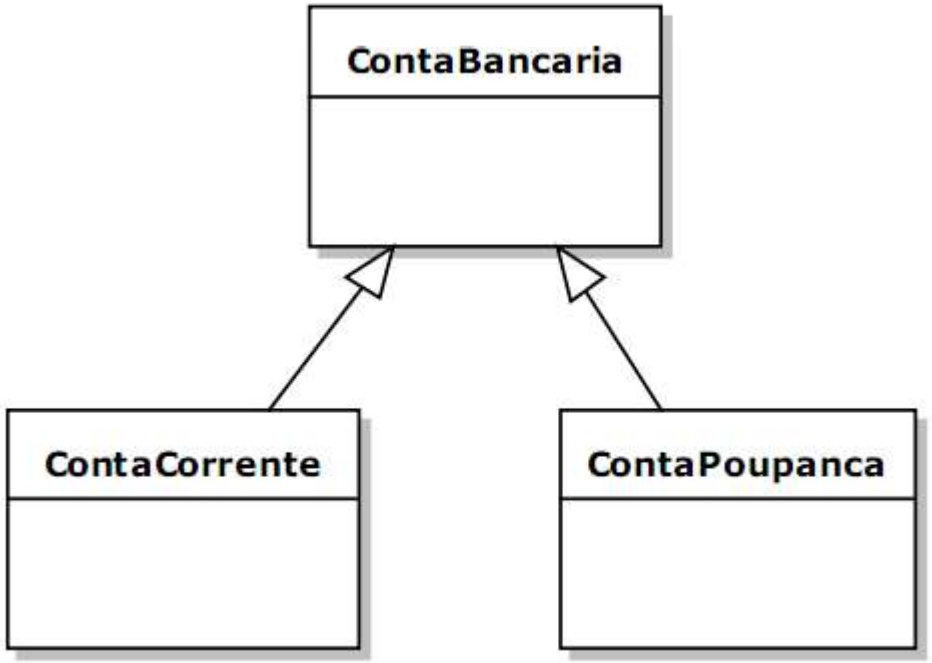


Herança

- Herança:
  - Mecanismo que nos permite definir uma nova subclasse apenas estendendo a definição de uma superclasse já existente.
  - Ponto de vista estrutural:
    - herança simples;
    - herança múltipla.

