

CC3642 - Orientação a Objetos

Prof. Danilo H. Perico
2019

Conceitos Básicos de Orientação a Objetos

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 (Informática)

- 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!

Exemplo: Diferença entre Classe e Objeto

- Classe:

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

- Objeto:

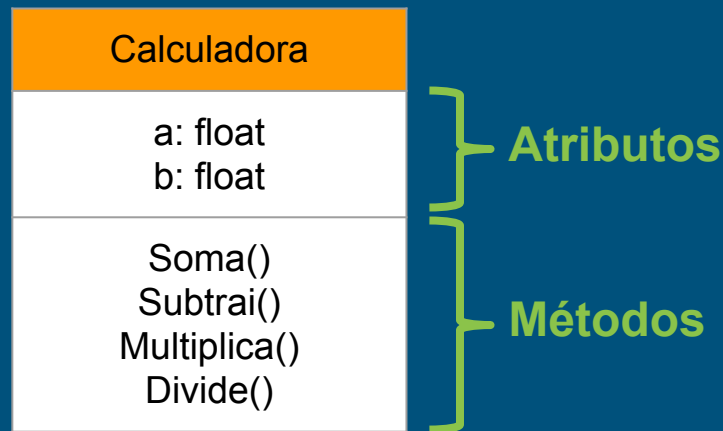
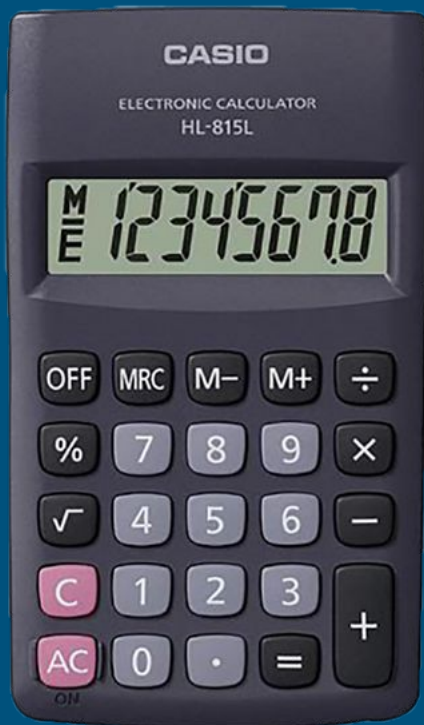
- É criado a partir da classe;
- É como se fosse a **própria casa** construída
- Pode-se construir várias casas a partir da mesma planta, assim como podemos instanciar vários objetos de uma só classe

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



Atributos

- Variáveis de instância (Java)
- 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

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;



Code Conventions

- *Class names should be nouns, in mixed case with the first letter of each internal word capitalized. Try to keep your class names simple and descriptive. Use whole words—avoid acronyms and abbreviations.*
- **Exemplos:**
 - `class Movie`
 - `class Calculadora`
 - `class ImageSprite`

Code Conventions

- *Methods* should be verbs, in mixed case with the first letter lowercase, with the first letter of each internal word capitalized.
- Exemplos:
 - `run();`
 - `runFast();`
 - `getBackground();`

Code Conventions

- All *instance* are in mixed case with a lowercase first letter. Internal words start with capital letters.
- Exemplos:
 - myGradeBook;
 - application;
 - testScope;

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

1. Declarando um Objeto:

- `<nome_da_classe> <nome_do_objeto>`
- `GradeBook myGradeBook`

2. Instanciando um Objeto:


- `myGradeBook = new GradeBook()`



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
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
```

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
4 public class GradeBookTest
5 {
6     // método main inicia a execução do 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
```

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

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

```

1 // Fig. 3.7: GradeBook.java
2 // classe GradeBook que contém uma variável de instância courseName
3 // e métodos para configurar e obter seu valor.
4
5 public class GradeBook
6 {
7     private String courseName; // nome do curso para esse GradeBook
8
9     // método para configurar o nome do curso
10    public void setCourseName( String name )
11    {
12        courseName = name; // armazena o nome do curso
13    } // termina o método setCourseName
14
15    // método para recuperar o nome do curso
16    public String getCourseName()
17    {
18        return courseName;
19    } // termina o método getCourseName
20
21    // exibe uma mensagem de boas-vindas para o usuário GradeBook
22    public void displayMessage()
23    {
24        // essa instrução chama getCourseName para obter o
25        // nome do curso que esse GradeBook representa
26        System.out.printf( "Welcome to the grade book for\n%s!\n",
27            getCourseName() );
28    } // termina o método displayMessage
29
30 } // fim da classe GradeBook

