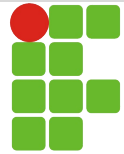
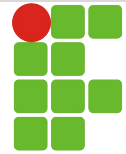


Programação orientada a objetos

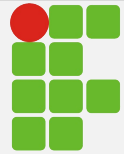
Maximilian Jaderson de Melo

Aula 6



Introdução a Orientação a objetos

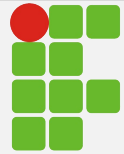
Conteúdo de hoje



- Conceitos básicos de orientação a objetos.
  - Aqui começa o pesadelo!

Pilares da orientação a objetos

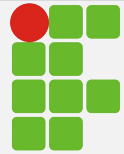
Abordagem procedural



- Imagine uma aplicação para validar o cpf de uma pessoa.

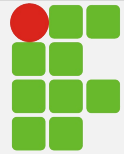
```
//assumindo leitura por Scanner
String cpf = input.next();

//deve retornar booleano
validar_cpf(cpf);
```

Abordagem orientada a objetos



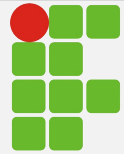
- Uma aplicação teria muitas variáveis desorganizadas, espalhadas.
- Uma possível melhoria na organização?
  - Structs.

Abstração



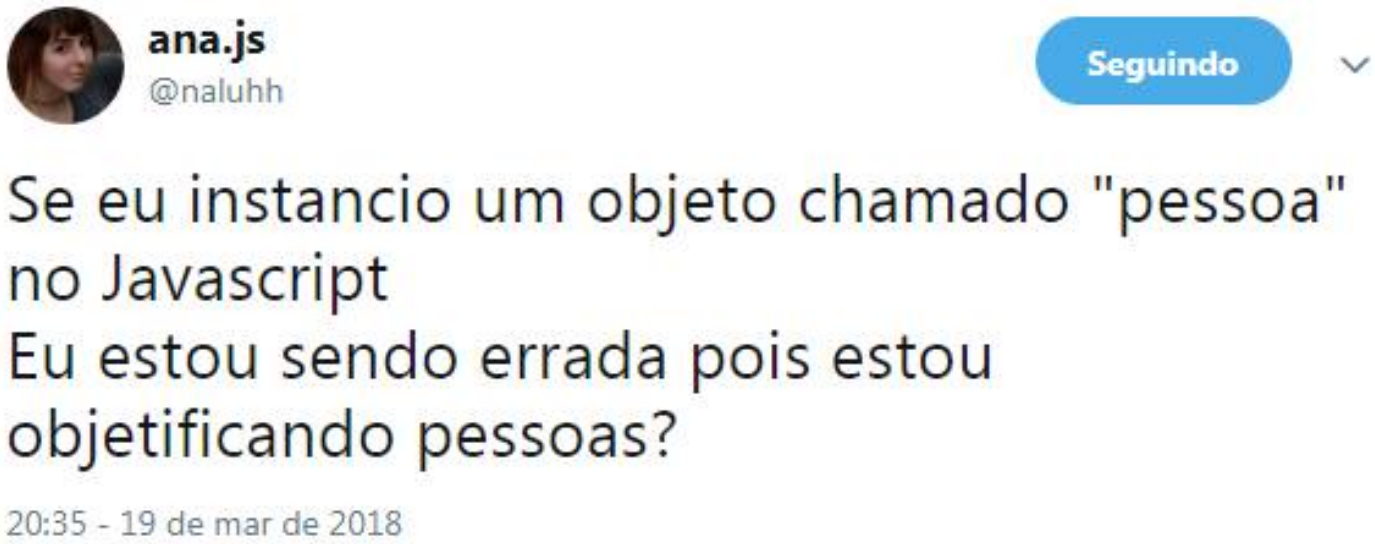
- O que é abstração?
  - É o processo de observar e descrever as entidades do mundo real.
- É muito mais fácil pensar nas entidades do mundo real, do que em variáveis isoladas.
- Aproximação da realidade, ideia de objetos.

Abstração: Tipos



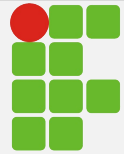
- Todas as coisas são entendidas como objetos.
- Um tipo (Classe) é o agrupamento lógico de elementos com semântica em comum.
- A resolução de problemas dar-se-á por meio da interação destes objetos, por meio de troca de mensagens.

