



CCM310

Arquitetura de Software e

Programação Orientada a Objetos

Profa. Dra. Gabriela Biondi

Prof. Dr. Isaac Jesus

Prof. Dr. Luciano Rossi

Classes e Objetos

Classes

- Representam itens do mundo real:
 - Exemplos:
 - Pessoas / Veículos / Robôs
- São Compostas de:
 - **Atributos** (variáveis de instância)
 - **Métodos** (funções-membro)

Objetos

- Todo objeto pertence a uma classe
- Representam instâncias de entidades no mundo real
- São criados a partir das classes
- São instâncias das classes
- Os objetos associam valores específicos aos atributos

Instanciar

- Instanciar é criar um objeto, ou seja, alocar um espaço na memória, para posteriormente poder utilizar os métodos e atributos que o objeto dispõe
- Em informática instância é usada com o sentido de exemplar. No contexto da orientação ao objeto, instância significa a concretização de uma classe

Classes e Objetos

Quando definimos uma classe de objetos, estamos, na verdade, definindo que propriedades e métodos o objeto possui!

Diferença entre Classe e Objeto

□ Classe:

- É um modelo
- De maneira mais prática, é como se fosse a **planta de uma casa**

□ Objeto:

- É criado a partir da classe
- É como se fosse a **própria casa construída**
- Pode-se construir várias casas a partir da mesma planta, assim como podemos instanciar vários objetos de uma só classe

Estrutura de uma Classe

Nome da Classe
- Atributos
- Métodos

- Atributos são variáveis que armazenam informações do objeto
- Métodos são as operações (funções) que o objeto pode realizar

Exemplo de Classe



Calculadora

a: float
b: float

Soma()
Subtrai()
Multiplica()
Divide()

Atributos

Métodos

Atributos

- Variáveis de instância
- Atributos são variáveis em que o objeto armazena informações
- Lembram:
 - os campos de uma ***struct*** em C

Métodos

- São sequências de **declarações** e **comandos** executáveis **encapsulados** como se fossem um mini-programa
- Similares:
 - sub-rotinas
 - procedimentos
 - funções

Exemplo de Classe e Objetos

Como você modelaria
um carro?

Quais atributos e
métodos você incluiria
na sua classe Carro?

Exemplo de Classe e Objetos

Como você modelaria
um carro?

Quais atributos e
métodos você incluiria
na sua classe Carro?

Carro

cor: string
nome: string
marca: string
rodas: int
portas: int
lugares: int
preco: double
ano: int
ligado: boolean

ligar()
desligar()
andarFrente()
virarDireita()
virarEsquerda()

Exemplo de Classe e Objetos

Carro

cor: string
nome: string
marca: string
rodas: int
portas: int
lugares: int
preco: double
ano: int
ligado: boolean

ligar()
desligar()
andarFrente()
virarDireita()
virarEsquerda()



Exemplo de Classe e Objetos

Carro

cor: string
nome: string
marca: string
rodas: int
portas: int
lugares: int
preco: double
ano: int
ligado: boolean

ligar()
desligar()
andarFrente()
virarDireita()
virarEsquerda()



Exemplo de Classe e Objetos

Cat
size: float
color: string
positionX: float
positionY: float
moveForward()
moveBackward()
moveUP()
moveDown()

}

Cat garfield;
Cat tom;
Cat felix ;
Cat scratchy;



Exemplo de Classe e Objetos

```
Robot  
  
positionX: float  
positionY: float  
direction: float  
  
moveForward()  
moveBackward()  
turnLeft()  
turnRight()
```

Robot b1;
Robot b2;
Robot b3;



Exemplo de Classe e Objetos

Movie
name: string
storyline: string
runtime: float
play() stop() pause()

Movie poderosoChefao;
Movie senhorDosAneis;
Movie forrestGump;



Orientação a Objetos no



Definindo a Classe

- A definição da classe indica ao compilador que **métodos** e **atributos** pertencem à **classe**
- A classe é iniciada pela palavra-chave ***class*** seguida do nome da classe
- O corpo da classe é colocado entre chaves { ... }

