CEUB

EDUCAÇÃO SUPERIOR

Programação Orientada a Objetos

ceub.br

CEUB
EDUCAÇÃO SUPERIOR

PROGRAMAÇÃO ORIENTADA A OBJETOS

Aula 04 – Classe e Objetos em C#

Agenda

Classes
Objetos
Métodos
Atributos

Programação Orientada a Objetos

Elementos básicos

Objetos

Classes

Instâncias

Programação Orientada a Objetos

Elementos básicos - OBJETOS

Objetos são entidades concretas ou abstratas

- **Tem características e podem executar ações**
- **"Um objeto representa um item identificável, uma unidade ou entidade, individual, seja real ou abstrato, com uma regra bem definida" com uma regra bem definida"**
- **Possuem:**
  - **Estado**
  - **Comportamento (MÉTODOS)**
  - **Identidade**

Programação Orientada a Objetos

CLASSE

Modelo ou esquema a partir do qual os objetos são criados (instanciados).

Modelam os objetos definindo:

- **Tipo de dados que o objeto armazena, ou seja, os estados possíveis que ele pode assumir (atributos)**
- **Tipos de operações que podem ser executadas pelo objeto, ou seja, o seu comportamento (métodos)**

Abstração de objetos de características semelhantes (molde).É a essência do objeto.

Objetos são instâncias de classes.

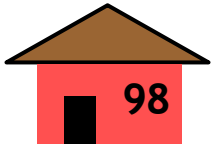
O que é uma classe?

- Classes são uma especificação para objetos
- Uma classe representa um tipo de dados (é uma estrutura de dados) complexo
- Classes descrevem
  - Tipos dos dados que compõem o objeto (o que podem armazenar)
  - Procedimentos que o objeto pode executar (o que podem fazer)

Classe

Instâncias da classe Casa (objetos)


```
Casa c1 = new Casa();
c1.numero = 1;
c1.cor = Color.yellow;
```




```
Casa c2 = new Casa();
c2.numero = 98;
c2.cor = Color.red;
```


```
Casa c3 = new Casa();
c3.numero = 25;
c3.cor = Color.white;
c3.abrePorta();
```

Classe

- A Classe é o molde, a planta, o esquema, o modelo a ser seguido pelos objetos
- A planta da casa é o modelo que as casas construídas terão
- Porém não é possível morar na planta da casa, apenas na casa já construída
- A Classe define as características da casa e as funções que ela terá: parte elétrica, hidráulica, saneamento e etc.

Síntese

Classes

- Em Orientação a Objetos, classes são a descrição de objetos do mundo real
- Descrevem atributos ou propriedades de objetos
- Descrevem métodos que operam sobre os atributos definidos.
- Quanto melhor a definição (mais completa), maior será a vida útil da classe e melhor será a definição do sistema em geral.
- A palavra-chave em POO: reuso

CEUB
EDUCAÇÃO SUPERIOR

CLASSES

Sintaxe básica de criação de classes em C#

Nomes de classes não podem ser exatamente iguais às palavras reservadas de C#.

• Caracteres maiúsculos e minúsculos são diferenciados em C#: as palavras Class, CLASS, ClAsS e class são consideradas como sendo diferentes, e somente a última pode ser usada para declarar uma classe.

Ex

```
public class NomeClasse {

}
```

```
class NomeClasse {

}
```

CEUB
EDUCAÇÃO SUPERIOR

CLASSES

Sintaxe básica de criação de classes em C#

Nomes de classes não podem ser exatamente iguais às palavras reservadas de C#.

• Caracteres maiúsculos e minúsculos são diferenciados em C#: as palavras Class, CLASS, ClAsS e class são consideradas como sendo diferentes, e somente a última pode ser usada para declarar uma classe.

Ex

```
public class Lampada {

}
```

```
class Lampada {

}
```

INSTANCIA

INSTANCIA É SINONIMO DE UM OBJETO;

O ATO DE INSTANCIAR UMA CLASSE É O ATO DE CRIAR UM OBJETO.