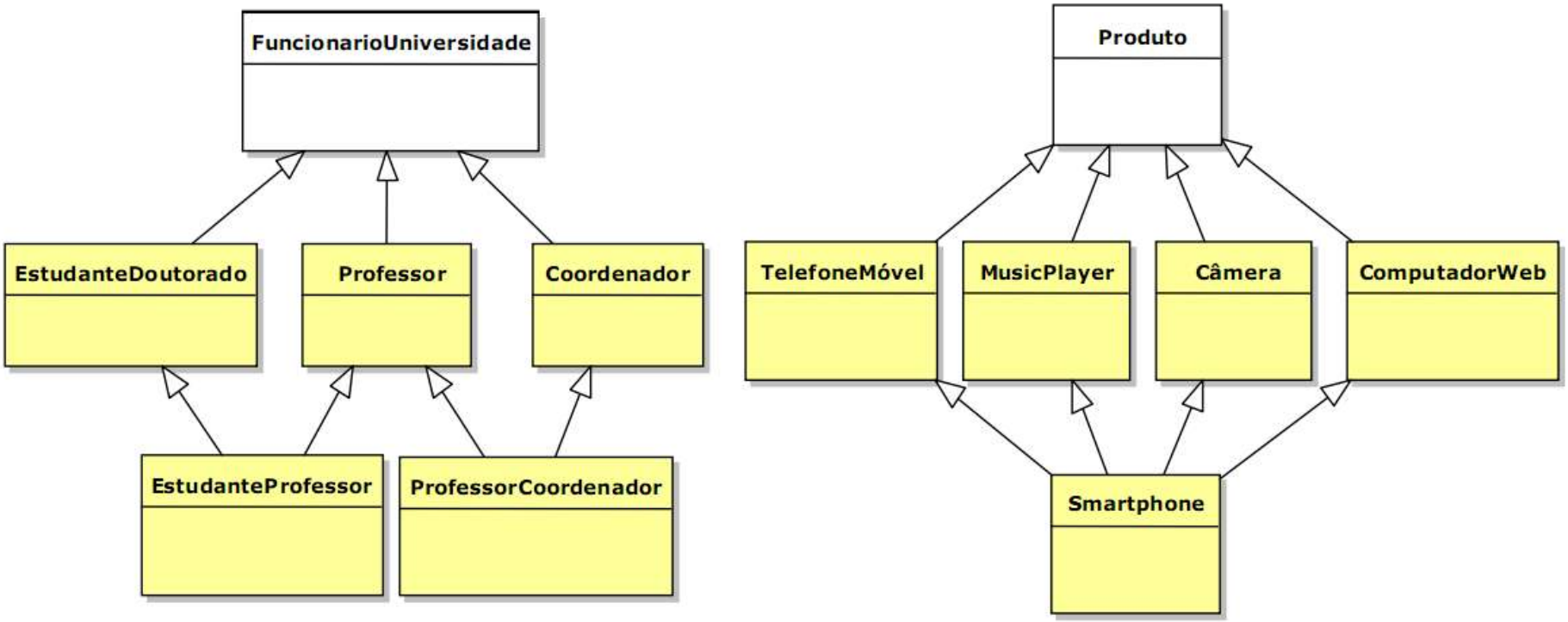
Herança

- Herança Simples:
  - uma subclasse é definida a partir de uma única superclasse.



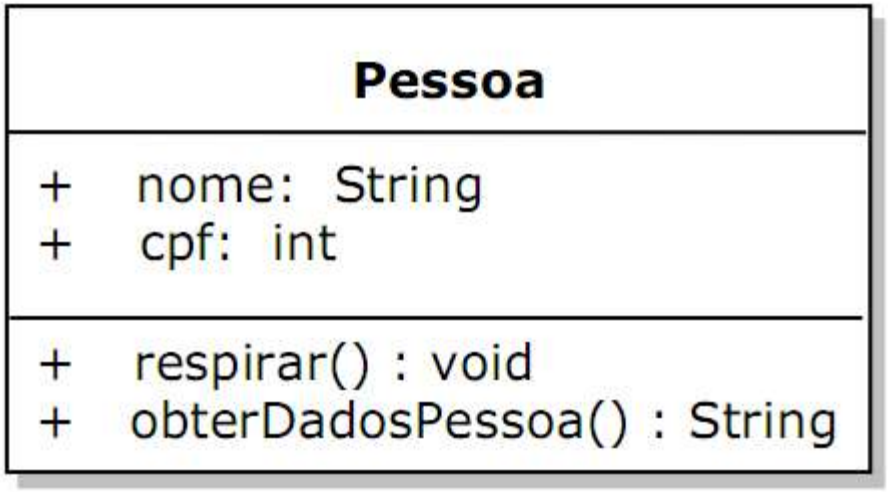
Herança

- Herança Múltipla:
  - Uma subclasse é definida a partir de muitas superclasses.



