Programação Orientada a Objetos

Modificadores de Acesso: *public* e *private*

Prof. Tulio Alberton Ribeiro

Instituto Federal de Santa Catarina – IFSC
campus São José
tulio.alberton@ifsc.edu.br

31 de julho de 2014

Modificadores de acesso: `public` e `private`

Paradigma da programação orientada a objetos

- **Objetos interagem** com objetos através da **troca de mensagens**.
- A troca de mensagens ocorre através da **invocação de métodos** de objetos.
- A **invocação de métodos** pode ou não possuir parâmetros tanto quanto retorno ou não.
  - Métodos que possuem retorno, o tipo deve ser definido.
  - Métodos que não possuem retorno, são definidos como *void*.

Modificadores de acesso: `public` e `private`

Paradigma da programação orientada a objetos

- **Objetos interagem** com objetos através da **troca de mensagens**.
- A troca de mensagens ocorre através da **invocação de métodos** de objetos.
- A **invocação de métodos** pode ou não possuir parâmetros tanto quanto retorno ou não.
  - Métodos que possuem retorno, o tipo deve ser definido.
  - Métodos que não possuem retorno, são definidos como *void*.

Encapsulamento

- Emissor da mensagem **não precisa saber como o resultado foi obtido**, para este só importa o resultado.
- O emissor precisa **conhecer quais operações o receptor sabe realizar** ou quais informações o receptor pode fornecer.

Modificadores de acesso: `public` e `private`

Modificadores de acesso

Indicam **quais atributos** e **métodos** de um objeto estarão **visíveis aos demais objetos** do sistema.

private Os membros de uma classe (atributos e métodos) definidos como privados só poderão ser acessados pelos demais métodos da própria classe

public Os membros de uma classe definidos como públicos poderão ser invocados por métodos de qualquer classe

Modificadores de acesso: `public` e `private`

Modificadores de acesso

Indicam **quais atributos** e **métodos** de um objeto estarão **visíveis aos demais objetos** do sistema.

private: Os membros de uma classe (atributos e métodos) definidos como privados só poderão ser acessados pelos demais métodos da própria classe

public: Os membros de uma classe definidos como públicos poderão ser invocados por métodos de qualquer classe

Princípios da POO

- Geralmente **atributos** de uma classe **devem ser** declarados como **privados**
- **Métodos** geralmente devem ser **públicos**, porém há casos que um método só interessa a própria classe e assim este deve ser privado
- Isto **garante a integridade do estado do objeto**, pois somente métodos da própria classe poderão alterá-lo

Valores iniciais de atributos

```
1  public class Pessoa{
2     private String nome;
3     private String cpf;
4     private int anoNasc;
5
6     public void imprimirDados(){
7      System.out.println("Nome: " + nome);
8      System.out.println("CPF: " + cpf);
9      System.out.println("Ano: " + anoNasc);
10    }
11 }// fim da classe
```

```
11
12  Pessoa p = new Pessoa();
13  p.imprimirDados();
```

Valores iniciais de atributos

```
1  public class Pessoa{
2     private String nome;
3     private String cpf;
4     private int anoNasc;
5
6     public void imprimirDados(){
7      System.out.println("Nome: " + nome);
8      System.out.println("CPF: " + cpf);
9      System.out.println("Ano: " + anoNasc);
10    }
11 }// fim da classe
```

```
11
12  Pessoa p = new Pessoa();
13  p.imprimirDados();
```

- O que será impresso?

Valores iniciais de atributos

```
1  public class Pessoa{
2     private String nome;
3     private String cpf;
4     private int anoNasc;
5
6     public void imprimirDados(){
7      System.out.println("Nome: " + nome);
8      System.out.println("CPF: " + cpf);
9      System.out.println("Ano: " + anoNasc);
10    }
11 }// fim da classe
```

```
11
12  Pessoa p = new Pessoa();
13  p.imprimirDados();
```

- O que será impresso?

```
13  Nome:
14  CPF:
15  Ano: 0
```

Valores iniciais de atributos

- Em Java atributos de um objeto que não forem iniciados na criação deste objeto, receberão valores padrões
  - números ficam `0`,
  - boolean com `false` e
  - referências de objetos com `null`

Valores iniciais de atributos

- Em Java atributos de um objeto que não forem iniciados na criação deste objeto, receberão valores padrões
  - números ficam `0`,
  - boolean com `false` e
  - referências de objetos com `null`

Uma boa prática de programação

Sempre inicie os atributos de forma explícita

```
18  Pessoa p = new Pessoa();
19
20  p.definirNome("João");
21  p.definirCPF("123.456.789-00");
22  p.definirAno(1950);
```

Método construtor

- Trata-se de um método especial cujo objetivo é iniciar com valores os atributos de um objeto
- O método possui o mesmo nome da classe e não possui tipo de retorno
- Uma classe pode conter métodos construtores **sobrecarregados**
  - Sobrecarga de métodos consiste em declarar métodos com o mesmo nome, porém com assinaturas diferentes.
  - A assinatura de um método é dada pelo tipo de retorno e pela lista de parâmetros
- Ao criar um objeto o desenvolvedor indica qual construtor irá chamar

Nota

Método **construtor padrão** é aquele cuja de lista de parâmetros está vazia. Toda classe Java possui um construtor padrão vazio implícito.

Método construtor: exemplo

```
22 public class Pessoa{
23    private String nome, cpf;
24    private int anoNasc;
25
26    // metodo construtor padrão
27    public Pessoa(){
28       nome = ""; cpf = ""; anoNasc = 0;
29    }
30
31    // metodo construtor com 1 parâmetro
32    public Pessoa(String no){
33       nome = no; cpf = ""; anoNasc = 0;
34    }
35
36    // metodo construtor com 3 parâmetros
37    public Pessoa(String no, String c, int a){
38       nome = no; cpf = c; anoNasc = a;
39    }
40 }// fim da classe
```

Invocando métodos construtores

```
42  Pessoa a = new Pessoa();
43  Pessoa b = new Pessoa("Maria");
44  Pessoa c = new Pessoa("Maria","123.456.789-00",1959);
```