Em C#, uma instância de uma classe é um novo objeto criado dessa classe, com o operador new.

Instanciar uma classe é criar um novo objeto do mesmo tipo dessa classe.

Uma classe somente poderá ser utilizada após ser instanciada.

Objeto

- Objetos são utilizados para representar conceitos do mundo real
- Objetos seguem fielmente as especificações de suas Classes
- Os Objetos são instâncias concretas das Classes
- As casas são instâncias concretas das plantas que lhes deram origem

Criando um Objeto

- Operador *new* cria um novo objeto a partir de uma classe especificada (cria uma instância)
- Retorna uma referência para esse objeto

new <tipo_classe> ([parametro, parametro, ...]);

[ ] = Opcionais
< > = Identificadores e palavras reservadas

Criando um Objeto

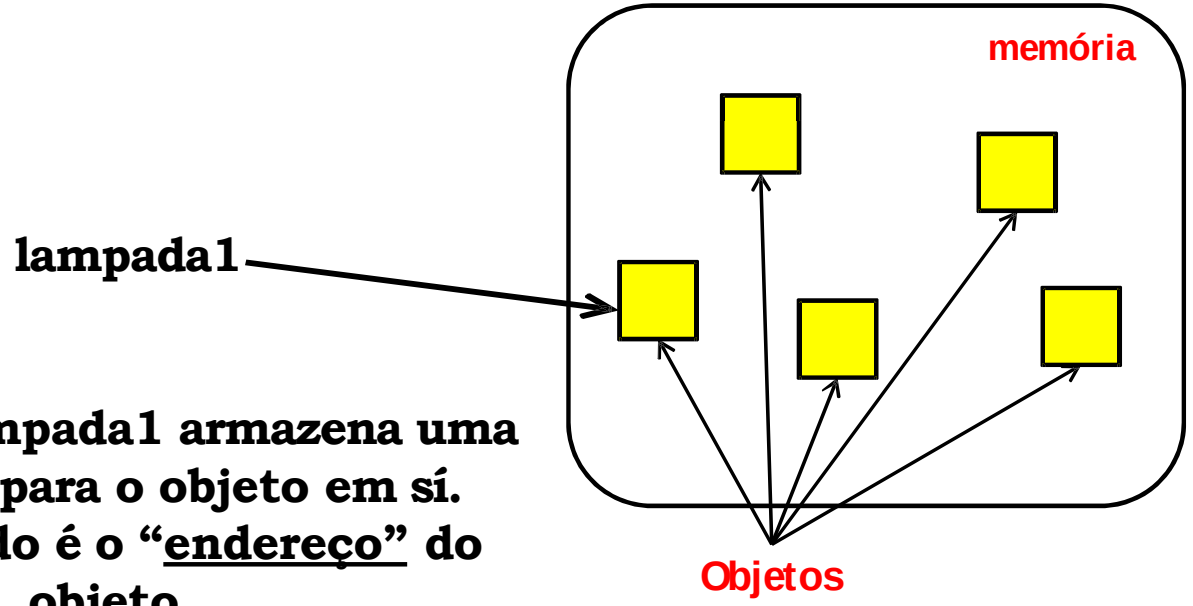
- **Passos:**
  - **Declarar variável, associando variável a tipo (classe):**
    - **NomeClasse nomeVariável;**
    - **Ex.: Lampada lampada1;**
  - **Criar objeto (instanciar) e fazer variável referenciar o objeto:**
    - **Ex.: lampada1 = new Lampada();**
    - **Ex2.:Lampada lampada1 = new Lampada();**

Criando um Objeto

- Ao utilizar o operador **new**:
  - Novo objeto é alocado dinamicamente na memória, e todas as suas variáveis de instancia são inicializadas com valores-padrão predefinidos.
    - null para variáveis objeto
    - 0 para todos os tipos básicos (exceto boolean)
    - false para boolean
  - O construtor do novo objeto é ativado
  - Após a execução do construtor, o operador new retorna uma referência (endereço de memória) para o objeto recém criado.

