1. **IMPORTAÇÃO DAS BIBLIOTECAS.**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

1. **DEFINIÇÃO DA CLASSE ”corpoCeleste”**

public class corpoCeleste

Esta é uma **classe base** que representa qualquer corpo celeste (planetas, asteroides, luas, etc.).

**Encapsula informações comuns** a todos os corpos celestes, como:

* Nome (n)
* Massa (m)
* Raio (r)

1. **ATRIBUTOS DA CLASSE ”corpoCeleste”**

protected string n;

protected double m;

protected double r;

Os atributos são **protected**, ou seja, podem ser acessados apenas pelas **subclasses** (exemplo: planeta e asteroide).

**Definem as propriedades básicas** de um corpo celeste.

1. **CONSTRUTOR DA CLASSE**

public corpoCeleste() { }

1. **MÉTODO ”setDados”**

public virtual void SetDados(string nome, double massa, double raio)

{

this.n = nome;

this.m = massa;

this.r = raio;

}

**Objetivo:** definir os dados básicos do corpo celeste.  
**Uso do this**: diferencia a variável de instância (n, m, r) dos parâmetros do método.  
**Palavra-chave virtual**: permite que esse método seja sobrescrito (override) nas subclasses.

1. **MÉTODO ”exibir\_informações”**

public virtual void exibir\_informações()

{

Console.WriteLine($" -Corpo Celeste: {n}");

Console.WriteLine($" - Massa: {m} kg");

Console.WriteLine($" - Raio: {r} m");

}

**Objetivo:** Exibir as informações do corpo celeste no console.  
**Uso do $ (interpolação de string)**: facilita a exibição dos valores.  
**Uso de virtual**: permite que subclasses alterem esse método.

1. **MÉTODOS PARA OBTER A MASSA E O NOME**

public double getmassa()

{

return m;

}

public string getnome()

{

return n;

}

**Objetivo:** Permitir acesso à massa e ao nome do corpo celeste de forma **controlada**.

1. **DEFINIÇÃO DA CLASSE ”planeta”**

public class planeta : corpoCeleste

A classe planeta **herda** de corpoCeleste, ou seja, recebe seus atributos e métodos. Essa é uma aplicação de **herança** da Programação Orientada a Objetos (POO).

1. **CONSTRUTOR DA CLASSE**

public planeta() { }

1. **Método SetDados (Sobrescrito)**

public override void SetDados(string nome, double massa, double raio)

{

base.SetDados(nome, massa, raio);

}

**Usa override para sobrescrever o método da classe base**.  
b**ase.SetDados(...) chama o método da classe corpoCeleste**, garantindo que o planeta armazene suas informações corretamente.

1. **MÉTODO PARA CALCULAR A GRAVIDADE**

public double GetGravidade()

{

double G = 6.674e-11;

return (G \* m) / (r \* r);

}

Retorna o valor da **gravidade na superfície do planeta**.

1. **EXIBIR INFORMAÇÕES DO PLANETA**

public override void exibir\_informações()

{

base.exibir\_informações();

Console.WriteLine($" - Gravidade: {GetGravidade():F2} m/s²");

}

Chama exibir\_informações() da classe base e **adiciona a gravidade**.  
F2 formata a saída com **duas casas decimais**.

1. **Definição da Classe ”asteroide”**

public class asteroide : corpoCeleste

Herda de corpoCeleste, assim como planeta.  
Adiciona novas propriedades específicas de asteroides, como:

* semiEixoMaior
* excentricidade
* orbitaEmTornoDe (referência ao planeta que ele orbita).

1. **CONSTRUTOR DA CLASSE ”asteroide”**

public asteroide(string nome, double massa, double raio, double semiEixoMaior, double excentricidade)

{

SetDados(nome, massa, raio);

this.semiEixoMaior = semiEixoMaior;

this.excentricidade = excentricidade;

}

Inicializa os valores do asteroide **e chama SetDados()** para armazenar os atributos herdados.

1. **MÉTODO PARA CALCULAR O PERÍODO ORBITAL**

public double GetPeriodoOrbital()

{

if (orbitaEmTornoDe == null)

{

throw new InvalidOperationException("O asteroide não está orbitando nenhum planeta!");

}

double G = 6.674e-11;

double M = orbitaEmTornoDe.getmassa();

return 2 \* Math.PI \* Math.Sqrt(Math.Pow(semiEixoMaior, 3) / (G \* M));

}

A**plica a Terceira Lei de Kepler** para calcular o tempo que o asteroide leva para orbitar um planeta.  
**Verifica se o asteroide tem um planeta associado** (orbitaEmTornoDe != null).

1. **MÉTODO PARA EXIBIR INFORMAÇÕES**

public override void exibir\_informações()

{

base.exibir\_informações();

Console.WriteLine($" - Orbita ao redor de: {(orbitaEmTornoDe != null ? orbitaEmTornoDe.getnome() : "Desconhecido")}");

Console.WriteLine($" - Semi-eixo maior: {semiEixoMaior:F0} m");

Console.WriteLine($" - Excentricidade: {excentricidade:F2}");

if (orbitaEmTornoDe != null)

{

Console.WriteLine($" - Período orbital: {GetPeriodoOrbital() / (60 \* 60 \* 24):F2} dias");

}

}

Exibe **os parâmetros da órbita** do asteroide.

Se houver um planeta de órbita definido, exibe **o período orbital**.

1. **PROGRAMA PRINCIPAL**

static void Main(string[] args)

Função principal do programa, onde os objetos são criados e testados.

1. **CRIANDO UM PLANETA (TERRA)**

planeta planeta = new planeta();

planeta.SetDados("Terra", 5.972e24, 6.371e6);

planeta.exibir\_informações();

**Cria e define a Terra**, exibindo suas informações.

**CRIANDO UM ASTEROIDE (CERES)**

asteroide ceres = new asteroide("Ceres", 9.393e20, 469730, 2.77e11, 0.08);

ceres.SetOrbita(planeta);

ceres.exibir\_informações();

Cria Ceres, define sua órbita ao redor da Terra e exibe suas informações.

1. **FINALIZAÇÃO**

Console.ReadKey();

Aguarda o usuário pressionar uma tecla para **manter o console aberto**.