Exemplo de classe

```
1 // Fig. 3.1: GradeBook.java
2 // Declaração de Classe com um método.
3
4 public class GradeBook
5 {
6     // exibe uma mensagem de boas-vindas para o usuário GradeBook
7     public void displayMessage()
8     {
9         System.out.println( "Welcome to the Grade Book!" );
10    } // termina o método displayMessage
11
12 } // fim da classe GradeBook
```

Exemplo: Criando um Objeto da Classe

Declarando um Objeto:

```
<nome_da_classe> <nome_do_objeto>
```

```
GradeBook myGradeBook
```

Instanciando um Objeto:

```
myGradeBook = new GradeBook()
```

a palavra-chave ***new***
cria um novo objeto
da classe especificada



Operador ponto (•)

- É usado para acessar variáveis de instância e métodos a partir de um objeto.
- Exemplo:
- `myGradeBook • displayMessage()`

Chama o método
displayMessage do
objeto **myGradeBook**

Exemplo de classe

```
1// Fig. 3.1: GradeBook.java
2// Declaração de Classe com um método.
3
4public class GradeBook
5{
6    // exibe uma mensagem de boas-vindas para o usuário GradeBook
7    public void displayMessage()
8    {
9        System.out.println( "Welcome to the Grade Book!" );
10   } // termina o método displayMessage
11
12} // fim da classe GradeBook
```

ponto separador:
acessa
membros/variáveis a
partir de um objeto

palavra-chave ***new*** cria
um novo objeto da
classe especificada à
direita da palavra-
chave

```
1// Fig. 3.2: GradeBookTest.java
2// Cria um objeto GradeBook e chama seu método displayMessage.
3
4public class GradeBookTest
5{
6    // método main inicia a execução de programa
7    public static void main( String args[] )
8    {
9        // cria um objeto GradeBook e o atribui a myGradeBook
10       GradeBook myGradeBook = new GradeBook();
11
12       // chama método displayMessage de myGradeBook
13       myGradeBook.displayMessage();
14   } // fim de main
15
16} // fim da classe GradeBookTest
```

Encapsulamento

Encapsulamento

- Encapsular significa separar o programa em partes independentes, o mais isoladas possível
- O propósito do encapsulamento é o de organizar os dados que sejam relacionados, encapsulando-os em objetos / classes
- O uso do modificador de acesso **private** nos membros, combinado com o uso dos métodos **get** e **set** é uma parte do encapsulamento

Encapsulamento - Modificadores de Acesso

- *public:*
 - Indica que um método ou atributo é acessível a outras funções e funções-membro de outras classes
- *private:*
 - Torna um membro de dados ou uma função-membro acessível apenas a funções-membro da classe
- *protected:*
 - Torna o membro acessível às classes do mesmo pacote ou através de herança

Encapsulamento - Modificadores de Acesso

- *package-private* (modificador padrão):
 - Se nenhum modificador for utilizado, todas as classes do mesmo pacote têm acesso ao atributo, construtor, método ou classe

Encapsulamento - Modificadores de Acesso

Regra Geral:

- atributos (variáveis de instância) devem ser declarados como *private*
- funções-membro (métodos) devem ser declaradas como *public*

Encapsulamento - Funções *set* e *get*

- Se as variáveis de instância devem ter especificador de acesso *private*, como então elas serão acessadas pelos objetos que já foram instanciados?