Criando um Objeto

```
Lampada lampada1 = new Lampada();
```



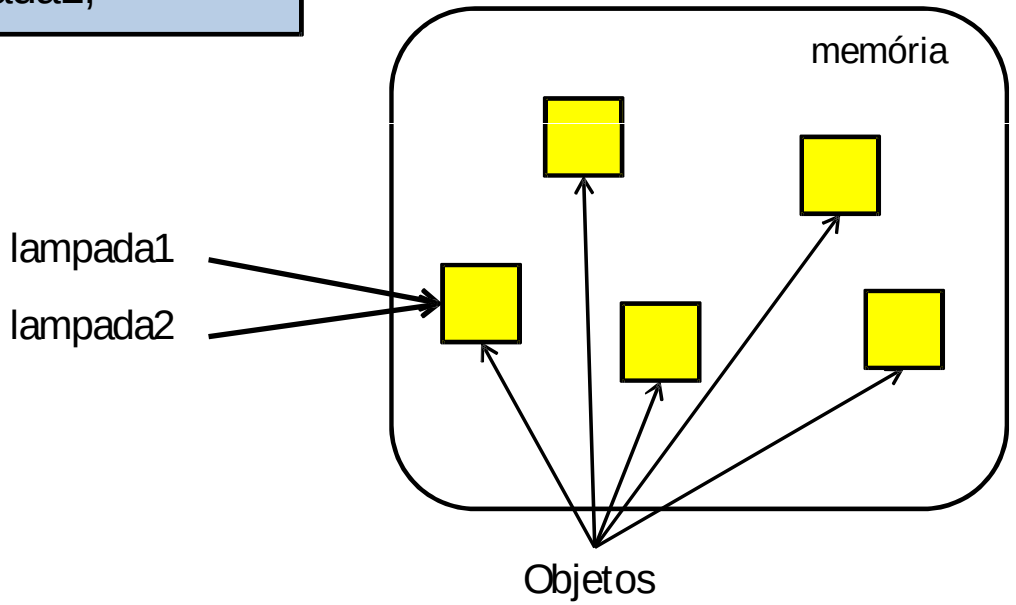
A variável lampada1 armazena uma referência para o objeto em sí. Seu conteúdo é o "endereço" do objeto

Criando um Objeto

- **Observações:**
  - **Lampada lampada1;**
    - **Declaração da lampada1 como referência para objeto da classe Lampada**
  - **lampada1 = new Lampada();**
    - **Cria objeto e faz lampada1 referenciar o objeto recém-criado**

Criando um Objeto

```
Lampada lampada1, lampada2;
lampada1 = new Lampada();
lampada2 = lampada1;
```



CLASSES

Sintaxe básica de criação de classes em C#

Uma classe em C# é sempre declarada com a palavra-chave class seguida do nome da classe.

O nome da classe não pode conter espaços, deve sempre ser iniciado por uma letra.

Para nomes de classes, métodos e campos em C#, o caractere sublinhado (_) e o sinal $ são considerados letras.

O nome da classe não deve conter acentos1
e pode conter números, contanto que estes apareçam depois de uma letra.

CEUB
EDUCAÇÃO SUPERIOR

CLASSES

Sintaxe básica de criação de objetos em C#

Para criar (construir, instanciar) uma classe, basta usar a palavra chave new.

Ex

```
class NomeClasse {
 public static void main(String[] args) {
      NomeClasse meuObjeto;
      meuObjeto = new NomeClasse();

      ou

      NomeClasse meuObjeto = new NomeClasse();

  }
}
```

CEUB
EDUCAÇÃO SUPERIOR

CLASSES

Sintaxe básica de criação de objetos em C#

Para criar (construir, instanciar) uma classe, basta usar a palavra chave new.

Ex

```
class NomeClassePrincipal {
 public static void main(String[] args) {
      NomeClasse meuObjeto;
      meuObjeto = new NomeClasse();

      ou

      NomeClasse meuObjeto = new NomeClasse();

  }
}
```

Atributos

- São as variáveis de instância
  - Fazem parte de cada objeto (instância)
- Declarada fora dos métodos
- "Vivem" enquanto o objeto "viver“
- Obs: Todo objeto possui um identificador chamado this, que é uma referência para o próprio objeto.

