<?php

    //INTERFACES:

    //Como o PHP é uma linguagem que não suporta o uso de classes múltiplas numa herança, a forma que existe de fazer com que o PHP force uma herança múltipla é criando um tipo diferente de classe chamado interface, veja a implementação dela logo abaixo.

    //Aqui temos a nossa classe abstrata que gera um usuário...

    abstract class Usuario {

        protected function \_\_construct (string $nome, string $senha){

            $this->nome = $nome;

            $this->senha = $senha;

        }

        abstract protected function mostraUsuario();

    }

    //Aqui temos a interface que deverá ser compatilhada entre as classes filhas junto com a classe abstrata.

    interface Autenticacao { //Note que invés de class usamos a palavra reservada "interface", nossas interfaces só

                                //poderão conter métodos sem corpo.

        public function autenticar(string $senha):bool; //Aqui temos um método sem corpo, que deve retornar um boolean.

                                                        //Os métodos nas interfaces são criados sem corpo para que

    }                                                   //possam ser implementados nas classes filhas segundo o desejo

                                                        //do desenvolvedor.

    //Aqui temos um Gerente que irá herdar de Usuario e terá os métodos da interface "Autenticacao" implementados nele...

    class Gerente extends Usuario implements Autenticacao { //Veja que para usar os métodos da interface usamos a

                                                            //palavra reservada "implements", e podemos implementar

                                                            //quantas interfaces quisermos, basta separá-las pelas

                                                            //vírgulas.

        public function \_\_construct (string $nome, string $senha){

            parent:: \_\_construct($nome, $senha);

        }

        public function autenticar(string $senha):bool //Perceba que implementamos o método abstrato da interface

        {                                               //Autenticacao...

            return $senha === '4321'; //Esse método vai comparar a senha passada com a senha '4321'...

        }

        public function mostraUsuario(){ //E temos um método que só mostra o nome de usuário se a senha correta for

            if($this->autenticar($this->senha) == TRUE){ //passada...

                echo "Nome: $this->nome" . PHP\_EOL . PHP\_EOL;

            } else

            echo "Senha inválida" . PHP\_EOL . PHP\_EOL;

        }

    }

    class Funcionario extends Usuario {

        public function \_\_construct (string $nome, string $senha){

            parent:: \_\_construct($nome, $senha);

        }

        public function autenticar(string $senha):bool

        {

            return $senha === '1234';

        }

        public function mostraUsuario(){

            if($this->autenticar($this->senha) == TRUE){

                echo "Nome: $this->nome" . PHP\_EOL . PHP\_EOL;

            } else

            echo "Senha inválida" . PHP\_EOL . PHP\_EOL;

        }

    }

    $gabriel = new Funcionario('Gabriel', '1234');

    $gabriel->mostraUsuario(); //Perceba que a função implementada mostra o nome do funcionário logo de cara por que

                                //passamos a senha correta...

    $marcelo = new Gerente('Marcelo', '1234'); //Mas veja que a senha do gerente nós passamos errado...

    $marcelo->mostraUsuario(); //Quando tentarmos ver o nome de usuário não vamos conseguir...

    $marcelo->senha = '4321'; //Mas depois de mudar a senha para a senha correta nós conseguimos visualizar o nome...

    $marcelo->mostraUsuario();

?>

[Running] php "c:\Users\Almoxarifado\Documents\php\arquivos\_das\_aulas\40-Interfaces.php"

Nome: Gabriel

Senha inválida

Nome: Marcelo

[Done] exited with code=0 in 0.117 seconds