

Programação Orientada a Objetos com PHP

Classes Abstratas e Polimorfismo

profº Mauricio Conceição Mario

Classes Abstratas

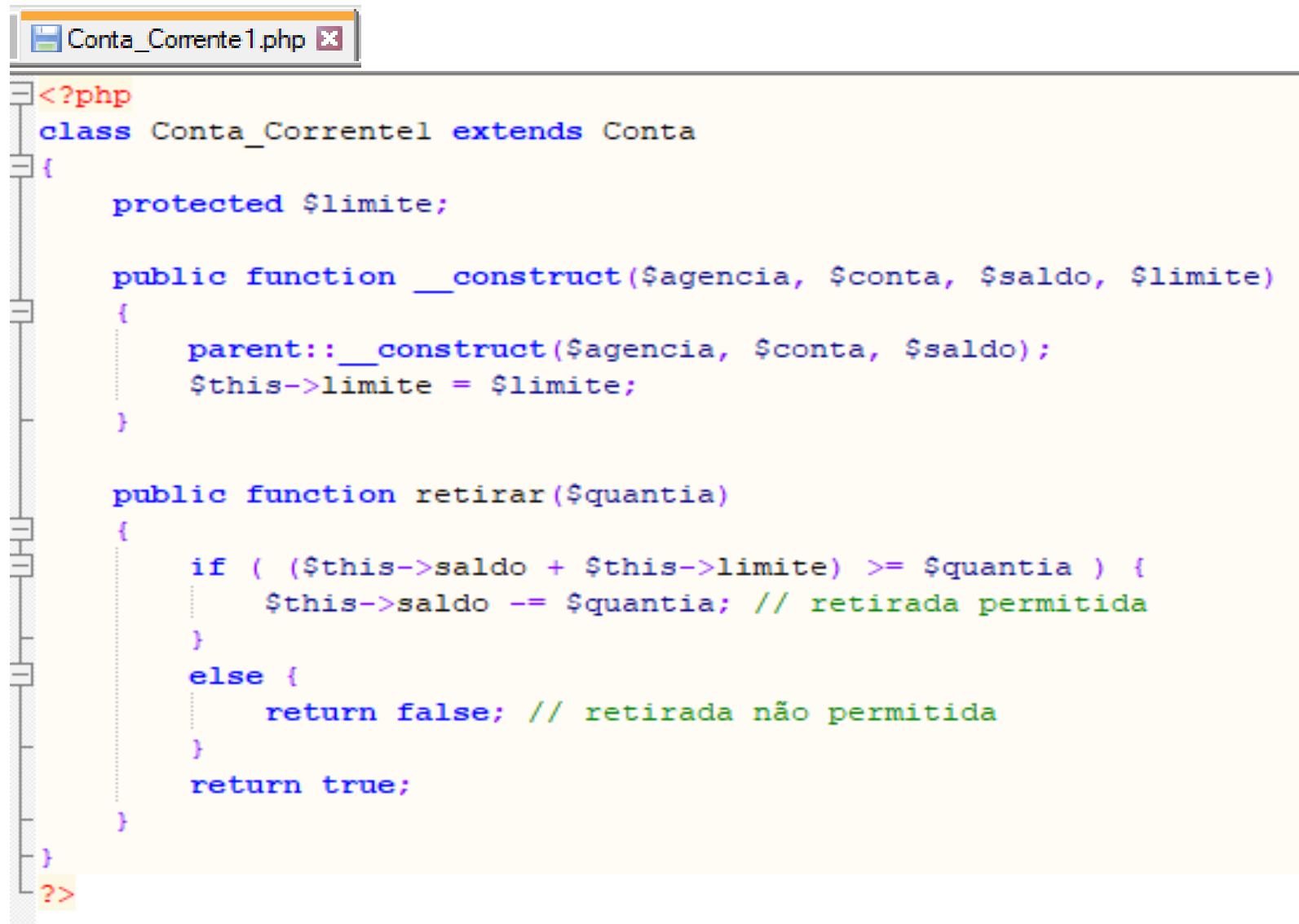
- São classes que dentro de uma hierarquia de classes servem para representar um modelo ou conceito da aplicação.
- Classes Abstratas não podem ser instanciadas diretamente na forma de objetos, estas instâncias devem ser feitas através de suas subclasses.