Atributos

[<modificadores_atributo>] <tipo_atributo> <nome_atributo> [= valor_inicial];

[ ] = Opcionais
< > = Identificadores e palavras reservadas

EXEMPLO

```
public boolean estadoLampada = false;
Double valor;
String marca = “fluorescente”;
```

Classes e Objetos

- **Como definir uma classe e seus atributos**

```
public class Cliente
{
    private int clienteId;
    private string nome;
    private decimal limiteCredito;
    private int quantidadeProdutos;
}
```

Como criar uma instância de uma classe

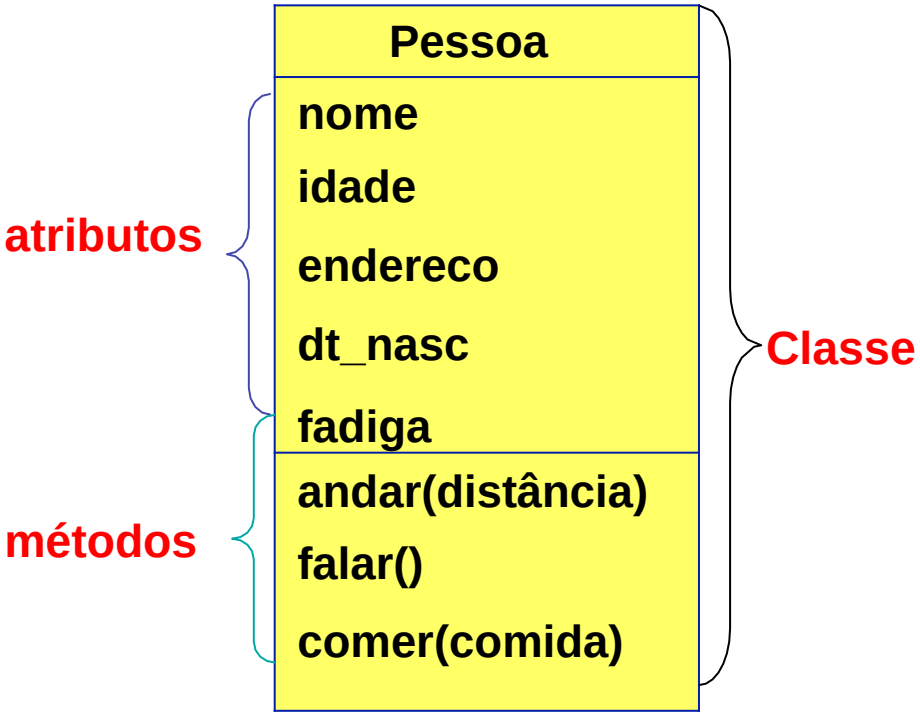
```
Cliente novoCliente = new Cliente();
```

this

- Todo objeto possui um atributo que é uma referência a ele mesmo.
  - É acessado para acesso a membros do próprio objeto
  - this.membro
  - Evita conflito Com parâmetros de métodos, por exemplo

```
class NomeClasse {
    int x, y;
        public void mover(int x,int y){
            this.x = x;
            this.y = y;
        }
}
```

UML



- A classe é composta por atributos (propriedades) e métodos (ações / mensagens).
- Os atributos possuem nomes significativos que melhor definem o dado de negócio que representam.
- Os métodos são sempre ações que operam sobre os atributos da classe.
- Em uma boa programação OO, somente os métodos de objeto podem alterar o estado do objeto (atributos).
- Os métodos recebem (ou não) parâmetros que permitem customizar uma ação sobre os atributos.