Herança

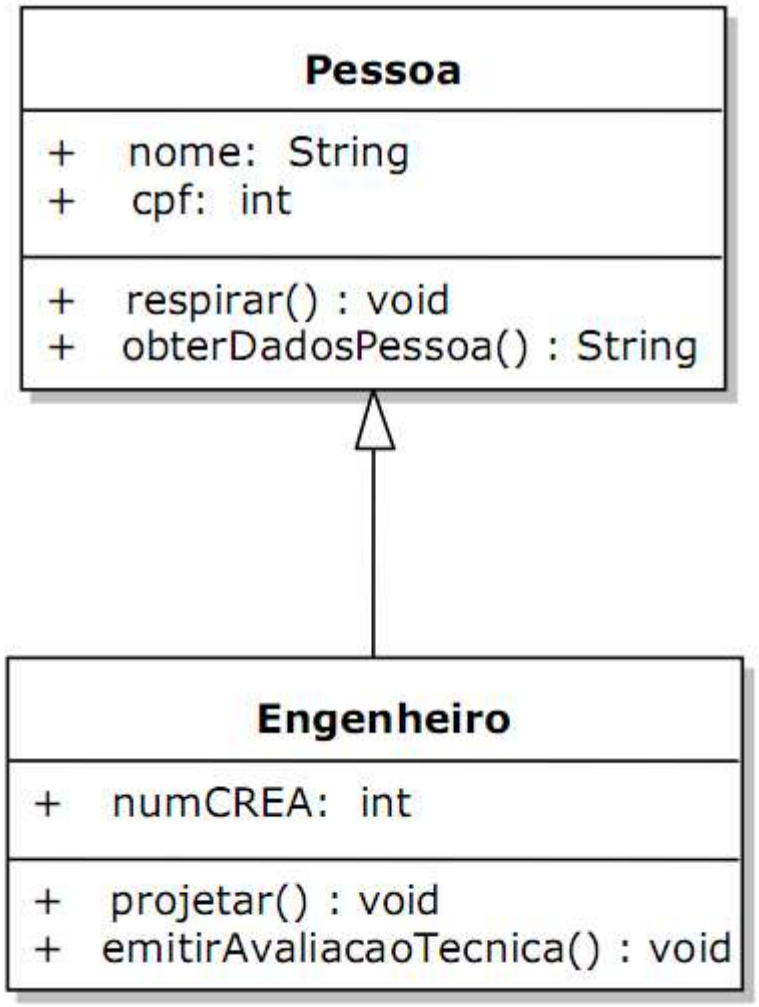
- Herança de Implementação:
  - a subclasse "herda" uma cópia da implementação da superclasse.
  - Pode ser combinada com herança simples ou múltipla.
- Exemplo: considere uma classe Pessoa.



Herança

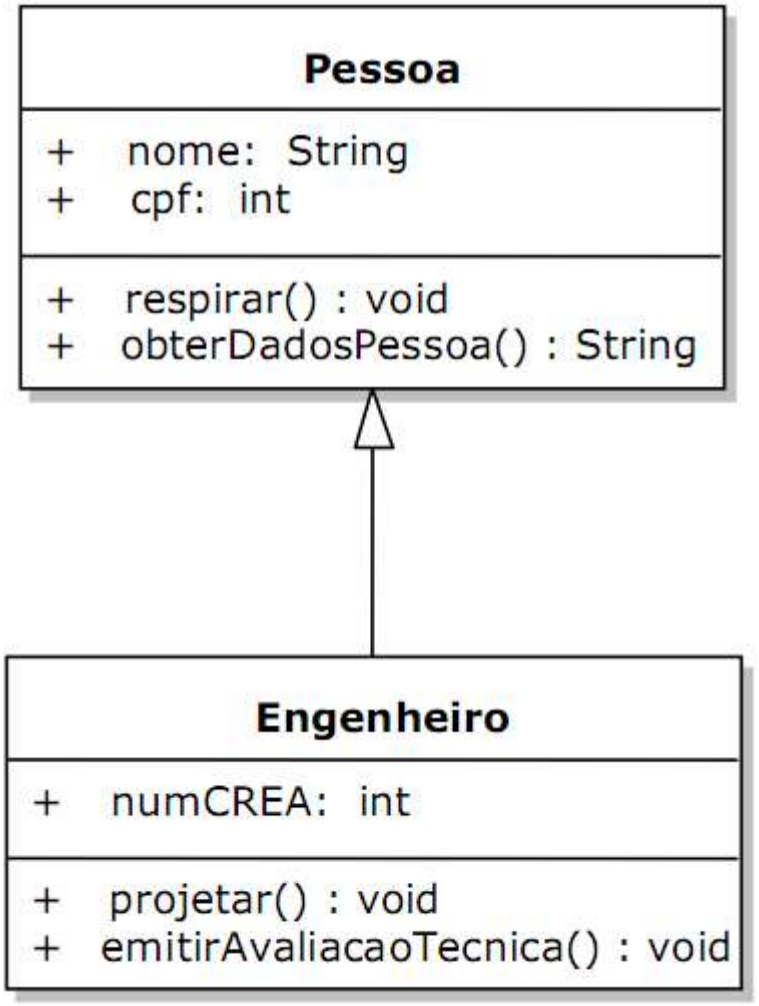
- Herança de Implementação:
- Vamos definir Engenheiro como subclasse de Pessoa.
  - Atributos de Engenheiro:
    - numCREA
    - nome e cpf (herdados de Pessoa)
  - Operações de Engenheiro:
    - projetar()
    - emitirAvaliacaoTecnica()
    - Respirar() e ObterDadosPessoa() (herdados de Pessoa)

