



CIÊNCIAS DA COMPUTAÇÃO

FACULDADE DE CIÊNCIAS | UNIVERSIDADE AGOSTINHO NETO



FUNDAMENTOS DE PROGRAMAÇÃO

Docente:

✓ Lufialuiso Sampaio Velho, MSc.

Monitor

✓ João Pedro



Cap. VIII

- Classes

CAP. VIII – Classes

Se repararmos, tudo o que nos rodeia é nada mais do que um conjunto de objectos; Ex: **mesa, carro, telefones, isto é, tudo o que é real.**

Para as linguagens orientadas a objecto, tudo que existe pode ser criado e manipulado a partir de um modelo. Para tal introduz-se o conceito de **classes**.

A **Classe** é considerada como o modelo para a criação de vários Objectos.

CAP. VIII – Classes (cont.)

Uma classe é constituída por **atributos** e **operações**.

Atributos (campos) definem as propriedades (características) de uma determinada classe.

Ex: Para a Classe Aluno, considera-se como atributo: nome, morada, sexo, num_aluno, etc.

Operações (métodos) definem comportamentos ou acções que serão exercidas por um determinado objecto bem como o modo como os objectos podem se relacionar entre si. Ex: **consultarNome**, **LevantarDinheiro**, **inserirMorada**, etc.

CAP. VIII – Classes (cont.)

Sintaxe	Exemplo
<pre>public class nomeDaClasse { /*Declaração de variáveis*/ /*definição de métodos*/ }</pre>	<pre>public class Pessoa { /*Declaração de variáveis*/ /*definição de métodos*/ }</pre>

CAP. VIII – Classes (cont.)

Objecto é definido como instância de uma classe. Podemos considerar igualmente como algo concreto criado a partir de uma classe (modelo).

Ex1: considere uma forma de Bolo (classe) e o bolo (objecto)

Ex2: carro (classe) e Hyunday (objecto)

Representação gráfica (Classe - Objecto) – Notação UML

CAP. VIII – Classes (cont.)

Objecto é definido como instância de uma classe. Podemos considerar igualmente como algo concreto criado a partir de uma classe (modelo).

Ex1: considere uma forma de Bolo (classe) e o bolo (objecto)

Ex2: carro (classe) e Hyunday (objecto)

CAP. VIII – Classes (cont.)

Representação gráfica (Classe - Objecto) – Notação UML



CAP. VIII – Classes (cont.)

Podemos considerar as classes como o recurso para a criação de tipos abstractos de dados (**TAD**); pelo facto de permitir a agregação de diferentes tipos de dados e dar-nos a possibilidade de serem implementados métodos para a manipulação destes dados.

Ex: **uma lista telefónica** (nome, endereço, sexo, telefone)

Instanciação de Classes (criação de Objectos)

Os objectos são criados da seguinte forma:

Sintaxe:

<classe> <nomeDoObjecto> = new <construtor>(parâmetros)

Ex1: Carro accent = new Carro();

Ex2: Carro accent= new Carro("Accent",2013,"LD-30-91-AF");

System.out.println(accent.ano); // **mostrará 2013**

CAP. VIII – Array de Objectos

- ❑ Também é possível criar um array de uma classe.

```
Pessoa todasPessoas[] = new Pessoa[10]; /*criou-se um vector de pessoa com 10 posições*/
```

- ❑ Para preencher o vector deve-se sempre instanciar a posição em causa:

```
todasPessoas[0] = new Pessoa();  
todasPessoas[0].nome = "Pedro Kondo";  
todasPessoas[0].idade = 19;  
todasPessoas[0].genero = 'M';
```

Ou:

```
Pessoa p = new Pessoa(); /*variavel auxiliar*/  
p.nome = "Pedro Kondo";  
p.idade = 19;  
p.genero = 'M';  
todasPessoas[0] = p;
```

CAP. VIII – ArrayList de Objectos

- ❑ Também é possível criar um ArrayList de uma classe.

```
ArrayList<Pessoa> todasPessoas = new ArrayList<>();
```

- ❑ Para preencher o ArrayList utiliza-se uma variável auxiliar:

```
Pessoa p = new Pessoa(); /*variavel auxiliar*/  
p.nome = "Pedro Kondo";  
p.idade = 19;  
p.genero = 'M';  
todasPessoas.add(p);
```

CAP. VIII – Classes (cont.)

Construtor: é um método sem retorno e **não void** cujo o nome é semelhante ao da classe, e é utilizado para a criação de objectos; ou seja inicializa o estado dos objectos.

Ex: Dada a classe Pessoa com os atributos: **nome**, **idade** e **sexo**, podemos definir um método construtor para os objectos desta classe. Vejamos:

```
Public Pessoa (String nome, int idade, char sexo){
```

```
this.nome=nome;
```

```
this.idade=idade;
```

```
this.sexo=sexo;}
```

Instanciação: Pessoa Professor =new Pessoa(“Lufialuiso”, 34,’M’);

Nota: Em caso de não criarmos algum construtor, o compilador assumirá um construtor *default* sem parâmetros (p.e Pessoa()) na criação do Objecto.

CAP. VIII – Classes (cont.)

Membros da Classe e de Instância (atributos, métodos)

Membros da Classe são também considerados como membros estáticos(**static**), indica que os mesmos são comuns para todos os objectos e podem ser executados sem que instanciemos um objecto.

Ex: **static float media** – esta variável pode ser acedida no método **main** directamente sem a necessidade de ser instanciado um objecto.

