Herança → conta bancária

profº Mauricio Conceição Mario

Herança

- Herança é uma propriedade originada a partir de uma ligação hierárquica entre classes, onde classes que estão em camadas inferiores dessa ligação hierárquica podem herdar atributos e comportamentos das classes que estão em hierarquia superior. O uso de Herança permite, portanto, o reaproveitamento de propriedades de uma classe.
- A classe que se encontra no topo da hierarquia é denominada de ***superclasse*** ou ***classe-mãe***, enquanto que as classes que estão na hierarquia inferior são chamadas de ***subclasses*** ou ***classes-filhas***.

(Dall'Oglio, 2014)

Exemplo: superclasse ou classe-mãe → Conta_Bancária.class.php.

Conta_Bancária.class.php

```php
<?php
 class Conta_Bancária
{
    protected $agencia;
    protected $conta;
    protected $saldo;

    public function __construct($agencia, $conta, $saldo)
    {
        $this->agencia = $agencia;
        $this->conta   = $conta;
        if ($saldo >= 0) {
            $this->saldo   = $saldo;
        }
    }
    public function getInfo()
    {
        return "Agência: {$this->agencia}, Conta: {$this->conta}";
    }
    public function depositar($quantia)
    {
        if ($quantia > 0) {
            $this->saldo += $quantia;
        }
    }

    public function getSaldo()
    {
        return $this->saldo;
    }
}
?>
```

(Dall'Oglio, 2014)

Subclasse ou classe-filha: Conta_Corrente.php.

Conta_Bancária.class.php | Consignado.php | Conta_Poupanca.php | aplica_herança2.php | Conta_Corrente.php

```php
<?php
class Conta_Corrente extends Conta_Bancária
{
    protected $limite;

    public function __construct($agencia, $conta, $saldo, $limite)
    {
        parent::__construct($agencia, $conta, $saldo);
        $this->limite = $limite;
    }

    public function retirar($quantia)
    {
        if ( ($this->saldo + $this->limite) >= $quantia ) {
            $this->saldo -= $quantia; // retirada permitida
        }
        else {
            return false; // retirada não permitida
        }
        return true;
    }
}
?>
```

parent::__construct (...) → acesso ao método construtor da superclasse

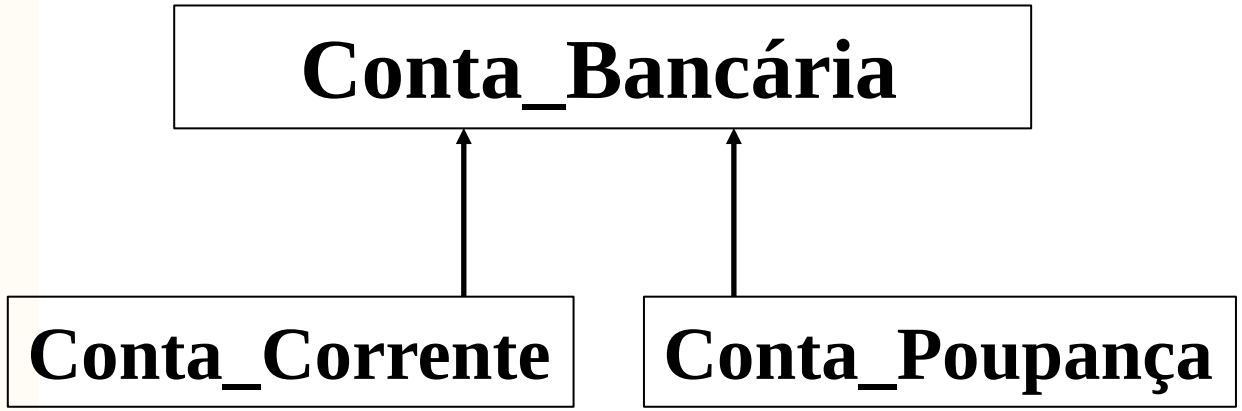
(Dall'Oglio, 2014)

Subclasse ou classe-filha: Conta_Poupança.php.