Herança


Pessoa
+ nome: String
+ cpf: int
+ respirar() : void
+ obterDadosPessoa() : String
Engenheiro
+ numCREA: int
+ projetar() : void
+ emitirAvaliacaoTecnica() : void

Herança

- Os membros herdados não são visualizados na subclasse (seja no código ou no diagrama de classes UML).



Somente os membros novos

Herança

- Em Java usamos a construção `extends` para utilizar herança de implementação.

Herança

```
package Pessoas;

public class Pessoa {

    String nome;
    String cpf;

    public Pessoa (String pNome, String pCpf){
        this.nome = pNome;
        this.cpf = pCpf;
    }

    Public void respirar(){
        System.out.println("Respirando....!!!");
    }

    public String obterDadosPessoa(){
        return (this.nome + "  " + this.cpf);
    }
```

Herança

```
package Pessoas;

public class Engenheiro extends Pessoa{

   int numCREA;

public Engenheiro (String pNome, String pCpf, int pNumCREA){
   super(pNome,pCpf);
   this.numCREA = pNumCREA;
   System.out.println("Instanciou engenheiro: " + this.nome);
}
public void projetar(){
   System.out.println("Engenheiro: " + this.nome +
   "Projetando...");
}
public void emitirAvaliacaoTecnica(){
   System.out.println("Engenheiro: " + this.nome +"Emitindo
   Avaliação Técnica...");
}
}
```

Herança

- A palavra super é usada para referenciar membros da superclasse:
  - super.nome = "Pedro";
    - atribui "Pedro" para o nome em Pessoa.
  - super();
    - chama o construtor default da superclasse (desnecessário, pois esta chamada é implícita).
  - super(nome, cpf);
  - chama o construtor com parâmetros da superclasse Pessoa.

