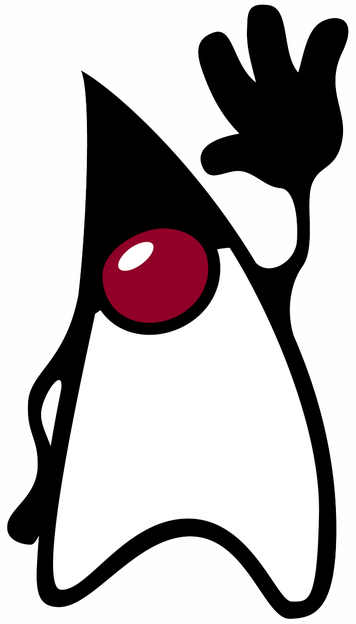
UNIVERSIDADE
FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Classes e Objetos

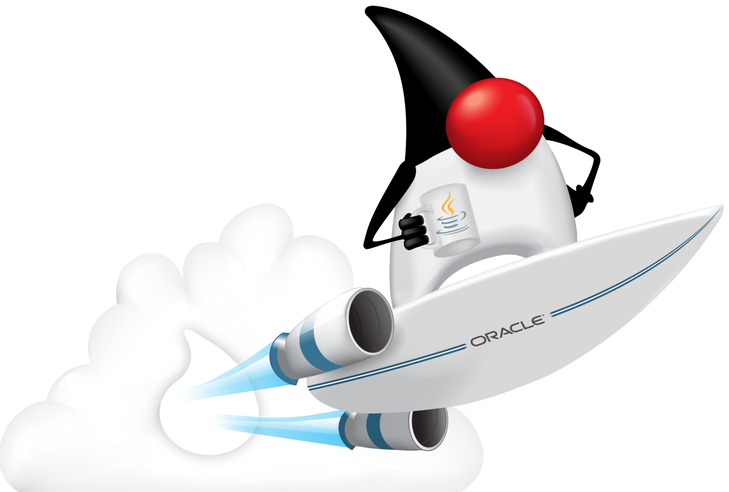
QXD0007 - Programação Orientada a Objetos



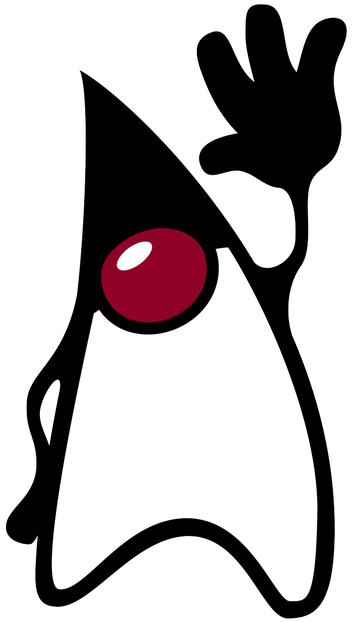
Prof. Bruno Góis Mateus (brunomateus@ufc.br)

Conteúdo

- Classes e Objetos
- Atributos
- Métodos
- Encapsulamento e Modificadores de acesso


ORACLE

Classes e Objetos



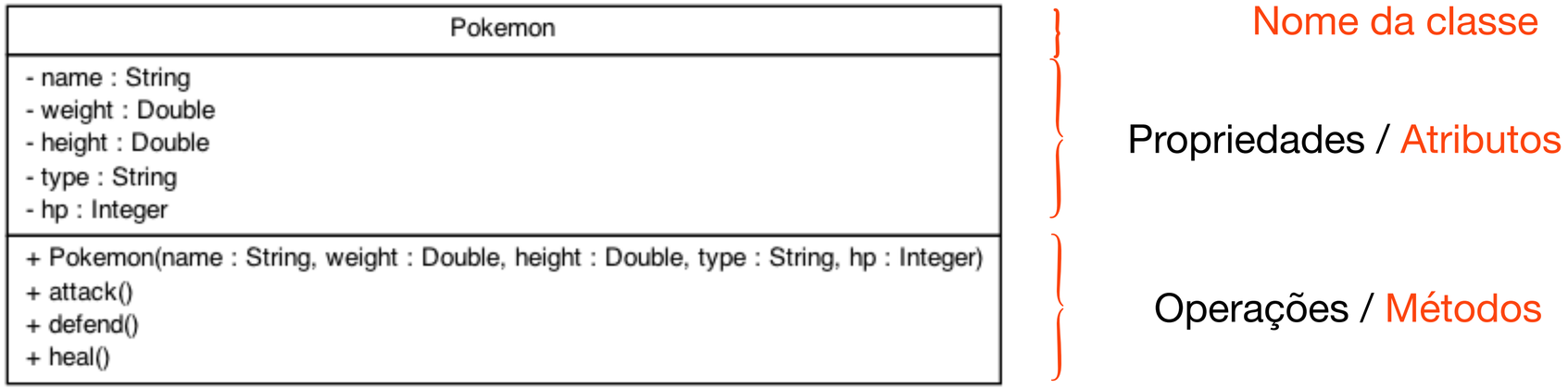
Classes e objetos

Classe de objetos

- Representa um conjunto de objetos com características afins.
- Define o comportamento dos seus objetos (através das suas operações/métodos) e quais estados eles podem alcançar (através das suas propriedades/atributos).