Exercícios

1)Escreva as classes (Cliente, Locacao e Carro) do sistema de informação que gerencie o aluguel de uma frota de carros. Para cada uma das classes abaixo criar os atributos das classes conforme listadas abaixo:

Carro (idCarro, placa, fabricante, modelo, ano, cor, valorDiaria)
Cliente (idCliente, cpf, nome, cnh)
Locacao (idLocacao, idCarro, idCliente, valorLocado, dataInicio, dataFim)

2)Crie uma classe principal com nome (AlugaFacil) instancie dois objetos do tipo Cliente e dois objetos do tipo Carro. Na sequencia atribua valores aos objetos criados e os imprima.

3)Criar uma classe conta corrente com os seguintes atributos (numeroConta; nomeBanco; nomeAgencia, nomeCliente; saldo;

4)Criar uma classe principal gerenciamento financeiro e instancias a classe conta corrente e atribuir valores a seus atributos. Imprimir esses valores na tela.

Sistemas Orientados a Objetos

Como surgem os sistemas?

- Através de um problema identificado e descrito

Como descrever um problema identificado?

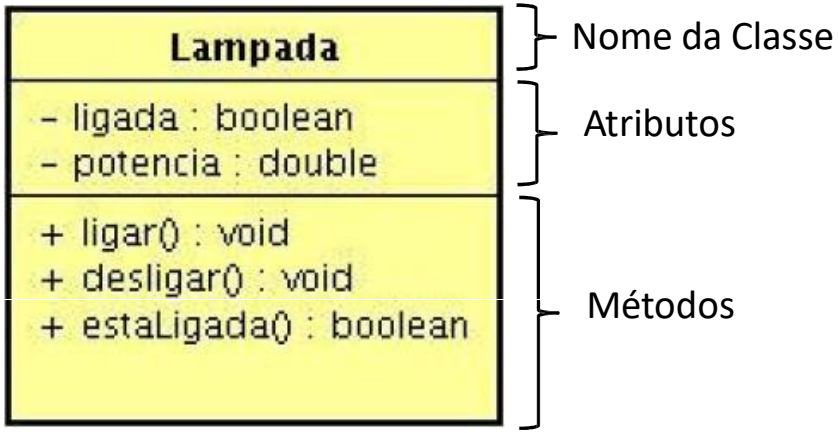
- Identificar o motivo ou objetivo
- Descrever o que o sistema deve fazer

Como conceber um sistema?

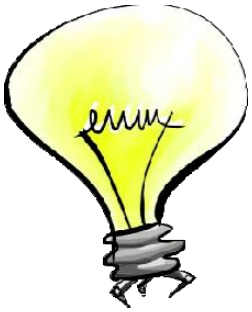
- A partir do que deve ser feito, identificar quais são as entidades que compõem este sistema.
- Descrever quais são os atributos destes objetos
- Descrever quais são as ações dos objetos e quais são as interações com outros objetos.

Classe

- Classe:



- *Classe Lampada*
  - *Atributos*
    - *potencia (double), ligada (boolean)*
  - *Operações*
    - *ligar, desligar, estaLigada*



Métodos

- Definem o comportamento de uma classe
- Podem ser utilizados para:
  - realizar algum trabalho dentro da classe
  - modificar o valor de algum atributo
  - resgatar o valor de um atributo
  - ativar ações em outros objetos
  - enviar dados pela rede
  - iniciar eventos de interface gráfica
  - iniciar sons
  - outras ações

Métodos em C#

Sintaxe

```
[<modificadores_método>] <tipo_retorno> <nome_método> ([<parametros>]){

        // Corpo do Método

}
```

[ ] = Opcionais
< > = Identificadores e palavras reservadas

Métodos em C#

- Passagem de parâmetros:
  - Deve ser informados o tipo e identificador dos parâmetros
  - Funciona no método como uma variável normal
  - Passam o valor do identificador

  Ex:

```
void sacar(double valorSacado){
        valor-=valorSacado;
}

void depositar(double valorDepositado){
        valor+=valorDepositado;
}
```

Métodos em C#

- Usamos o operador "." (ponto) para acessar um método
  - Sintaxe:
    - objeto.método();
  - Executa método em objeto
    - Objeto deve existir
  - A variável deve referenciar objeto válido
    - Se referenciar null ocorre erro
  - Exemplos:
    - obj1.nomeMetodo();
    - obj1.nomeMetodo(arg1, arg2);
    - (new NomeClasse()).nomeMetodo();
    - obj1.nomeAtributo;