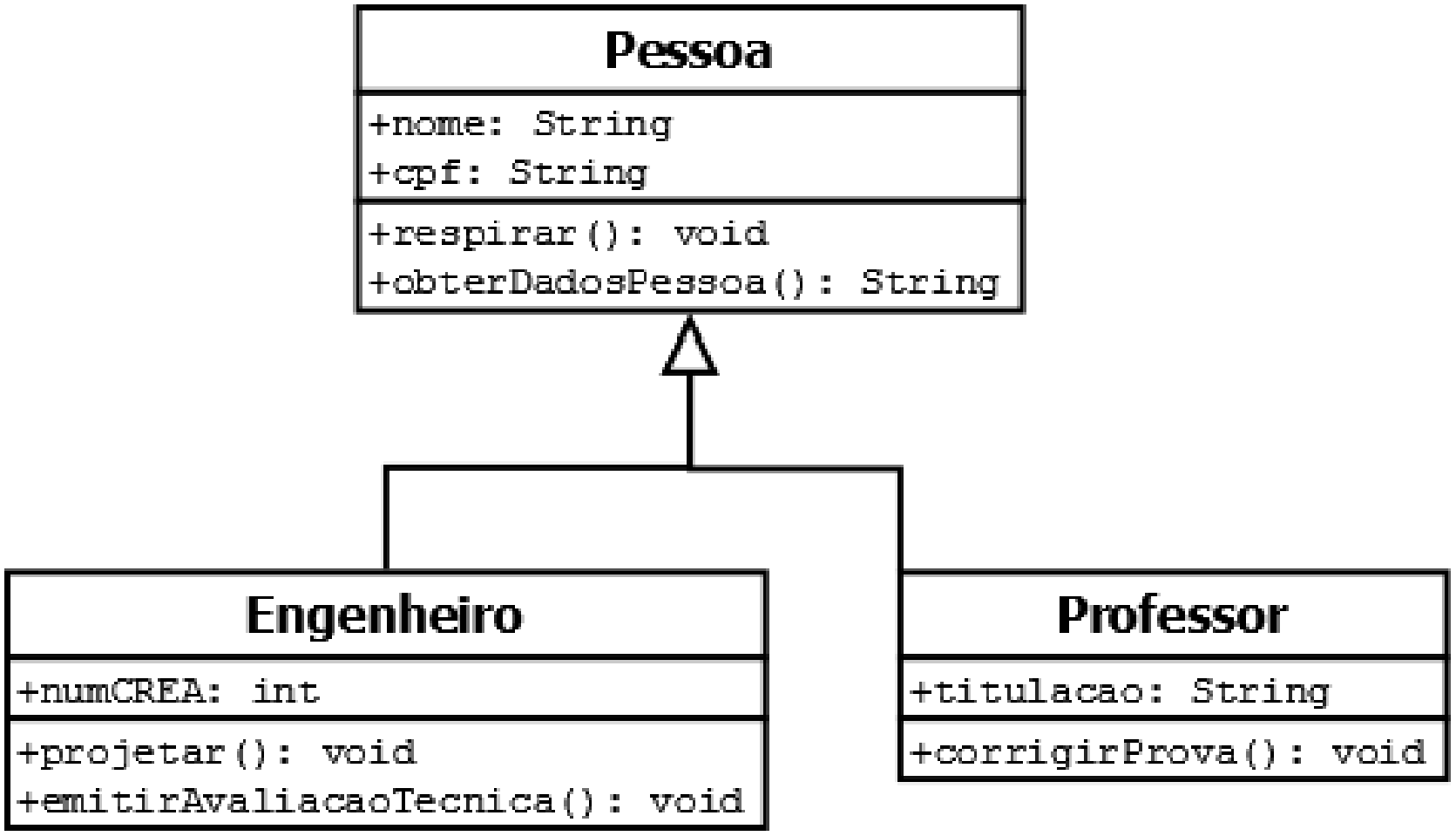
```
package Pessoas;
public class principal {
public static void main(String[] args) {

   Engenheiro a, b;
   a = new Engenheiro( "Oberon", "1234", 1111 );
   b = new Engenheiro( "Ada", "4321", 2222 );
   System.out.println( a.obterDadosPessoa() );
   a.respirar();
   System.out.println( b.obterDadosPessoa() );
   b.respirar();
   b.projetar();
   b.emitirAvaliacaoTecnica();

if ( a instanceof Engenheiro ){
   System.out.println(a.nome + " eh instancia de Engenheiro.");
} else {
   System.out.println("a NAO eh instancia de Engenheiro.");
}

if ( a instanceof Pessoa ){
   System.out.println(a.nome + " eh instancia de Pessoa.");
}else{
   System.out.println("a NAO eh instancia de Pessoa.");
}  } }
```