Encapsulamento - Funções *set* e *get*

- Para isso utilizamos as funções (com acesso *public*):
 - *set*: para atribuir valores
 - *get*: para obter valores

Code Conventions

- Nomes de Classes devem ser substantivos com a primeira letra de cada palavra escrita em maiúsculo e o restante das letras em minúsculo
- Exemplos:
 - class Movie
 - class Calculadora
 - class ImageSprite

Code Conventions

- Nomes de Métodos devem ser verbos escritos com letras minúsculas, com exceção da primeira letra de cada palavra após a primeira palavra
- Exemplos:
 - run();
 - runFast();
 - getBackground();

Code Conventions

- Nomes de objetos seguem a mesma regra dos nomes de métodos
- Exemplos:
 - myGradeBook;
 - application;
 - testScope;

Atributos vs. Variáveis Locais

Atributos (Variáveis de Instância):

Existem por toda a vida do **objeto**

São representados como **membros de dados**

Todo **objeto** de classe mantém sua própria **cópia** de atributos

Variáveis Locais:

Variáveis declaradas no corpo de uma definição de **método**

Não podem ser utilizadas fora do corpo deste método

São conhecidas apenas por este método

Quando o método **termina**, os valores das respectivas **variáveis locais** são perdidos
(escopo local)

Packages

Pacotes

Pacotes

- Os pacotes são utilizados para organizar as classes em diretórios distintos, dependendo da similaridade ou relacionamento existente entre as classes.
- Por exemplo, como já foi mostrado, a classe *Scanner* faz parte do pacote *java.util* e para utilizarmos a classe Scanner precisamos importar o pacote e a classe: `import java.util.Scanner`
- Podemos ter outras classes chamadas *Scanner*, desde que elas estejam em outros pacotes.

Pacotes

- Os pacotes estão diretamente relacionados com os diretórios em que vamos salvar nossas classes
- O padrão da nomenclatura dos pacotes está ligado com o nome da empresa que os criou. Exemplo:

```
import br.edu.fei.roboticsAPI.applicationModel.RoboticsAPIApplication;
```

- O nome do pacote indica que a classe *RoboticsAPIApplication* está no diretório **br -> edu -> fei -> roboticsAPI -> applicationModel**

Exercício 1 - Crie a classe Pessoa

- Variáveis:
 - `cpf` e `nome`, ambos *private* e do tipo *String*.
 - `idade` *private* e `int`
- Métodos:
 - `set` e `get` para cada um dos atributos
- `main()`:
 - Classe `TestePessoa`
 - crie três objetos (`p1`, `p2` e `p3`) do tipo `Pessoa`
 - obtenha pelo teclado o valor de `cpf`, `nome` e `idade` para `p1`, `p2` e `p3`
 - initialize os atributos de `p1`, `p2` e `p3` com os métodos `set`
 - exiba o conteúdo dos atributos de `p1`, `p2` e `p3` utilizando o método `get`

Exercício 2 - Crie a classe Swapper

- Variáveis:
 - **x** e **y**, ambos *private* e do tipo *float*.
- Métodos:
 - *set* e *get* para cada um dos atributos
 - *void swap()* que troca os valores de **x** e **y**
- main():
 - classe SwapperDemo
 - crie um objeto, chamado **troca**, do tipo **Swapper**
 - obtenha pelo teclado o valor de **x** e **y** para o objeto **troca**
 - inicialize os atributos de **troca** com os métodos *set*
 - utilize o método *swap()* para trocar os valores de **x** e **y**
 - exiba os valores trocados utilizando os métodos *get*

Exercício 3 - classe Retângulo

- Atributos:
 - `lado1` e `lado2`, ambos *private* e do tipo *int*
- Métodos:
 - `set` e `get` para os atributos
 - `area()`, que retorna a área do retângulo
 - `perimetro()`, que retorna o perímetro do retângulo
- `main()`:
 - crie dois objetos (`ret1` e `ret2`) do tipo `Retangulo`
 - obtenha pelo teclado o valor de `lado1` e `lado2` para `ret1` e `ret2`
 - inicialize os atributos de `ret1` e `ret2` com os métodos `set`
 - exiba o conteúdo dos atributos de `ret1` e `ret2` utilizando os `gets`
 - exiba o perímetro e a área dos objetos `ret1` e `ret2` por meio dos métodos `area()` e `perimetro()`