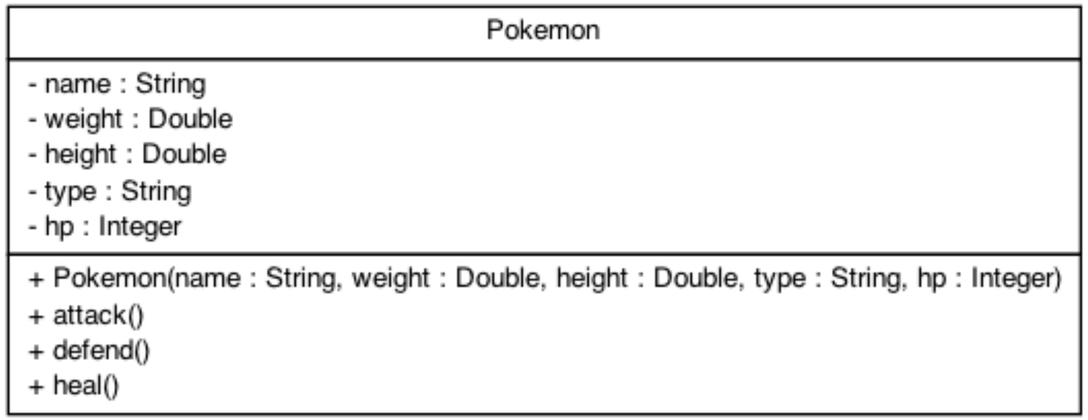
Classes e objetos

Representação UML



Classes e objetos

Assinatura




```java
public class Pokemon {
    // Atributos
    private String name;
    private Double weight;
    private Double height;
    private String type;
    private Integer hp;
    // Métodos
    public void attack()
    public void defend()
    public void heal()
}
```

Classes e objetos

Objeto

- É uma instância de uma classe
- É capaz de armazenar estados através de seus atributos e reagir à mensagens enviadas a ele, assim como se relacionar e enviar mensagens para outros objetos

Classes e objetos

Um objeto é uma entidade independente

- Estado interno: uma memória interna em que valores podem ser armazenas e o modificados ao logo da vida do objeto (conjunto de atributos ou variáveis de instância)
- Comportamento: conjunto de ações pré-definidas (métodos) através das quais o objeto responderá a demanda de processamento por parte de outros objetos