Herança

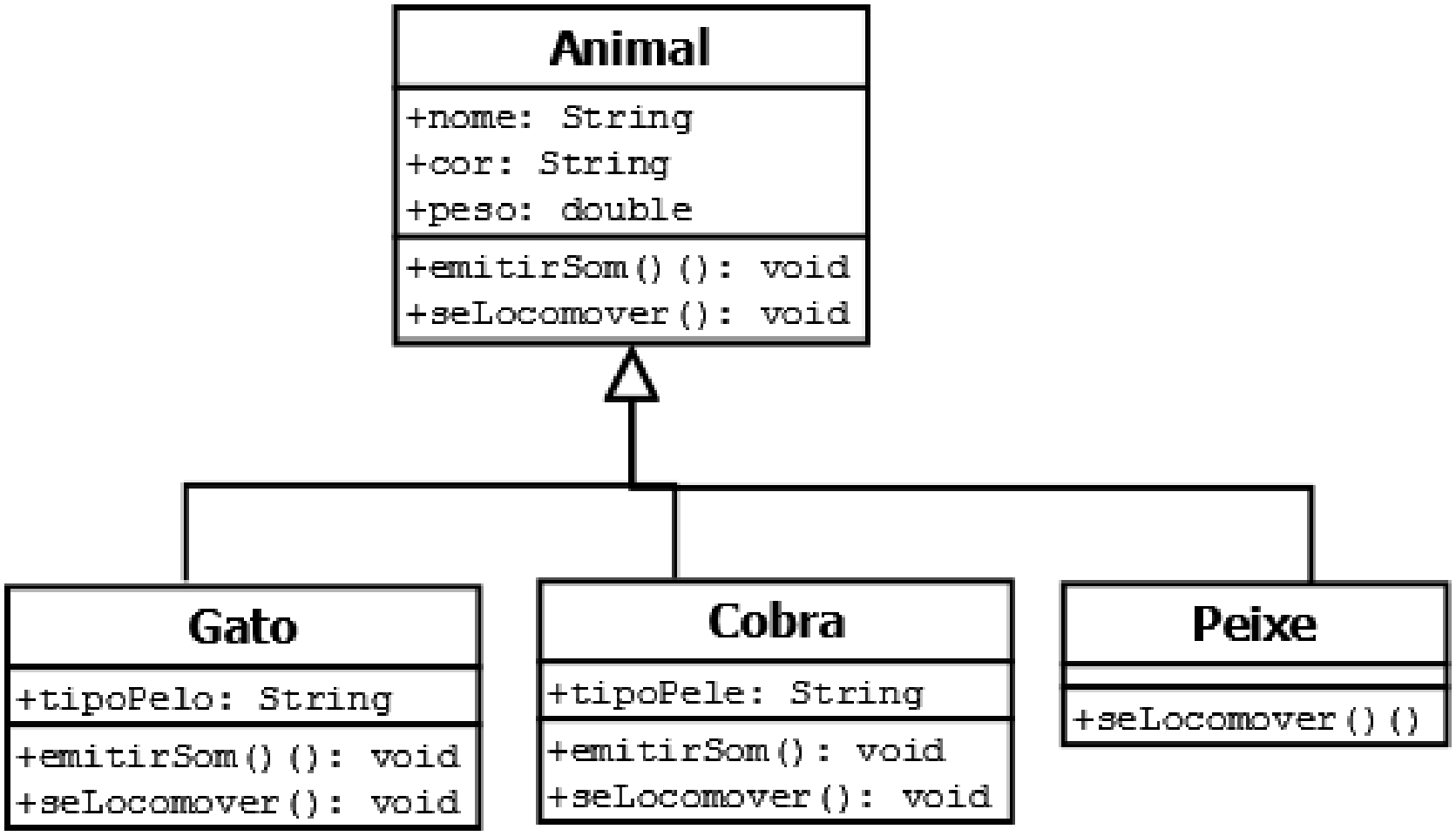

Pessoa
+nome: String
+cpf: String
+respirar(): void
+obterDadosPessoa(): String
Engenheiro
+numCREA: int
+projetar(): void
+emitirAvaliacaoTecnica(): void
Professor
+titulacao: String
+corrigirProva(): void

Sobrescrita (ou *override*)

- Tipo de Polimorfismo.
- Com a sobrescrita, conseguimos especializar os métodos herdados das superclasses, alterando o seu comportamento nas subclasses por um mais específico.
- Consiste em criar um novo método na classe filha contendo a mesma assinatura e mesmo tipo de retorno do método da classe pai.

Sobrescrita (ou *override*)

- O método deve possuir a mesma assinatura:
  - possuir o mesmo nome,
  - a mesma quantidade e
  - o mesmo tipo de parâmetros utilizado no método sobrescrito.


Animal
+nome: String
+cor: String
+peso: double
+emitirSom(): void
+seLocomover(): void
Gato
+tipoPelo: String
+emitirSom(): void
+seLocomover(): void
Cobra
+tipoPele: String
+emitirSom(): void
+seLocomover(): void
Peixe
+seLocomover()()

Sobrescrita

- Regras:
- Não modificar a qtde, ordem e tipo dos argumentos.
- O tipo de retorno igual respeitando a herança
- A visibilidade do método que sobrescreve não pode ser mais restritiva do que o método sobrescrito.