Abstração: Tipos



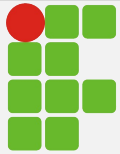
Abstração: Tipos

Tipos, atributos e métodos



- Pense em uma pessoa. De que ela é composta?
  - Quais as características presentes em qualquer pessoa?
  - Quais as ações que qualquer pessoa pratica?

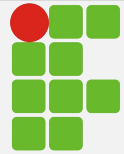
Exercício prático



1. Assim como nas entidades de Banco de dados, descreve em termos de características e acões praticáveis:

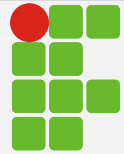
- Uma escola.
- Uma turma.
- Um aluno.
- Um professor.

Tipos, atributos e métodos



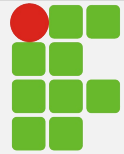
```
class Pessoa{

  String nome;
  String rg;
  Date dataNascimento;

  void falar(){
      //faz alguma coisa
  }
}
```

Convenções



- Para classes
  - A primeira letra sempre é **MAIÚSCULA**.
- Para atributos
  - A primeira letra sempre é **minúscula**.
- Para ambos
  - As palavras **não** são mais separadas por _.
  - São sempre capitalizadas.
  - Nome de classe: PessoaFisica.
  - Nome de atributo: dataNascimento.

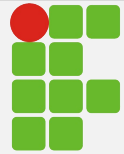
Convenções



- Para constantes
  - A palavra é MAIÚSCULA.
  - Cada palavra é separada por _.
- Exemplos

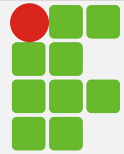
```
class Pessoa{
      final int NUMERO_MAXIMO = 10;
}
```

Classes e objetos



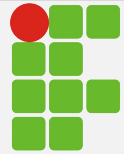
- Cada objeto no mundo real é único.
- Cada objeto possui algum atributo que o torna único.
- Na vida real, cada uma das pessoas é diferente, mesmo que eventualmente possuam atributos em comum, como nome, por exemplo.

Classes e objetos



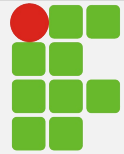
- A classe é uma abstração que permite descrever as propriedades de um elemento da vida real.
- Atributos = variáveis.
- Métodos = funções.
- Comportamento = conjunto dos métodos de uma classe.

Classes e objetos



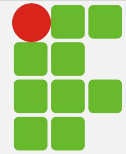
- Cada objeto possui algum atributo que o torna único.
- Cada objeto no mundo real é único.
- Na vida real, cada uma das pessoas é diferente, mesmo que eventualmente possuam atributos em comum, como nome, por exemplo.
- O objeto é a instância, ou cada representação “viva” de um elemento de certa classe.

Classes e objetos



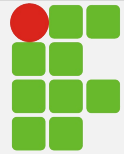
```
class Pessoa{

  String nome;
  String rg;
  Date dataNascimento;

  void falar(){
      //faz alguma coisa
  }

  //continua
```

Classes e objetos



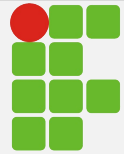
```
    //continuação

    public static void main(String args[]){
      //objeto(instância) do tipo Pessoa
      Pessoa p = new Pessoa();
    }
}
```