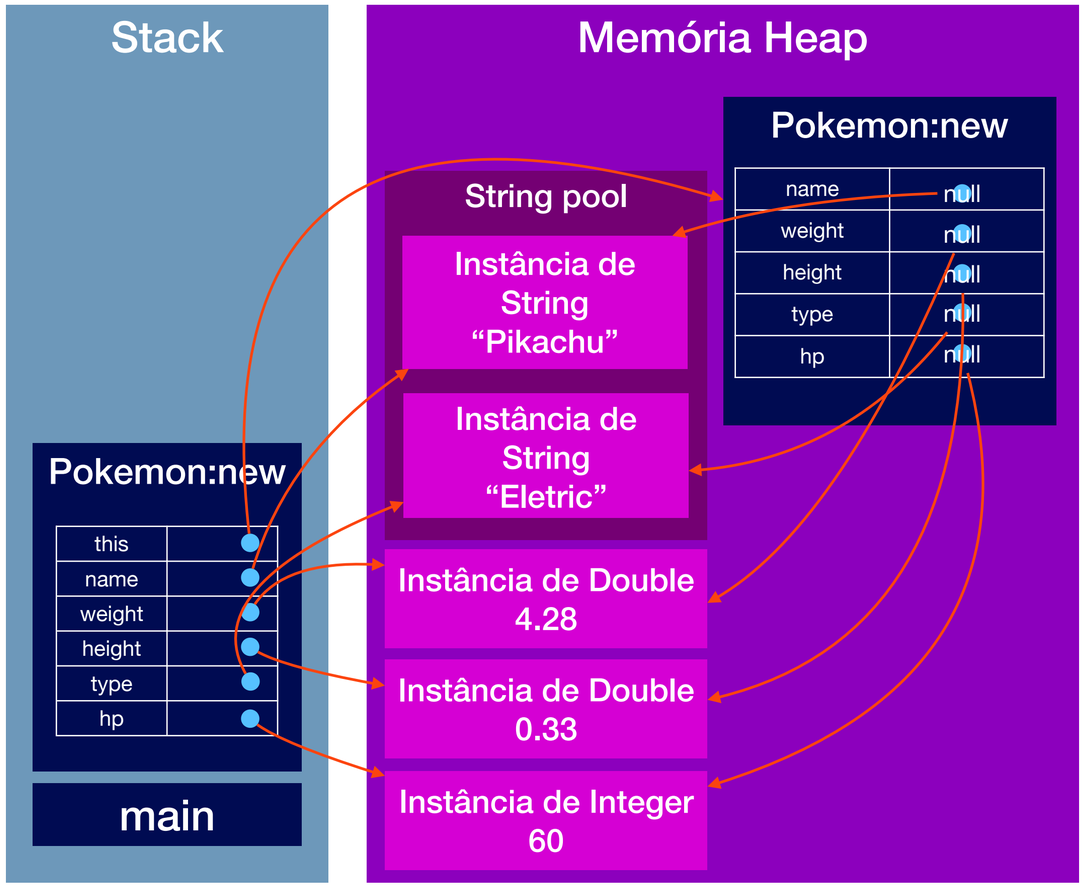
Classes e objetos

Instanciação

```
public class Pokemon {
  private String name;
  private Double weight;
  private Double height;
  private String type;
  private Integer hp;

  public Pokemon(String name, Double weight,
  Double height, String type, Integer hp) {
          this.name = name;
          this.weight = weight;
          this.height = height;
          this.type = type;
          this.hp = hp;
  }

public static void main(String[] args) {
   Pokemon pikachu = new Pokemon(“Pikachu",
   4.28,0.33, "Eletric", 60);
}
```



Atributos

Atributos

Atributos

- São propriedades de um objeto
  - Representam a estrutura dos dados que a classe vai representar
- Os atributos possuem valores que podem ser alterado durante o ciclo de vida de um objeto

Atributos

Atributo de classe

- Em determinadas situações compartilhar atributos entre objetos é uma boa ideia
- Usando a palavra reservada static é possível compartilhar tais atributos entre objetos da mesma classe

```
public class Pokemon {
  private static int contador = 0;
}
```

