FACAMP
FACULDADES DE CAMPINAS

Laboratório de Programação

Aula 3

Java e Orientação a Objetos

2º semestre de 2019

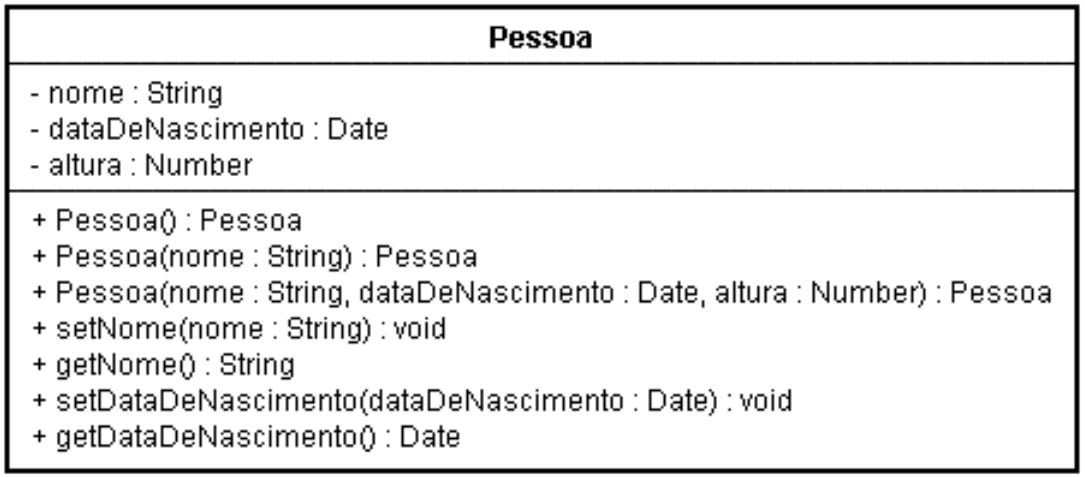
Prof José Martins Junior

Classes em Java

- As classes em Java, como já visto
  - Definem modelos para instâncias de objetos
  - São identificadas por nomes simples ou compostos (sem espaços ou caracteres especiais)
  - O código-fonte de uma classe deve ser armazenado em um arquivo com o mesmo nome da classe + extensão **.java**
  - Uma classe deve ser compilada (javac) e produzida a sua versão *bytecode*, armazenada em um arquivo com o mesmo nome, mas com extensão **.class**

Exemplo de classe

```
//o código deve ser armazenado em um arquivo Pessoa.java
public class Pessoa {  //chaves delimitam o código da classe
    //atributos de classe - início
    private String nome;
    private java.util.Date dataDeNascimento;
    private Number altura;
    //atributos de classe - fim
    public Pessoa() {
    }
    public Pessoa(String n) {
        nome = n;
    }
    // ... outros métodos
}
```



Instância de objetos

- Na maioria das vezes, utiliza-se o operador **new**
  - O operador **new** referencia uma das implementações (se existirem várias) do construtor da classe; método especial que tem o mesmo nome da classe
  - Existem exceções nas quais a instância de classes não se dá pelos métodos construtores (que são declarados privados), como no caso de *singletons*
- Exemplos de instâncias

```
Pessoa p1 = new Pessoa();
Pessoa p2 = new Pessoa("Maria");
```

  - Cada variável, p1 e p2, guardará a referência de um objeto diferente, instanciado na memória (*heap*)

Construtores e Destrutores

- Um construtor de uma classe
  - É um método declarado sem retorno e com o nome da classe
  - Suporta sobrecarga (podem existir vários construtores)
    - Nenhum (construtor *default*) ou vários parâmetros de entrada
  - Não pode ser chamado em um objeto já existente
- Destrutores em Java
  - Não são necessários, pois o *garbage collector* retira objetos não mais utilizados do *heap*
  - Porém, há situações em que se deve liberar recursos do sistema (arquivos aberto, conexões com bancos de dados, e outros)
    - Java permite a definição de um método `finalize()` que é chamado pelo *garbage collector* ao retirar um objeto da memória
    - Na prática, não é muito bom confiar totalmente nisso

Outros métodos

- Dois grupos principais
  - Alteradores (setters) - alteram os dados de um objeto

    Ex.: `public void setAltura(Number altura) { ... }`
  - Acessadores (getters) - acessam os dados de um objeto

    Ex.: `public String getNome() { ... }`
- Não se usa passagem de parâmetros por referência em Java
  - Método recebe parâmetros declarados como entrada
    - Alterações internas nas variáveis de entrada não são visíveis fora do método

```
Pessoa p = new Pessoa("Joao");
p.setAltura(new Float(1.70f));
```

  - Retorna parâmetros do tipo declarado para o método
    - No exemplo abaixo, o método `getNome()` da variável p, que referencia um objeto do tipo Pessoa, retornará um objeto String

```
System.out.println("Nome: " + p.getNome());
```

Métodos estáticos (static)

- Campos de dados
  - Não mudam de uma instância de uma classe para outra

    Ex.: `public static int n = 0;`
  - Utilizado também para a declaração de constantes

    Ex.: `public static final int ZERO = 0;`
- Métodos
  - Não operam quando da instância de uma classe
  - Só podem acessar campos estáticos
  - JVM faz a chamada do método `main()` sem a instancia da classe
    - Para que sejam acessados métodos e atributos não estáticos dessa mesma classe, uma instância explícita (**new**) deve ser realizada (mesmo que chamada dentro do método `main()`)
- Campos e métodos estáticos de uma classe podem ser acessados (se públicos) por outros objetos, sem instância

  Ex.: `File.separator`

Pacote (package)

- Java permite a definição de pacotes
  - Ex.: `java.lang`
- O nome de um pacote refere-se à estrutura de diretórios como está organizado
  - Tal estrutura é mantida em arquivos .zip, .jar, .tar, e portanto, um pacote pode ser compactado em um único arquivo
- Um pacote é declarado com a sentença **package**
- Padrão para criação de pacotes
  - Usar uma estrutura de diretórios que represente o ***full qualified domain name*** do local onde o projeto está sendo desenvolvido
  - Ex.: um projeto da aula de Java

```
package br.instituicao.java;
```

Visibilidade

- Membros e métodos devem especificar a visibilidade externa (permissões de acesso) do objeto
- **public** - indica que aquele membro ou método tem visibilidade externa para qualquer outro objeto
  - Exs.:
```
public String nome;
public void setNome(String nome);
```
- **private** - indica que é um membro ou método privado da classe
  - Exs.:
```
private String nome;
private void setNome(String nome);
```
- **protected** - visível para todas as subclasses
- Sem modificador **(default)** - visível para as classes do pacote

Bibliografia

DEITEL, P.; DEITEL, H. Java TM: como programar. 8ª edição. São Paulo: Pearson Prentice Hall, 2012. 1144p.

GOSLING, J.; ARNOLD, K.; HOLMES, D. A Linguagem de Programação Java. 4ª edição, Porto Alegre: Bookman, 2007.