Exemplo: classe abstrata → Conta.class.php.

```
1 <?php
2 abstract class Conta
3 {
4     protected $agencia;
5     protected $conta;
6     protected $saldo;
7
8     public function __construct($agencia, $conta, $saldo)
9     {
10         $this->agencia = $agencia;
11         $this->conta    = $conta;
12         if ($saldo >= 0) {
13             $this->saldo = $saldo;
14         }
15     }
16     public function getInfo()
17     {
18         return "Agência: {$this->agencia}, Conta: {$this->conta}";
19     }
20     public function depositar($quantia)
21     {
22         if ($quantia > 0) {
23             $this->saldo += $quantia;
24         }
25     }
26
27     public function getSaldo()
28     {
29         return $this->saldo;
30     }
31
32     abstract function retirar($quantia);
33 }
34 ?>
```

/*método abstrato: não é implementado dentro da classe, mas deve ser implementado pelas subclasses da classe abstrata*/

Subclasse ou classe-filha: **Conta_Corrente1.php.**



```
<?php
class Conta_Corrente1 extends Conta
{
    protected $limite;

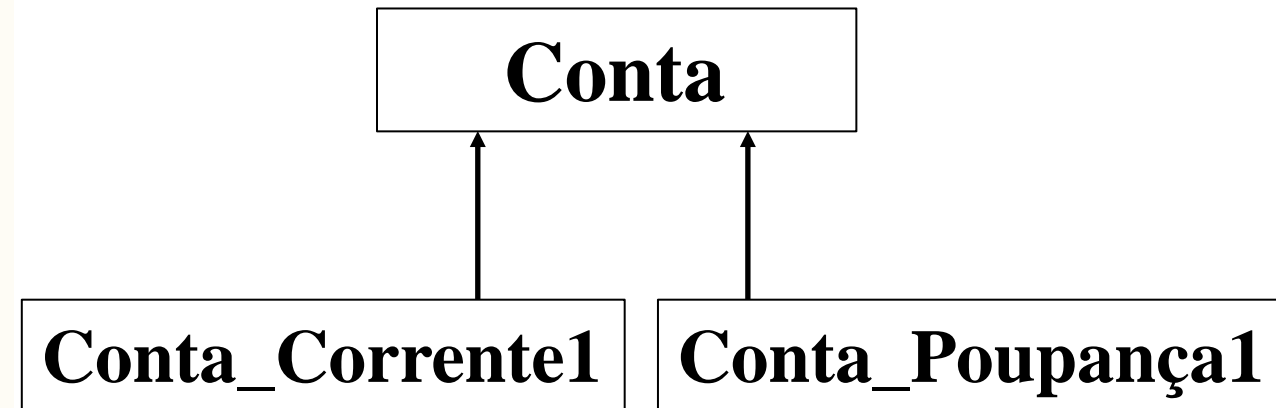
    public function __construct($agencia, $conta, $saldo, $limite)
    {
        parent::__construct($agencia, $conta, $saldo);
        $this->limite = $limite;
    }

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

Subclasse ou classe-filha: **Conta_Poupança1.php.**

```
a_Bancária.class.php x Consignado.php x Conta_Poupanca1.php x aplica_herança2.  
<?php  
class Conta_Poupanca1 extends Conta  
{  
  
    public function __construct($agencia, $conta, $saldo)  
    {  
        parent::__construct($agencia, $conta, $saldo);  
    }  
  
    function retirar($quantia)  
    {  
        $this->saldo = (100.00 + $quantia);  
    }  
}
```

Diagrama de Classes → hierarquia:



aplicação: aplica_herança3.php.

```
<?php
require_once 'Conta.class.php';
require_once 'Conta_Poupanca1.php';      /*não pode ser feita uma
require_once 'Conta_Correntel.php';      instância à classe Conta*/

// criação de objetos tipo Conta_Corrente e Conta_Poupança
$cc = new Conta_Correntel(121268, "CC.33868.88", 1000, 800);
$cp = new Conta_Poupanca1(15620, "CC.8251.513", 600.88);

print "Conta Corrente: {$cc->getInfo()} <br>\n";
print "    Saldo atual: {$cc->getSaldo()} <br>\n";
$cc->depositar(-500.90);
print "    Depósito de: -500.90 <br>\n";
print "    Saldo atual: {$cc->getSaldo()} <br>\n";
$cc->retirar(80.66);
print "    Retirada de: 80.66 <br>\n";
print "    Saldo atual: {$cc->getSaldo()} <br>\n";