Nota: O valor atribuído a este atributo será o mesmo para todos os objectos.

CAP. VIII – Classes (cont.)

Membros da Classe e de Instância (atributos, métodos)

Membros de Instância: são membros não estáticos, indica que cada objecto possui o seu próprio valor e método.

Ex: `float media` – para cada instância criada, este atributo assume um valor diferente.

Nota: Um método estático não pode invocar um não estático da mesma classe caso não seja instanciado um objecto.

CAP. VIII – Classes (cont.)

Modificadores de Acesso

Os modificadores de acesso determinam o nível de acesso dos atributos e métodos de uma classe em relação as outras classes.

Estes podem ser:

Público (public): indica que os elementos (**atributos, métodos**) podem ser acedidos dentro como fora da classe.

Privado (private): indica que os elementos (**atributos, métodos**) são acedidos somente no interior da classe, não sendo visíveis fora dela. pela própria classe e pelas classes do mesmo pacote.

CAP. VIII – Classes (cont.)

Protegido (protected): indica que os elementos (**atributos, métodos**) podem ser acedidos pela própria classe, subclasses desta classe ou ainda classes que estejam no mesmo pacote.

Default: consideramos por *default* (padrão) aos elementos cujo o modificador não é especificado. Ex: **int nome** .

Isto implica que estes elementos podem ser acedidos pela própria classe e pelas classes do mesmo pacote.

CAP. VIII – Modificadores de acesso

- ❑ Classe pessoa com modificadores de acesso private na variável idade.

Resolução

```
public class Pessoa {  
    /*Declaração de variáveis*/  
    String nome;  
    private int idade; /*variaveis privada*/  
    char genero;  
  
    /*definição dos métodos para acesso a variavel privada idade*/  
    void setIdade(int idade){  
        this.idade = idade;  
    }  
    int getIdade(){  
        return idade;  
    }  
    /*definição dos outros métodos*/  
    int tamanhoNome(){  
        return nome.Length();  
    }  
    void anoNascimento(int anoActual){  
        int nascimento = anoActual - idade;  
        System.out.println("Ano de Nascimento = "+ nascimento);  
    }  
}
```

CAP. VIII – Classes (cont.)

Referência `this`

É uma palavra reservada, utilizada para fazer referência ao objecto corrente de uma classe. É utilizada para aceder aos membros de instância dentro da classe. Ex: `this.nome` – referencia o atributo nome do objecto corrente.

-A referência `this` não deve ser utilizada com membros da classe (`static`)

-A referência `this` pode ser utilizada para distinguir uma variável de instância de outra variável local (p.e um `parâmetro`) caso as mesmas tenham o mesmo nome.

CAP. VIII – Classes (cont.)

Encapsulamento de Dados

É um conceito utilizado em programação orientada a objectos para determinar a regra de acesso aos atributos e métodos de uma classe; isto é ocultar a informação de modos que os interessados a acedê-las obedecem as regras que impomos.

Para tal existem dois métodos **setters** (**modificador**) e outro **getters** (**inspector**), também conhecidos por **get** e **set**.

Os métodos **getters** permitem obter o estado (**valor**) de um determinado objecto num determinado instante. São conhecidos por **inspectores** pois fornecem informações.

Os métodos **setters** permitem alterar o estado de um determinado objecto num determinado instante.

CAP. VIII. Exercícios

1. Problema – Aluno

1.1 Crie uma classe Aluno que agrega os seguintes membros: Três campos: nome, idade e o número de estudante (com modificador padrão).

Dois métodos:

- uma função **public int tamanho()** que retorna o tamanho do nome
- um procedimento **public void imprimirDados()** que imprime o nome a idade e o número do aluno.

1.2 Crie a classe TesteAluno com os seguintes membros:

- a) Um vector todosAlunos[50] do tipo Aluno
- b) Um procedimento **public static void inserirAluno()** – que insere um novo aluno no vector.
- c) Um procedimento **public static void consultarAluno()** – que solicita o numero do aluno e apresenta os seu dados.
- d) Um procedimento **public static void listarTodosAlunos()** – que lista os dados de todos alunos armazenados no vector.
- e) Implemente um menu interativo no método main e chama as operações :
 - 1 – Inserir Aluno
 - 2 – Consultar Aluno
 - 3 – Listar Alunos
 - 4 - Sair

CAP. VIII. Exercícios

2. Problema – Conta Bancaria

2.1 Crie uma classe Conta que agrega os seguintes membros: Três campos: numero, nomeDono e o saldo da conta(com modificador padrão).

- Um procedimento public void **levantamento(double valor)** - que efectua um levantamento na conta em função do valor especificado.
- um procedimento **public void depositar(double valor)** que efectua um deposito na conta em função do valor especificado.
- um procedimento **public void imprimirDados()** que imprimi o número, o nome e o saldo da conta bancaria.

2.2 Crie a classe TesteConta com os seguintes membros:

- a) Um vector todascontas[50] do tipo Conta
- b) Um procedimento **public static void criarConta()** – que cria uma nova conta no vector.
- c) Um procedimento **public static void consultarConta()** – que solicita o numero do conta e apresenta os seu dados.
- d) Um procedimento **public static void listarContas()** – que imprimi todas as conta armazenadas no vector.
- e) Implemente um menu interativo no método main e chama as operações :
 - 1 – Criar conta
 - 2 – Consultar Conta
 - 3 – Listar Contas
 - 4 - Sair

Até a próxima Aula