Exercício



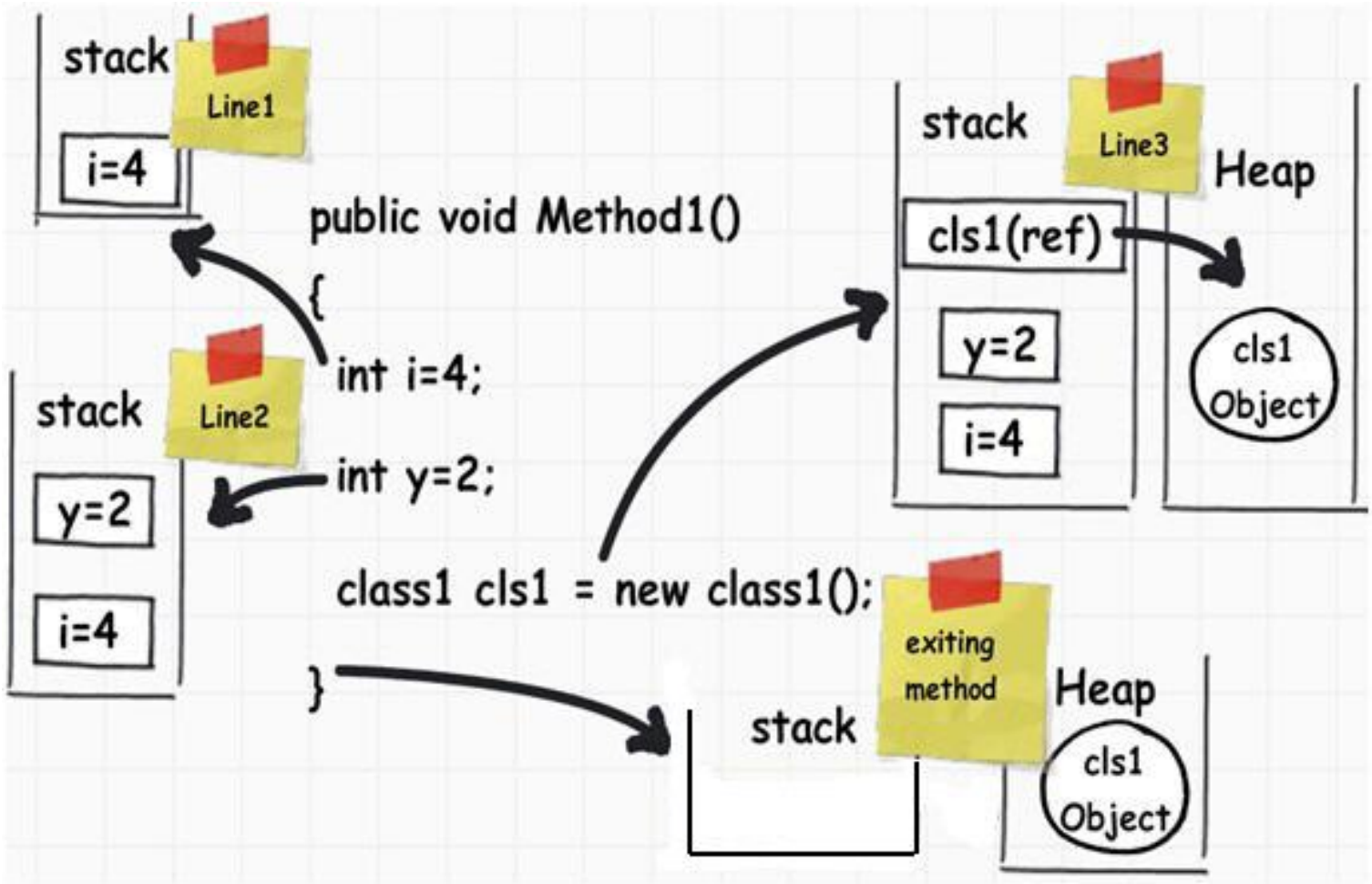
- Crie classes para representar os seguintes elementos do mundo real (não é necessário especificar a implementação dos métodos):
  - Uma escola.
  - Uma turma.
  - Um aluno.
  - Um professor.

Instâncias

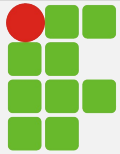


- Dentro da main, há um operador novo, o **new**.
- Na frente dele, há sempre o nome de uma classe. Por quê?
  - *malloc* apenas?

Instâncias

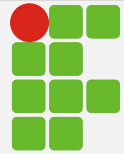

stack
Line1
i=4
public void Method1()
{
int i=4;
stack
Line2
y=2
i=4
int y=2;
stack
Line3
Heap
cls1(ref)
y=2
i=4
cls1
Object
class1 cls1 = new class1();
}
exiting
method
stack
Heap
cls1
Object

Construtores



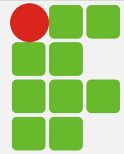
- Como o próprio nome já diz, auxilia na construção de um objeto.
- Sua invocação é feita na criação do objeto (*new*).
- Seu papel é inicializar cada um dos atributos do objeto.
- É um método sem tipo de retorno, com o mesmo nome da classe.
- A frente do *new* **não** está o nome da classe, mas sim **qual construtor** será invocado!