Conta_Bancária.class.php | Consignado.php | Conta_Poupanca.php | aplica_herança2.php

```php
<?php
 class Conta_Poupanca extends Conta_Bancária
{

    public function __construct($agencia, $conta, $saldo)
    {
        parent::__construct($agencia, $conta, $saldo);

    }

    function retirar($quantia)
    {
        if ($this->saldo >= $quantia) {
            $this->saldo -= $quantia;
        }
        else {
            return false; // retirada não permitida
        }
        return true; // retirada permitida
    }
}
?>
```

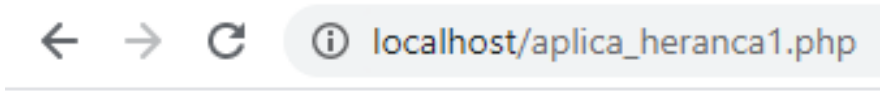
Diagrama de Classes → hierarquia:



(Dall'Oglio, 2014)

aplicação: aplica_heranca1.php.

```php
<?php
require_once 'Conta_Bancária.class.php';
require_once 'Conta_Poupanca.php';
require_once 'Conta_Corrente.php';

// criação dos objetos
$contas = array();
$contas[0] = new Conta_Corrente(6677, "CC.1234.56", 300, 400);
$contas[1] = new Conta_Poupanca(6678, "PP.1234.57", 100);

// percorre as contas
foreach ($contas as $key => $conta)
{
    print "Conta: {$conta->getInfo()} <br>\n";
    print "    Saldo atual: {$conta->getSaldo()} <br>\n";
    $conta->depositar(200);
    print "    Depósito de: 200 <br>\n";
    print "    Saldo atual: {$conta->getSaldo()} <br>\n";

    if ($conta->retirar(700)) {
        print "    Retirada de: 700 <br>\n";
    }
    else
    {
        print "    Retirada de: 700 (não permitida)<br>\n";
    }
    print "    Saldo atual: {$conta->getSaldo()} <br>\n";
}

?>
```



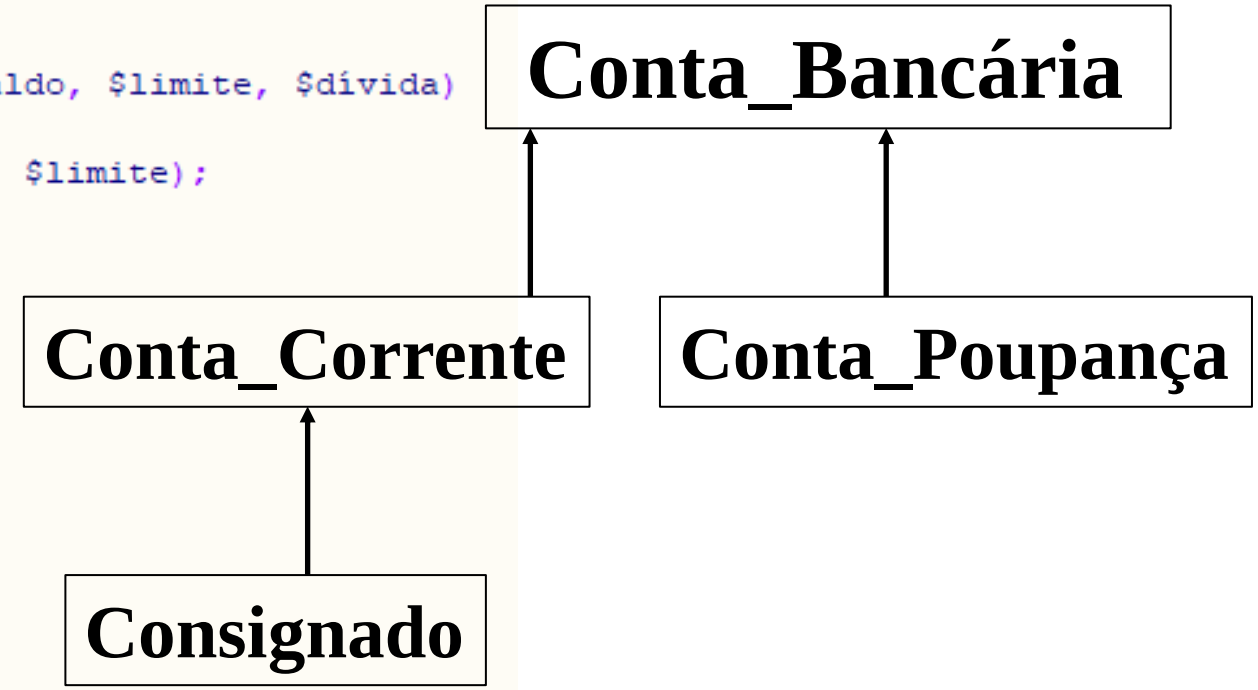
Conta: Agência: 6677, Conta: CC.1234.56
Saldo atual: 300
Depósito de: 200
Saldo atual: 500
Retirada de: 700
Saldo atual: -200
Conta: Agência: 6678, Conta: PP.1234.57
Saldo atual: 100
Depósito de: 200
Saldo atual: 300
Retirada de: 700 (não permitida)
Saldo atual: 300

