Programação Orientada a Objetos (POO)

Prof. Dr. Alan Souza

alan.souza@unama.br

2020

4. Construtor

- O objeto pode ser **instanciado** de várias formas:
  1. Declarar primeiro a variável e depois instanciar o objeto (**new**);
  2. Declarar a variável e instanciar o objeto ao mesmo tempo;
- Toda classe tem um construtor (construtor implícito) e é possível criar construtores parametrizados;
- Se programar um construtor parametrizado, então é possível:
  - Declarar a variável, colocar valores nos atributos e instanciar o objeto (tudo ao mesmo tempo!)

PARTE PRÁTICA

4. Construtor

ATRIBUTOS DA CLASSE "Pessoa" EM JAVA

```
public class Pessoa {
    String nome;
    String cpf;

}
```

4. Construtor

CONSTRUTOR PADRÃO DA CLASSE "Pessoa" EM JAVA

```
public class Pessoa {
 //atributos:
 String nome; String cpf;

 Pessoa() {
    System.out.println("Instanciando o objeto...");
    //isso é o construtor padrão
 }

}
```

4. Construtor

CONSTRUTOR PARAMETRIZADO DA CLASSE "Pessoa" EM JAVA

```
public class Pessoa {
 //atributos:
 String nome; String cpf;

 Pessoa ( String nome, String cpf ) {
      this.nome = nome;
      this.cpf  = cpf;
 }

}
```

Construtor parametrizado

4. Construtor

CRIANDO OBJETO DA CLASSE "Pessoa" EM JAVA

```
public class ProjetoPraia {
    public static void main(String args[]) {
        Pessoa p1 = new Pessoa();
    }
}
```

4. Construtor

ALTERANDO OS ATRIBUTOS DOS OBJETOS CRIADOS - SEM CONSTRUTOR

```
class ProjetoPraia {
    public static void main(String args[]) {
        Pessoa p2 = new Pessoa();

        p2.nome = "André Silva";
        p2.cpf  = "045.098.149-05";
    }
}
```

Alterando os atributos um a um

4. Construtor

ALTERANDO OS ATRIBUTOS DOS OBJETOS CRIADOS - COM CONSTRUTOR

```
class ProjetoPraia {
  public static void main(String args[]) {

    Pessoa p3 = new Pessoa( "Paulo",
"067.987.102-01" );

  }
}
```

declarando variável, atribuindo valores nos atributos (pelo Construtor Parametrizado) e criando o objeto (já "preenchido")

4. Construtor

Exercícios

1) Crie um projeto chamado ProjetoHerois e crie a classe Heroi dentro do pacote que contém a classe do método main. Em seguida, declare três atributos da classe Heroi e um método também. Depois, programe o construtor não parametrizado e outro parametrizado (usando todos os atributos).
2) Agora, na classe que contém o método principal (main), criar objetos sem e com construtor (um objeto de cada).

Programação Orientada a Objetos (POO)

Prof. Dr. Alan Souza

alan.souza@unama.br

2020

5. Modificadores de Acesso

- Também conhecidos como modificadores de visibilidade;
- Permitem controlar o acesso a classes, atributos, métodos e construtores;
- Existem quatro tipos:
  - **public**
  - **private**
  - **protected**
  - **default**

5. Modificadores de Acesso

- **public**
  - Classe: fica acessível para todo o projeto (todas as outras classes "enxergam" uma classe pública);
  - Atributos, construtores e métodos: ficam acessíveis para todo o projeto (podem ser chamados dentro de qualquer outra classe);

5. Modificadores de Acesso

- **private**
  - Classe: não pode ser declarada como private, exceto as classes aninhadas (classe dentro de outra classe);
  - Atributos, construtores e métodos: ficam incapazes de serem acessados por outras classes (só podem ser "vistos" dentro da classe que foram declarados);

PARTE PRÁTICA

5. Modificadores de Acesso

ATRIBUTOS DA CLASSE "Pessoa" CONFIGURADOS COMO public

```
public class Pessoa {
        public String nome;
        public String cpf;

}
```

O que acontece com o acesso a esses atributos fora da classe Pessoa?