Corpo do Método

- **Corpo do método:**
  - **Implementa as operações do método**
  - **Fica entre chaves ({})**
  - **Variáveis podem ser criadas**
    - **Ela é dita local**
    - **Não é pré-inicializada**
    - **Só existe enquanto o método está em execução**

Classe Completa

```
public class Lampada
{
public boolean estadoLampada;

        public Lampada(){
                estadoLampada = false;
        }

        public void acenderLampada() {
                estadoLampada = true;
        }

        public void apagarLampada() {
                estadoLampada = false;
        }

        public boolean verEstadoLampada() {
                return estadoLampada;
        }
}
```

Atributos

Construtor

Métodos

Método

- **É possível que uma Classe possua 2 métodos com o mesmo nome?**
- **Sim, é possível, mas devem ter parâmetros diferentes (quantidade e tipo)!!!**
  O nome que se dá a isso é sobrecarga ou clonagem!

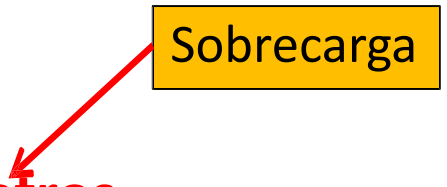
Exemplo:

```
double calcularMedia(double nota1, double nota2){
        return (nota1+nota2)/2;
}

double calcularMedia(double nota1, double nota2, int peso1, int peso2){
        return (nota1*peso1+nota2*peso2)/peso1+peso2;
}
```

Construtor

- Uma classe pode conter vários construtores
  - Mesmo nome da classe
  - Não possui retorno
  - Diferença na quantidade e tipo dos parâmetros



- Construtor padrão é fornecido
  - Se não houver pelo menos um definido
  - Não possui parâmetros

- É chamado na execução do new

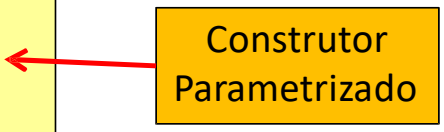
Construtor

Classe com 2 construtores

```
public class Pessoa {
        String nome;
        int rg, cpf;

        public Pessoa(){
                nome="";
                rg=0;
                cpf=0;
        }

        public Pessoa(String nome, int rg, int cpf) {
                this.nome = nome;
                this.rg = rg;
                this.cpf = cpf;
        }
}
```



Construtor

- Criando objeto de uma classe com 2 construtores:

```
Pessoa obj = new Pessoa();
Pessoa obj2 = new Pessoa("MARIA", 1111, 333333333);
```

Métodos

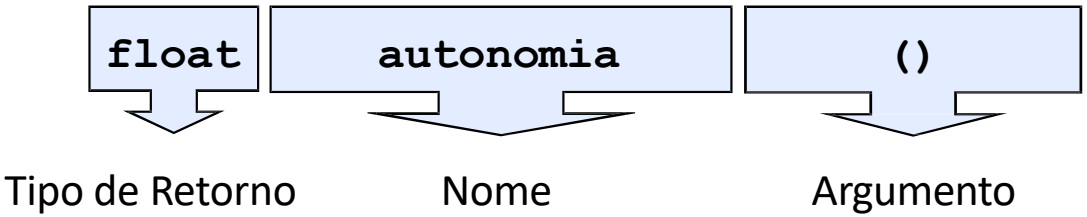
- **Definem o comportamento da classe;**
- **Possuem sintaxe semelhante à sintaxe de definição das funções de um programa procedural;**
- **Determinam o comportamento da classe e a troca de mensagens com outras classes.**

DECLARAÇÃO

```
class Carro {
    Stringfabricante,  cor;
    int capacidadeTanque;
    float consumo;
    public float autonomia ( ){
      return capacidadeTanque * consumo;
    }
}
```

Métodos

ASSINATURA



A PALAVRA-CHAVE RETURN

- A palavra-chave **return** especifica o que será retornado após a chamada a um método. Se o método for **void**, não haverá o uso do **return**.

```
boolean método() {
    if (condição) {
        instrução;
        return true;
    }
    resto do método
    return false;
}
```

Chamadas de Métodos

- A troca de mensagens entre os objetos é realizada através da chamada de métodos.