Subclasse ou classe-filha: Consignado.php.

Conta.class.php | Consignado.php | Conta_Poupanca.php | aplica_herança2.php | Conta_Corrente.php | apli

```php
<?php
class Consignado extends Conta_Corrente
{
    protected $empréstimo;
    protected $dívida;

    public function __construct($agencia, $conta, $saldo, $limite, $dívida)
    {
        parent::__construct($agencia, $conta, $saldo, $limite);
        $this->dívida = $dívida;
    }

    public function set_empréstimo($empréstimo)
    {
        $this->empréstimo = $empréstimo;
    }

    public function get_empréstimo( )
    {
       return $this->empréstimo;
    }

    public function get_dívida( )
    {
       return $this->dívida;
    }
}
?>
```

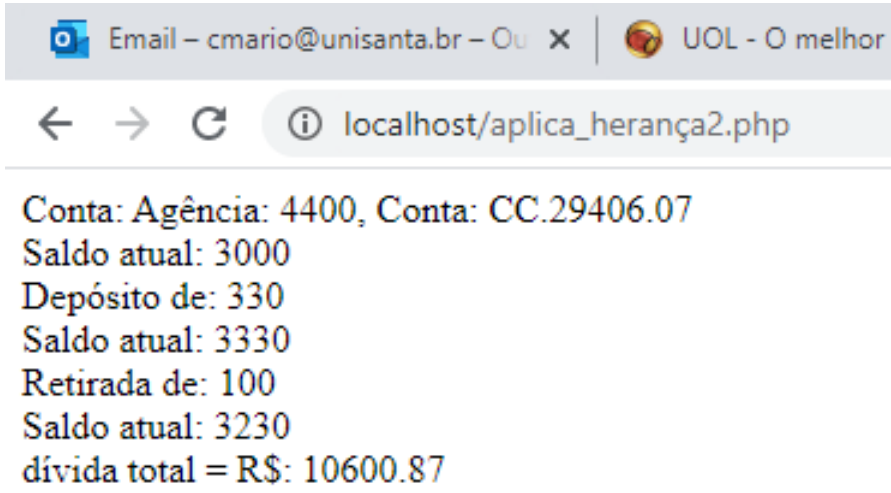
Diagrama de Classes → hierarquia:



aplicação: aplica_herança2.php.

Conta_Bancária.class.php | Consignado.php | Conta_Poupanca.php | aplica_herança2.php | Conta_Corren

```php
<?php
require_once 'Conta_Bancária.class.php';
require_once 'Conta_Poupanca.php';
require_once 'Conta_Corrente.php';
require_once 'Consignado.php';
// criação de objeto do tipo Consignado
$consig = new Consignado (4400, "CC.29406.07", 3000, 8800, 10000);

    print "Conta: {$consig->getInfo()} <br>\n";
    print "    Saldo atual: {$consig->getSaldo()} <br>\n";
    $consig->depositar(330);
    print "    Depósito de: 330 <br>\n";
    print "    Saldo atual: {$consig->getSaldo()} <br>\n";

    $consig->retirar(100);
        print "    Retirada de: 100 <br>\n";
        print "    Saldo atual: {$consig->getSaldo()} <br>\n";

    $consig->set_empréstimo(600.87);
    $dívida_total = ($consig->get_empréstimo()+ $consig->get_dívida());
    print " dívida total = R$: {$dívida_total} <br>\n";

?>
```



Referências Bibliograficas

- *php* Programando com Orientação a Objetos
  Pablo Dall'Oglio - editora Novatec - 2014

- Desenvolvimento web com PHP e MySQL
  Evaldo Junior Bento – editora Casa do Código - 2018