R: nada, pois eles podem ser acessados fora da classe Pessoa...

5. Modificadores de Acesso

ATRIBUTOS DA CLASSE "Pessoa" CONFIGURADOS COMO private

```
public class Pessoa {
        private String nome;
        private String cpf;

}
```

O que acontece com o acesso a esses atributos fora da classe Pessoa?

R: ocorre erro, pois não podem ser acessados fora da classe Pessoa...

5. Modificadores de Acesso

ATRIBUTOS DA CLASSE "Pessoa" CONFIGURADOS COMO private

```
class ProjetoPraia {
    public static void main(String args[]) {
        Pessoa p2 = new Pessoa();

        p2.nome = "André Silva";
        p2.cpf  = "045.098.149-05";

    }
}
```

Erros de acesso!!!

5. Modificadores de Acesso

Exercícios

1) Altere a classe Heroi, configurando o acesso aos atributos desta classe como privados (private).
2) Depois do exercício 1, complete a frase: Por enquanto, o projeto não poderá ser compilado, pois _____.
3) Por que é importante não liberar acesso público (public) aos atributos de uma classe?

Programação Orientada a Objetos (POO)

Prof. Dr. Alan Souza

alan.souza@unama.br

2020

6. Encapsulamento

- Um dos pilares da POO (conceito muito importante);
- Técnica que torna os atributos da classe privados e fornece acesso a eles através de métodos públicos;
- Como os atributos são privados, eles ficam protegidos, pois não podem ser acessados de fora da classe;
- A única forma de acesso é através de métodos públicos;
- Normalmente, no método de acesso, podemos implementar regras antes de atribuir determinado valor em um atributo;
  - Exemplo: classe Pessoa: verificar se o CPF é válido...

6. Encapsulamento

- Podemos notar o uso de encapsulamento em quase todos os objetos que usamos no dia a dia;
- Exemplo:
  - **TV**: internamente existem vários componentes eletrônicos, mas interagimos somente através das opções do controle remoto;

6. Encapsulamento - JavaBeans

- JavaBeans são classes Java reutilizáveis, que encapsulam os atributos em um único objeto (bean);
- Uma classe JavaBean deve conter:
  - Construtor padrão (não parametrizado ou implícito);
  - Métodos públicos acessadores (métodos de configuração ***setters*** e ***getters***);
  - Às vezes, não é necessário criar métodos set e get para todos os atributos.

PARTE PRÁTICA

6. Encapsulamento

ENCAPSULANDO A CLASSE "Pessoa"

```
public class Pessoa {
        private String nome;
        private String cpf;

        public void setNome( String nome ) {
                this.nome = nome;
        }
        public String getNome( ) {
                return this.nome;
        }
}
```

Programar regra aqui no set

6. Encapsulamento

ENCAPSULANDO A CLASSE "Pessoa"

```
public class Pessoa {
        private String nome;
        private String cpf;

        public void setCpf( String cpf ) {
                this.cpf = cpf;
        }
        public String getCpf( ) {
                return this.cpf;
        }
}
```

Programar regra aqui no set

6. Encapsulamento

Exercícios

1) Programar o encapsulamento da classe Heroi.
2) Na classe que contém o método main(), atribuir valor e pegar valor através dos métodos de configuração set e get;
3) Programar regras básicas de validação:
   1) Gênero: conter no mínimo um caractere e no máximo 100;
   2) Força de ataque: maior que zero.
   3) Altura: deve ser maior que zero.

6. Encapsulamento

Exercícios

1) Programar o encapsulamento da classe Pessoa.
2) Na classe que contém o método main(), atribuir valor e pegar valor através dos métodos de configuração set e get
3) Programar regra básica de validação do CPF na classe Pessoa:
   - Se o CPF passado for 000.000.000-00, 111.111.111-11, 222.222.222-22, 333.333.333-33, 444.444.444-44, 555.555.555-55 , 666.666.666-66, 777.777.777-77, 888.888.888-88 ou 999.999.999-99; então mostrar a mensagem “CPF inválido” na tela e não alterar o valor do atributo **cpf** da classe **Pessoa**.