EXEMPLO

```
class Aplicacao
{
   public static void main (String args[]) {
      Carro car1 = new Carro();
      … Console.Writlen(car1.autonomia());
   }
}
```

Chamada do método

Parâmetros

- Parâmetros são utilizados para passar valores para métodos
- São utilizados em casos em que o método precisa de um valor externo para realizar o seu trabalho
- Os parâmetros são passados entre parênteses logo após o nome do método
- Cada parâmetro tem um nome e um tipo

Visibilidade

- Definem quem pode visualizar atributos e métodos
- Modificadores de visibilidade do C#:
  - public
  - private
  - protected
  - “default”

Acessando um Método

- Utiliza o operador “.” (ponto):
  - objeto.método();
  - Ex.: lampada.acender();
- Executa método em objeto
- Objeto deve existir
  - A variável deve referenciar objeto válido (se null, ocorre erro)
- Exemplos:
  - obj1.nomeMetodo();
  - obj1.nomeMetodo(arg1, arg2);
  - (new NomeClasse()).nomeMetodo();

Acessando um Método

```
public class Lampada {

    public boolean estadoLampada = false;

    public void acenderLampada() {
            estadoLampada = true;
    }

    public void apagarLampada() {
            estadoLampada = false;
    }

    public boolean verEstadoLampada() {
            return estadoLampada;
    }
}
```

Classe Lampada

Classe TesteLampada

```
public class TesteLampada {
    public static void main(String[] args) {
            Lampada lampada1 = new Lampada();
            boolean valorAtual = ampada1.verEstadoLampada());
            Console.Writlen(valorAtual);
            lampada1.acenderLampada();
            Console.Writlen(lampada1.verEstadoLampada(
            )); lampada1.apagarLampada();
            Console.Writlen(lampada1.verEstadoLampada(
    }       ));
}
```

```
false
true
false
```

Saída da Classe TesteLampada

Exemplo OO C# Criação de Classe Funcionário

```
using System;
using System.Collections.Generic;
using System.Text;

namespace ExemploOO.BLL
{
    class Funcionario
    {
        private string _nome;
        private int _idade;
        private string _sexo;
        private double _salario;

        public string Nome { get => _nome; set => _nome = value; }
        public int Idade { get => _idade; set => _idade = value; }
        public string Sexo { get => _sexo; set => _sexo = value; }
        public double Salario { get => _salario; set => _salario = value; }

        //MÉTODOS
        public double calculaDecimoTerceiro(int mesesTrabalhos)
        {
            double decimoTerceiro;
            decimoTerceiro = _salario * mesesTrabalhos / 12;
            return decimoTerceiro;
        }
        public void calculaFerias()
        {
            double ferias;
            ferias = _salario + _salario / 3;
            Console.WriteLine("O SALARIO DE FERIAS É R$: " + ferias + " Reais");
        }

    }
}
```

Exemplo OO C# Aplicação Console

```
using ExemploOO.BLL;
using System;

namespace ExemploOO
{
    class Program
    {
        static void Main(string[] args)
        {
            Funcionario funcionario = new Funcionario();

            Console.WriteLine("Digite o nome do funcionário");
            funcionario.Nome = Console.ReadLine();
            Console.WriteLine("Digite a idade do funcionário");
            funcionario.Idade = int.Parse(Console.ReadLine());
            Console.WriteLine("Digite o sexo do funcionário");
            funcionario.Sexo = Console.ReadLine();
            Console.WriteLine("Digite o salário do funcionário");
            funcionario.Salario = double.Parse(Console.ReadLine());

            Console.WriteLine("O funcionário {0} possui {1} anos,  ele é do sexo {2} e seu salário é R$ {3}", funcionario.Nome, funcionario.Idade, funcionario.Sexo, funcionario.Salario);

            Console.WriteLine("O funcionário {0} estará de férias no período de 16/09/2020 à 14/09/2020", funcionario.Nome);
            funcionario.calculaFerias();

            Console.WriteLine("O funcionário {0} receberá R$ {1} Reais referente ao seu décimo terceiro", funcionario.Nome, funcionario.calculaDecimoTerceiro(12));
        }
    }
}
```