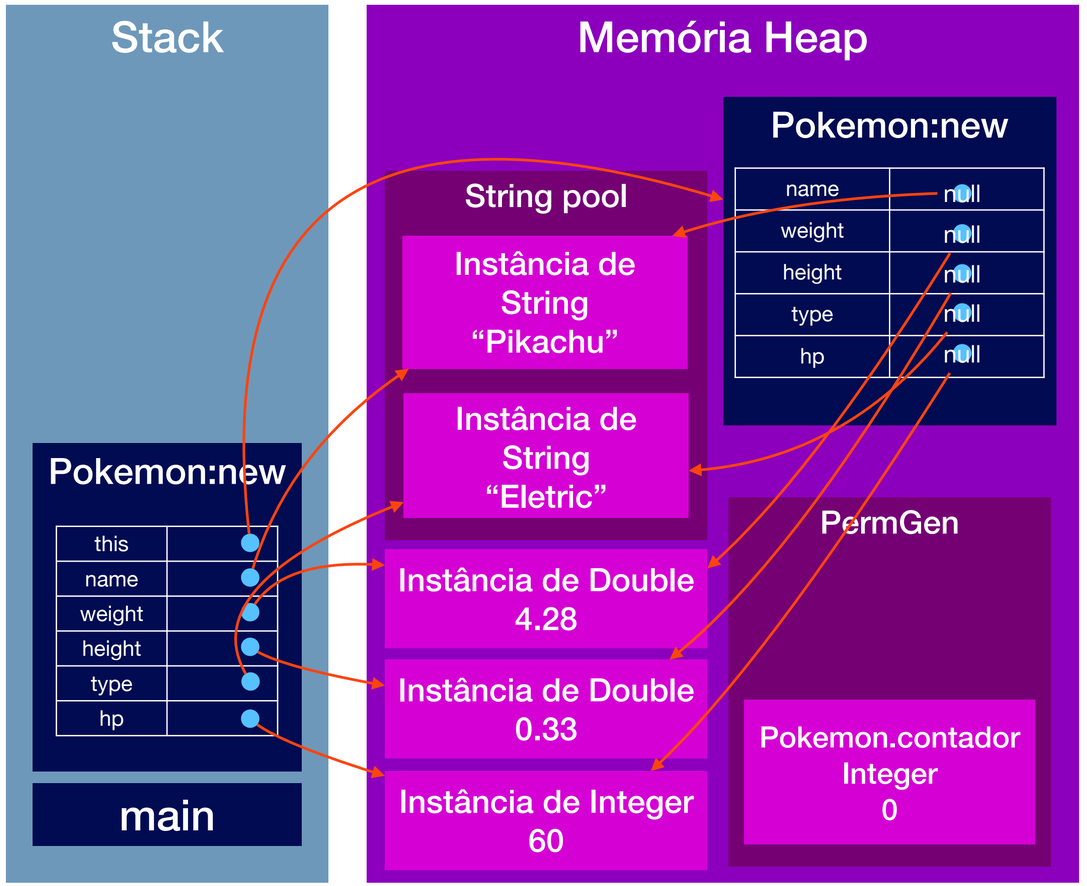
Atributos

Atributo de classe

```
public class Pokemon {
  private String name;
  private Double weight;
  private Double height;
  private String type;
  private Integer hp;
  private static Integer contador;

  public Pokemon(String name, Double weight,
  Double height, String type, Integer hp) {
          this.name = name;
          this.weight = weight;
          this.height = height;
          this.type = type;
          this.hp = hp;
  }

public static void main(String[] args) {
   Pokemon pikachu = new Pokemon(“Pikachu",
   4.28,0.33, "Eletric", 60);
}
```



Métodos

Métodos

Métodos

- Definem as “habilidades” dos objetos
- A ação definida por métodos ocorre somente quando um método é invocado através do objeto
- Assim como funções em C, métodos podem ou não retornar valores

Métodos

O método construtor

Métodos construtores

- Conceito introduzido por linguagens orientadas a objetos
- Determina as ações que serão executadas durante a criação (instanciação) de um objeto
- São métodos especiais que possuem o mesmo nome da classe e não possuem retorno
- Unicamente invocado no momento da criação do objeto através do comando new
- Fornece uma oportunidade de inicializar de maneira apropriada o objeto

Métodos

O método construtor

O construtor padrão

- Toda classe em Java possui um construtor, mesmo quando não declaramos nenhum
- Outra ação do construtor padrão é chamar o construtor da superclasse

```
public class Pokemon {

  public Pokemon() {
     super();
  }
}
}
  }
```

Métodos

A necessidade de um método construtor

Um construtor

- Possibilita ou obriga o usuário de uma classe a passar argumentos durante o processo de criação
  - Ex: Garante que um Pokemon sem nome não seja criado
- Pode fornecer diversas maneiras para se criar um objeto ao definirmos múltiplos construtores
- É uma boa prática definir pelo menos um construtor

Métodos

A necessidade de um método construtor

```
public class Pokemon {
  public Pokemon(String name, Double weight, Double height,
  String type, Integer hp) {
    this.name = name;
    this.weight = weight;
    this.height = height;
    this.type = type;
    this.hp = hp;
   }

public Pokemon(String name, String type) {
    this.name = name;
    this.type = type;
 }
}
```

Métodos

O método de classe

Métodos de classe

- Ex: Métodos da Classe Collections
- Só acessam atributos estáticos
- Não é possível acessar atributos dos objetos da mesma classe diretamente
  - Não temos acesso ao this, já que o método é chamado via classe

Métodos

Passagem de parâmetros

Passagem de parâmetros

- O número de parâmetros passados deve ser igual ao da definição
  - Cada parâmetro individualmente, deve ter tipo compatível com o da definição
  - Para tipos primitivos, a passagem é por valor
  - Para vetores (arrays) e objetos, uma cópia da referência é passada
- Uma expressão pode ser usada como argumento

Encapsulamento e Modificadores de acesso

Encapsulamento e Modificadores de acesso

Encapsulamento

- É uma forma de restringir o acesso ao estado interno do objeto e permitir a colaboração apenas via mensagens (requisitante/requisitado)
- Boa parte das linguagens orientadas a objetos da suporte a esse conceito através das classes
  - Escondendo o membros de uma classe
  - Escondendo o funcionamento das rotinas(métodos) do sistema
- Tal conceito é fundamental quando pensamos e desenvolver sistemas suscetíveis a mudanças

Encapsulamento e Modificadores de acesso

- O conjunto de métodos públicos de uma classe é também chamado de interface da classe
  - É a única maneira a qual você se comunica com objetos dessa classe
- Cada classe especifica sua interface de inicialização e operação
- Na maioria das linguagens OO, os métodos partes da interface são public