print "<br>\n Conta Poupança: {$cp->getInfo()} <br>\n";
print "    Saldo atual: {$cp->getSaldo()} <br>\n";
$cp->depositar(70.50);
print "    Depósito de: 70.50 <br>\n";
print "    Saldo atual: {$cp->getSaldo()} <br>\n";
$cp->retirar(40.36);
print "    Retirada de: 40.36 <br>\n";
print "    Saldo atual: {$cp->getSaldo()} <br>\n";
```

?>

aplica_herança3.php



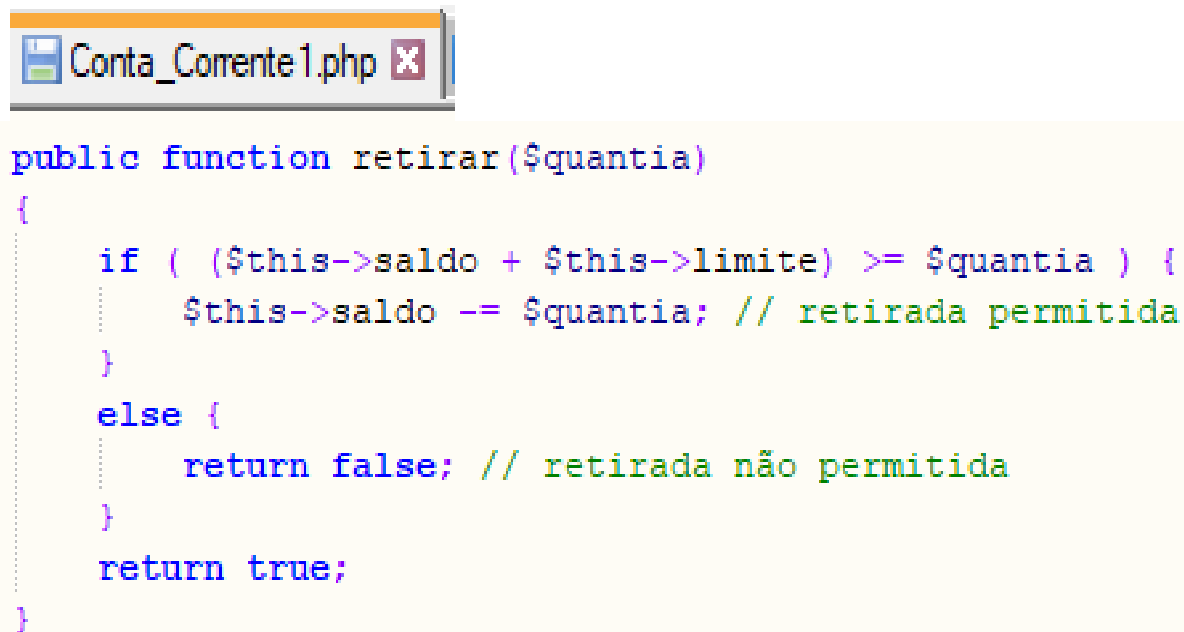
localhost/aplica_herança3.php

Conta Corrente: Agência: 121268, Conta: CC.33868.88
Saldo atual: 1000
Depósito de: -500.90
Saldo atual: 1000
Retirada de: 80.66
Saldo atual: 919.34

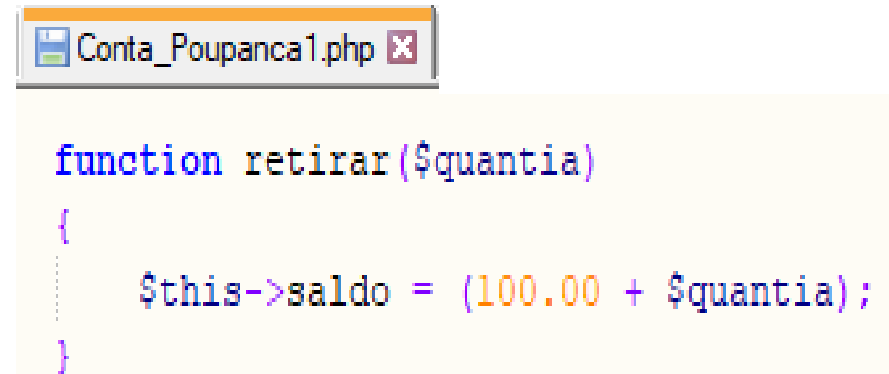
Conta Poupança: Agência: 15620, Conta: CC.8251.513
Saldo atual: 600.88
Depósito de: 70.50
Saldo atual: 671.38
Retirada de: 40.36
Saldo atual: 140.36

Polimorfismo \equiv muitas formas.

- Em Orientação a Objetos é o princípio que permite a classes derivadas de uma mesma superclasse ter métodos iguais (mesma identidade e mesmos parâmetros), mas comportamentos (funcionalidades) diferentes, redefinidos em cada uma das classes-filha. **Exemplo: método** retirar(...) nas classes abaixo.



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



```
function retirar($quantia)
{
    $this->saldo = (100.00 + $quantia);
}
```

Exercícios propostos:

14. Criar uma classe abstrata que represente o conceito de um escritório de contabilidade, com ao menos um método abstrato. Criar uma classe filha da abstração, que utilize suas propriedades através da aplicação “.php” que mostrará os dados no navegador.
15. Utilizando o conceito de polimorfismo, criar uma classe que na hierarquia de herança esteja em um nível inferior da classe filha pedida no exercício anterior, que implemente versões diferentes de métodos desta classe filha. Incluir o seu uso através da aplicação “.php” que mostrará os dados no navegador.