Exercício 1

- Criar uma classe Carro com os seguintes atributos, marca, modelo, placa, cor, numero de marchas, ano de fabricação, ano modelo, velocidade, ligar e desligar; A classe carro deverá ter os seguintes métodos :
- buzinar (Ao ser acionado deverá mostrar uma mensagem "BIIIIIIIIII!)
- acelerar,
- ligar (Ao ser invocado deverá verificar se o carro está ligado ou não. Se estiver ligado deverá imprimir uma mensagem carro já está ligado, caso contrário deverá mostrar uma mensagem o carro foi ligado)
- desligar(Ao ser invocado deverá verificar se o carro está desligado ou não. Se estiver ligado deverá imprimir uma mensagem carro já está ligado, caso contrário deverá mostrar uma mensagem o carro foi desligado)

Exercício 2

Criar uma classe Pessoa com os seguintes atributos: Nome, peso, altura.

- **A classe pessoa deverá ter um método para calcular o IMC – Índice de massa corporal da pessoa. O índice é obtido pela formula = peso/altura*altura;**
- **Deverá também ter um método que retorne a classificação do IMC conforme descrito a seguir:**
- **Se IMC <=19 categoria será "ABAIXO DO PESO"**
- **Se IMC <=25 categoria será "PESO IDEAL"**
- **Se IMC <=30 categoria será "ACIMA DO PESO"**
- **Se IMC <=35 categoria será "OBESIDADE LEVE" OU**
- **ACIMA DE 35 categoria será "OBESIDADE"**

Exercício 3

- **Criar uma nova classe que:**
  - **Vai conter o método main do C#**
  - **Dentro do main, instancia dois objetos da classe pessoa e solicitar que o usuário digite o nome da pessoa, peso e altura.**
  - **Na sequencia o sistema deverá mostra o nome da pessoa seu IMC e em qual categoria do IMC ele está categorizado.**

Exercício 4

Criar uma classe correspondente a uma Conta de Banco

- **A classe terá como atributos nome, numero da conta, saldo**

- **Terá os métodos sacar, depositar, consultar saldo, consultar nome, alterar nome onde:**

  - **Sacar -> Diminui o valor sacado do valor do saldo**
  - **Depositar -> Soma o valor depositado com o valor do saldo**
  - **Consultar Saldo -> Retorna o valor do salto atual**
  - **Consultar Nome -> Retorna o nome atual**
  - **Alterar Nome -> Altera o nome cadastrado**

- **Saldo inicial será de 100 reais.**

Exercício 5

- **Crie uma classe Calculadora, onde a mesma terá 4 métodos: somar, subtrair, dividir e multiplicar.**
  - **Todos os métodos recebem 2 valores reais como parâmetros, e retornam o resultado da operação**
- **Crie outra classe, com o método main, para testar a Calculadora.**
  - **Crie um objeto calculadora, e realize as 4 operações acessando os métodos oferecidas por ela.**

Exercício 6

- Crie uma classe Pessoa. Nela terá os atributos nome, idade, cpf.
  - Crie um construtor parametrizado inicializando todas as variáveis com os valores recebidos dos parâmetros.
  - Crie um construtor default (Inicializando as variáveis da classe com valores padrões).
  - Crie um método para receber os 3 valores dos atributos da classe Pessoa e alterá-los.
- Crie outra classe, com o método main, para testar a classe Pessoa:
  - Nela, crie 2 objetos da classe Pessoa. Um dos objetos criados deve inicializar as variáveis pelo construtor. O segundo objeto deve usar o construtor default para criar o objeto, e mudar os valores de Pessoa acessando o método de alterar