Encapsulamento e Modificadores de acesso

Programando para a interface

- É sempre bom programar pensando na interface
  - Como os usuário irão utilizar
- A implementação em si não tem tanta importância
- O usuário precisa saber usar a a classe, não precisa se preocupar como é feito
  - A implementação pode mudar com o tempo

Encapsulamento e Modificadores de acesso

Programando para a interface

- Forneça apenas o necessário para o usuário
  - Forneça uma interface mínima
- É melhor adicionar interfaces do que fornecer mais do que o necessário
- No momento de projetar a classe, sempre pense na perspectiva do usuário
- Tenha certeza que a classe em questão trata todos os requisitos do usuário
- A medida que o sistema evoluir, a sua classe pode evoluir também

Encapsulamento e Modificadores de acesso

Private

- private é um modificador de acesso ou modificador de visibilidade
- Marcando o atributo como privado evitamos o acesso ao mesmo
  - Apenas a própria classe pode acessá-lo diretamente
- Na orientação a objetos, é prática **quase que obrigatória** proteger seus atributos com private

Encapsulamento e Modificadores de acesso

Private

- Cada classe é responsável por seus atributos
  - Ela deve decidir se um novo valor é válido ou não
- Centralizar essa responsabilidade facilita mudanças futuras
- É comum que esconder a existência de um atributo por completo
  - Nesses casos tal atributo diz respeito ao funcionamento interno do objeto

Encapsulamento e Modificadores de acesso

Private

- O modificador private também pode ser usada para modificar o acesso a um método
  - Quando existe um método que serve apenas para auxiliar a própria classe
    - Ex: Quando há código repetido dentro de dois métodos da classe
- Sempre devemos expôr o mínimo possível de funcionalidades, para criar um baixo acoplamento entre as nossas classes

Encapsulamento e Modificadores de acesso

Public

- O modificador de acesso public permite a todos acessarem um método ou atributo
- É muito comum, e faz todo sentido, que seus atributos sejam private e quase todos seus métodos sejam public **(não é uma regra!)**
  - Toda conversa de um objeto com outro é feita por troca de mensagens
  - *Mais educado que mexer diretamente em um atributo que não é seu!*

Encapsulamento e Modificadores de acesso

Protected

- Restringe o acesso ao item sob o modificador deixando-o visível somente para as classes filhas da classe que possui o item e para classes que se encontram sob o mesmo pacote da classe que possui o item

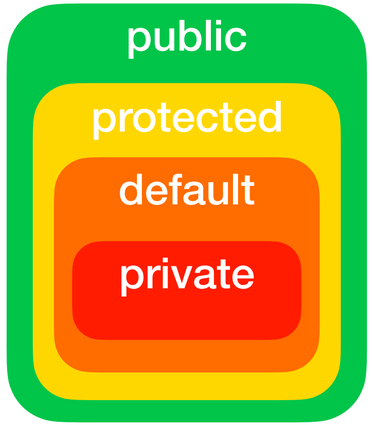
Encapsulamento e Modificadores de acesso

Default

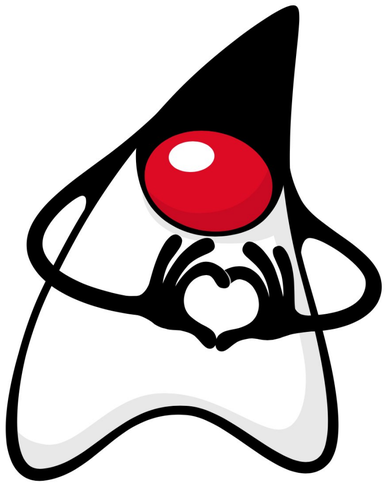
- E quando não há modificador de acesso ?
- O método ou atributo fica num estado de visibilidade intermediário entre private e o public
  - Para esse modificador não há uma palavra chave definida para o uso aqui
  - O modificador aqui é a omissão dos outros modificadores
- Restringe o acesso ao item (atributo ou método) sob o modificador deixando-o visível somente para as classes que se encontram no mesmo pacote da classe que possui o item

Encapsulamento e Modificadores de acesso

Visibilidade	+ public	# protected	~ default	- private
A partir da mesma classe	Visível	Visível	Visível	Visível
Qualquer classo no mesmo pacote	Visível	Visível	Visível	X
Qualquer classe final no mesmo pacote	Visível	Visível	X	X
Qualquer classe filha em pacote diferente	Visível	Visível	X	X
Qualquer classe em pacote diferente	Visível	X	X	X



Por hoje é só



Mão na massa

NOT VISIBLE