```php
<?php

    //HERANÇA:
    //OBS: O PHP não aceita herança múltipla (quando uma classe aceita a herança de mais de uma classe mãe), isso é feito justamente para evitar o problema diamante - problema diamente é quando 2 classes mãe possuem o mesmo nome para um método ou atributo herdado, gerando conflito de entendimento na linguagem sobre: "qual classe deve ser chamada para executar aquele atributo ou método".

    //Vamos usar como classe mãe a classe "Animal" e irá herdar seus atributos e métodos sobre as classes "Cachorro" e "Gato"...
    class Animal {

        private $nome;
        private $idade;

        public function __construct(string $nome, string $idade){
            $this->nome = $nome;
            $this->idade = $idade;
        }

        protected function getNome(){ //Perceba que deixamos os nossos métodos protegidos...
            return $this->nome;
        }

        protected function getIdade(){ //Perceba que deixamos os nossos métodos protegidos...
            return $this->idade;
        }

    }

    class Cachorro extends Animal {

        private $falar;

        public function __construct(string $nome, string $idade, string $falar){ //Na classe filha nós empregamos todos
                                                                                  //os atributos de ambas as classes
            parent:: __construct($nome, $idade);//Mas note que para referenciar     //mãe e filhas...
            $this->falar = $falar;              //os atributos da classe mãe nós
```

```php
                                        //usamos a palavra reservada "parent"
    }                                   //para referenciar os atributos da
                                        //classe mãe...
    public function getFalar():string
    {
        return $this->falar;
    }

    public function getDados():string
    {
        return $this->getNome() . PHP_EOL . $this->getIdade() . ' ano(s)'; //Note que métodos protegidos só podem
    }                                                                      //ser usados pelas classes filhas se
                                                                            //chamarmos o método dentro de outro
                                                                            //método e usando o "$this" para chamar
                                                                            //o método, existe outra forma de chamá-lo
                                                                            //usando o "parent", veja na classe "Gato"
}

class Gato extends Animal {

    private $falar;

    public function __construct(string $nome, string $idade, string $falar){

        parent:: __construct($nome, $idade);
        $this->falar = $falar;

    }

    public function getFalar():string
    {
        return $this->falar;
    }

    public function getDados():string
    {
```

```php
            return parent::getNome() . PHP_EOL . parent::getIdade() . ' ano(s)'; //Essa é outra forma de referenciar
        }                                                                          //métodos privados usando o "parent"

    }

    $pluto = new Cachorro('Pluto', '7', 'Au! Au!');
    echo $pluto->getDados();echo PHP_EOL;
    echo $pluto->getFalar();echo PHP_EOL;
    echo PHP_EOL;

    $gato_de_botas = new Gato('Gato de Botas', '5', 'Meow!');
    echo $gato_de_botas->getDados();echo PHP_EOL;
    echo $gato_de_botas->getFalar();echo PHP_EOL;
    echo PHP_EOL;

    var_dump($pluto);echo PHP_EOL;
    var_dump($gato_de_botas);

?>
```

```
[Running] php "c:\Users\Almoxarifado\Documents\php\arquivos_das_aulas\36-Heranca.php"
Pluto
7 ano(s)
Au! Au!

Gato de Botas
5 ano(s)
Meow!

object(Cachorro)#1 (3) {
  ["nome":"Animal":private]=>
  string(5) "Pluto"
  ["idade":"Animal":private]=>
```

```
  string(1) "7"
  ["falar":"Cachorro":private]=>
  string(7) "Au! Au!"
}

object(Gato)#2 (3) {
  ["nome":"Animal":private]=>
  string(13) "Gato de Botas"
  ["idade":"Animal":private]=>
  string(1) "5"
  ["falar":"Gato":private]=>
  string(5) "Meow!"
}

[Done] exited with code=0 in 0.147 seconds
```