Exercício 4 - classe Televisão

- Atributos (Variáveis de Instância):
 - `modelo` - *private* e do tipo *string*
 - `preco` e `tamanho` - *private* e do tipo *float*
 - `volume` e `canal` - *private int*
 - `ligada` - *private boolean*
- Métodos:
 - `set` e `get` para todos atributos
 - `alteraVolume(int)` - aumenta ou diminui o valor atual de volume, dado arg
 - `alteraCanal()` - aumenta ou diminui o valor atual do canal em 1 unidade
- main():
 - crie dois objetos (`tv1` e `tv2`) do tipo `Televisao`
 - obtenha pelo teclado os valores dos atributos e inicialize com `set`
 - altere o volume e o canal de `tv1` e `tv2`
 - desligue a `tv2`

Exercício 5 - classe Funcionário

(3.14 adaptado Deitel) – Crie uma classe chamada **Funcionario** com três partes de informações como atributos: um nome (tipo string), um sobrenome (tipo string) e um salário mensal (tipo double). Forneça uma função set e uma função get para cada atributo. Se o salário mensal não for positivo, configure-o como 0 (zero). Forneça uma função que retorna o salário anual de **Funcionario**. Escreva um programa de teste que demonstre as capacidades da classe **Funcionario**. Crie dois objetos da classe **Funcionario**; utilize a função set para entrar com as informações de cada **Funcionario** (nome, sobrenome e salário mensal) e exiba o salário anual de cada um. Então, dê a cada **Funcionario** um aumento de 10% e exiba novamente o salário anual de cada um.

Exercício 6 - classe Invoice (Fatura)

(Deitel 3.13) Crie uma classe chamada **Invoice** que uma loja de suprimentos de informática possa utilizar para representar uma fatura de um item vendido na loja. Uma **Invoice** (fatura) deve incluir quatro partes de informações como atributos — um número *identificador* (tipo string), uma *descrição* (tipo string), a *quantidade* comprada de um item (tipo int) e o *preço* por item (tipo double). Forneça uma função *set* e uma função *get* para cada membro de dados. Além disso, forneça um método chamado *getInvoiceAmount* que calcula a quantia da fatura (isto é, multiplica a quantidade pelo preço por item) e depois retorna a quantidade como um valor double.

Exercício 6 - classe Invoice (Fatura)

Se a quantidade não for positiva, ela deve ser configurada como 0. Se o preço por item não for positivo, ele deve ser configurado como 0. Escreva um programa de teste que demonstre as capacidades da classe Invoice.

Exercício 7 - classe Perfil de Saúde

(3.17 adaptado Deitel) Um tema de saúde que tem estado no noticiário ultimamente é a informatização dos registros de saúde. Essa possibilidade está sendo abordada com cautela devido a questões sensíveis de privacidade e segurança, entre outras. A informatização dos registros de saúde pode tornar mais fácil que pacientes compartilhem seus perfis e históricos de saúde entre vários profissionais de saúde. Isso pode aprimorar a qualidade da assistência médica, ajudar a evitar conflitos e prescrições erradas de medicamentos, reduzir custos e, em emergências, salvar vidas. Neste exercício, você deve projetar a classe *PerfilDeSaude* “inicial” para uma pessoa.

Exercício 7 - classe Perfil de Saúde

Os atributos da classe devem incluir o nome, sobrenome, sexo, data de nascimento da pessoa (consistindo em atributos separados para o mês, dia e ano de nascimento), altura (em metros) e peso (em kg). Para cada atributo, forneça os métodos set e get. A classe também deve incluir métodos que calculam e retornam a idade do usuário em anos e o índice de massa corporal ($IMC = \text{peso} (\text{kg}) / \text{altura}^2 (\text{m})$). Escreva um aplicativo Java que solicite as informações da pessoa, instancia um objeto da classe PerfilDeSaude para essa pessoa e imprime as informações desse objeto - incluindo o nome, sobrenome, sexo, data de nascimento, altura e peso da pessoa - então calcule e imprime a idade da pessoa em anos e o IMC.

Obrigada pela sua
participação, nos vemos na
próxima aula! :)