```

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)**

```

1 // Fig. 3.8: GradeBookTest.java
2 // Cria e manipula um objeto GradeBook.
3 import java.util.Scanner; // programa utiliza Scanner
4
5 public class GradeBookTest
6 {
7     // método main inicia a execução de programa
8     public static void main( String args[] )
9     {
10         // cria Scanner para obter entrada a partir da janela de comando
11         Scanner input = new Scanner( System.in );
12
13         // cria um objeto GradeBook e o atribui a myGradeBook
14         GradeBook myGradeBook = new GradeBook();
15
16         // exibe valor inicial de courseName
17         System.out.printf( "Initial course name is: %s\n\n",
18             myGradeBook.getCourseName() );
19
20         // solicita e lê o nome do curso
21         System.out.println( "Please enter the course name:" );
22         String theName = input.nextLine(); // lê uma linha de texto
23         myGradeBook.setCourseName( theName ); // configura o nome do curso
24         System.out.println(); // gera saída de uma linha em branco
25
26         // exibe mensagem de boas-vindas depois de especificar nome do curso
27         myGradeBook.displayMessage();
28     } // fim de main
29
30 } // fim da classe GradeBookTest

```

Exercício 8 - 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`
 - inicialize 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 9 - 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 (*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 utilizados os métodos *get*

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 com 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*

Computer programming: What is object oriented language?



Discussão sobre o conceito de Orientação a Objetos



Exercício 10 - classe Quadrado

- Atributos:
 - `lado`, *private* e do tipo `int`
- Métodos:
 - `set` e `get` para o atributo `lado`
 - `area`, que retorna a área do quadrado
 - `perimetro`, que retorna o perímetro do quadrado
 - `print`, para exibir `lado`, `area` e `perimetro`
- `main()`:
 - crie dois objetos (`q1` e `q2`) do tipo `Quadrado`
 - obtenha pelo teclado o valor de `lado` para `q1` e `q2`
 - inicialize os atributos de `q1` e `q2` com o método `set`
 - exiba o conteúdo dos atributos de `q1` e `q2` utilizando o método `print`
 - exiba o conteúdo dos atributos de `q1` e `q2` utilizando `get`

Exercício 11 - classe Data

(3.15 adaptado Deitel) - Crie uma classe chamada **Data** que inclui três partes de informações como variáveis de instância: um **mês** (tipo int), um **dia** (tipo int) e um **ano** (tipo int). Forneça um método **set** e um **get** para cada variável de instância. Forneça um método **exibeData** que exibe o mês, o dia e o ano separados por barras normais (/). Escreva um aplicativo chamado DataTeste, que demonstra as capacidades da classe **Data**.

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

Início
do
corpo
da
classe

Fim do
corpo
da
classe

```
1 // Fig. 3.1: fig03_01.cpp
2 // Define a classe GradeBook com uma função membro displayMessage;
3 // Cria um objeto GradeBook e chama sua função displayMessage.
4 #include <iostream>
5
6 using namespace std;
7
8 // Definição da classe GradeBook
9 class GradeBook
10 {
11 public:
12     // função que exibe uma mensagem de boas-vindas ao usuário do GradeBook
13     void displayMessage()
14     {
15         cout << "Welcome to the Grade Book!" << endl;
16     } // fim da função displayMessage
17 }; // fim da classe GradeBook
```

class inicia a
classe chamada
GradeBook

```

1// Fig. 3.1: fig03_01.cpp
2// Define a classe GradeBook com uma função membro displayMessage;
3// Cria um objeto GradeBook e chama sua função displayMessage.
4#include <iostream>
5
6using namespace std;
7
8// Definição da classe GradeBook
9class GradeBook
10{
11public:
12    // função que exibe uma mensagem de boas-vindas ao usuário do GradeBook
13    void displayMessage()
14    {
15        cout << "Welcome to the Grade Book!" << endl;
16    } // fim da função displayMessage
17}; // fim da classe GradeBook

```

C++

```

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

```

Java

Exemplo: Instanciando um Objeto da Classe

```
20 // a função main inicia a execução do programa
21 int main()
22 {
23     GradeBook myGradeBook; // cria um objeto GradeBook chamado myGradeBook
24     myGradeBook.displayMessage(); // chama a função displayMessage do objeto
25     return 0; // indica terminação bem-sucedida
26 } // fim de main
```

objeto
myGradeBook
criado

myGradeBook
chama o método
displayMessage()

Encapsulamento

Funções *set* e *get*

```
1 // Fig. 3.5: fig03_05.cpp
2 #include <iostream>
3 #include <string> // o programa utiliza classe de string padrão C++.
4
5 using namespace std;
6
7 // Definição da classe GradeBook
8 class GradeBook
9 {
10 public:
11     void setCourseName(string name)
12     {
13         courseName = name;
14     }
15
16     string getCourseName()
17     {
18         return courseName;
19     }
20
21     void displayMessage()
22     {
23         cout << "Welcome to the grade book for\n" << getCourseName() << "!"
24         << endl;
25     }
26
27 private:
28     string courseName;
29 };
```

Exercício 12 - classe Televisão

- Atributos:
 - `modelo` - *private* e do tipo *string*
 - `preco` e `tamanho` - *private* e do tipo *float*
 - `volume` e `canal` - *private int*
 - `ligada` - *private bool*
- Métodos:
 - *set* e *get* para todos atributos
 - `aumentaVolume` - aumenta 1 no valor atual de volume
 - `diminuiVolume` - diminui 1 no valor atual de volume
- `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`

Fim