Construtores



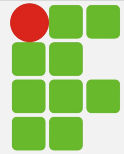
```
class Pessoa{

  String nome;
  String rg;
  Date dataNascimento;

  //continua
```

Construtores



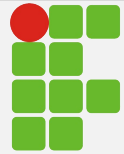
```
//continuação
//construtor sem parâmetros
public Pessoa(){
    nome = "";
    rg   = "";
    dataNascimento = new Date();
}


//continua
```

Construtores



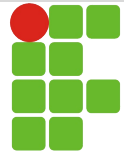
```
//continuação

//construtor inicializando um nome
public Pessoa( String n ){
    nome = n;
    rg   = "";
    dataNascimento = new Date();
}

//continua
```

Construtores



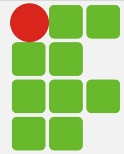
- Quais construtores estão sendo invocados abaixo?

```
//continuação

public static void main(String args[]){
    Pessoa p  = new Pessoa();
    Pessoa p2 = new Pessoa("max");
}
```



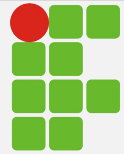
Classes em PHP

Classes em PHP



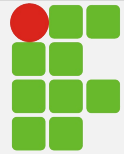
- Em PHP, não há a obrigatoriedade do nome da classe combinar com o nome do arquivo.
  - Em Java - Classe: Pessoa, arquivo Pessoa.java
- Para projetos em PHP podemos usar o netbeans.
  - O projeto deve ficar salvo na pasta **htdocs** do Xampp.

Classes em PHP



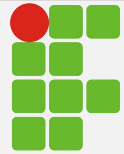
- Em PHP, os tipos são definidos no momento da utilização.
- Variáveis em PHP são definidas com uso do $.
- Funções em PHP são definidas com uso da palavra reservada *function*.

Classes em PHP



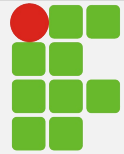
```php
<?php
    $var = "nome";

    function sayHello(){
      echo "hello world!";
    }
?>
```

Classes em PHP



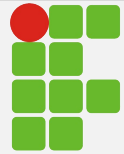
- As convenções para criação de classes são as mesmas do Java (ou será que valem para qualquer linguagem que suporte OO?).
- O acesso à elementos de classe não é mais feito pelo símbolo “.”, mas sim por “->”

Classes em PHP



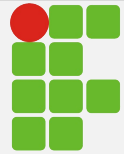
```php
<?php
  class Pessoa{
    public $nome;
    public $rg;

    function falar(){
      print("olá, meu nome é ".$this->nome);
    }
  }//fim da classe
?>
```

Classes em PHP



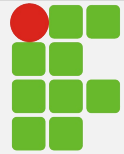
```php
    //continuação
    $p = new Pessoa();

    echo $p->nome."<br/>";
    echo $p->falar();
?>
```

Classes em PHP



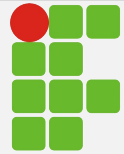
- Em PHP não há a necessidade de criar a main.
  - Qualquer coisa fora da definição da classe será executado em ordem linear.
- O atributo nome foi impresso?
  - Nele tem algum lixo de memória?
  - O PHP implementa construtores?

Construtores em PHP



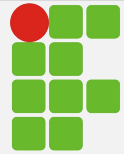
- Método ___construct .
- Mesmo funcionamento visto em Java.
  - Entretanto só podemos criar **um** construtor em PHP!
  - Veremos como contornar essa limitação mais a frente!

Construtores em PHP



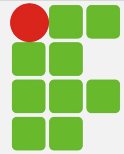
```php
<?php
  class Pessoa{
    public $nome;
    public $rg;
    public function __construct(){
      $this->nome = "max";
      $this->rg   = "1234";
    }
    public function falar(){
      print("olá, meu nome é ".$this->nome);
    }
  }//fim da classe
```

Construtores em PHP



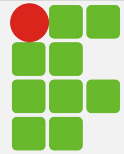
```php
    //continuação
    $p = new Pessoa();

    echo $p->nome."<br/>";
    echo $p->falar();
?>
```

Exercício



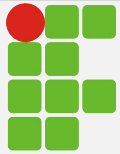
1. Identifique em cada uma das classes abaixo os atributos, métodos, objetos e classes. Em seguida, crie um construtor padrão e outro que receba cada um dos atributos da classe (Java e PHP).
    - Uma escola.
    - Uma turma.
    - Um aluno.
    - Um professor.

Próxima aula



- Encapsulamento.

Dúvidas, críticas ou sugestões?



maximilian.melo@ifms.edu.br
max.mjm.melo@gmail.com