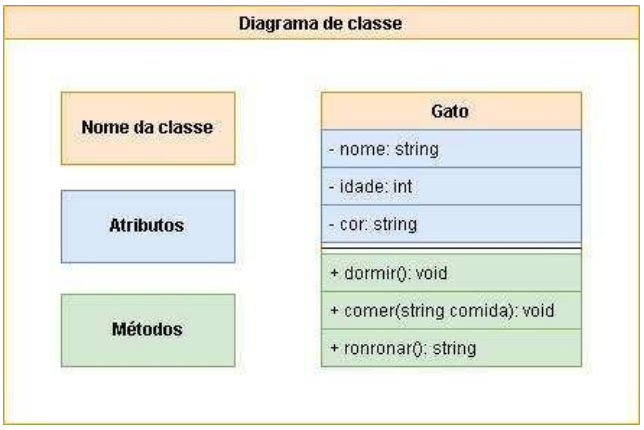
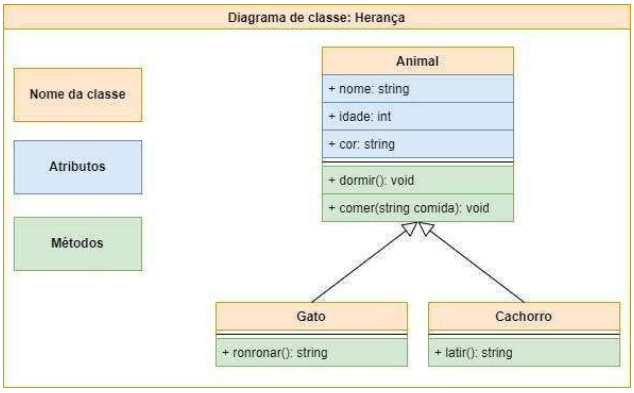
1. 

A imagem representa:

A Uma classe em UML

B Uma classe abstrata em UML

C Uma interface em UML

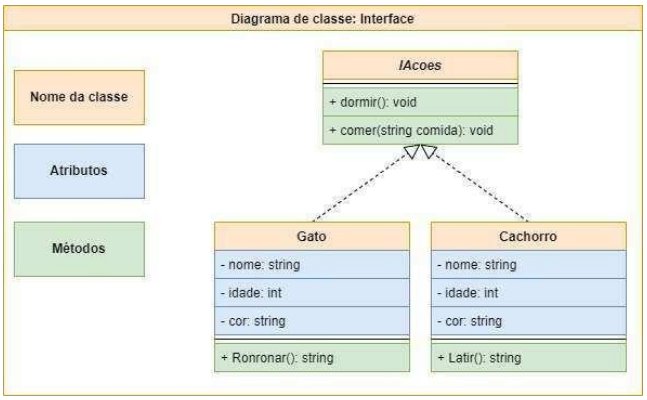
2. 

Na imagem podemos ver:

A Um exemplo de Polimorfismo

B Um exemplo de Interface

C Um exemplo de Herança

3. 

Na imagem é possível ver:

A Todas estão corretas

B Um exemplo de Interface

C Um exemplo de Herança

4. A herança é realizada usando uma derivação, o que significa que uma classe é declarada usando uma classe base, da qual ela herda o comportamento e os dados.

A Falso

B Verdadeiro

5. A classe Object, é considerada uma herança implícita, e todas as classes possuem uma herança implícita com a classe Object que por si possui 8 membros (métodos) em sua especificação!.

A Falso

B Verdadeiro

6. Uma interface não pode conter qualquer tipo de código, muito menos código padrão. Uma classe abstrata pode fornecer código completo , código padrão ou ter apenas a declaração de seu esqueleto para ser posteriormente sobrescrita.

A Verdadeiro

B Falso

7.
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace Zoologico.src
{
    2 references
    public class Gato : Animal
    {
        0 references
        public Gato() { }

        2 references
        public Gato(string nome, string cor, string classificacao) : base(nome, cor, classificacao) { }

        3 references
        public override void Comunicar(string comunicacao)
        {
            Console.WriteLine($"{Nome} está falando: {comunicacao}");
        }
    }
}
```

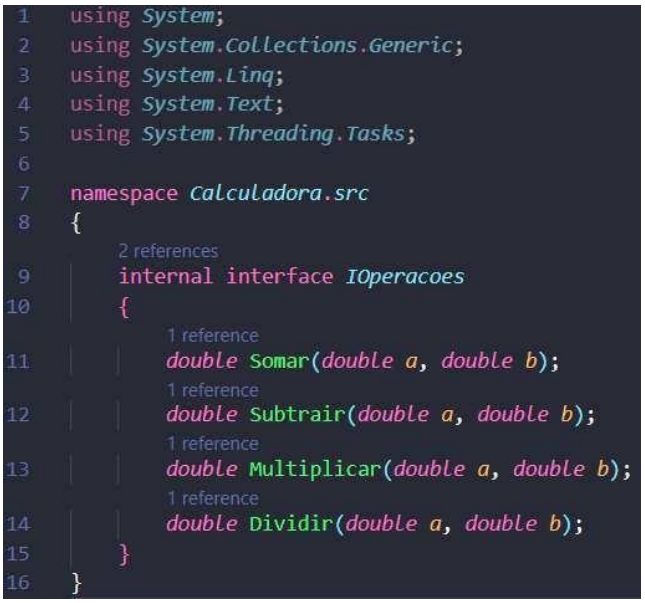
A Na linha 12, esta definido um construtor de Gato, que define o construtor da classe Animal

B Na linha 10 o método construtor da classe Gato recebe parâmetros que definem a classe Animal

C Na linha 14 esta definido um método que sobrescreve o método Comunicar da classe Animal

D Na linha 14 o método pertence a palavra override informa que o método esta sobrescrevendo

8.



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Calculadora.src
{
    2 references
    internal interface IOperacoes
    {
        1 reference
        double Somar(double a, double b);
        1 reference
        double Subtrair(double a, double b);
        1 reference
        double Multiplicar(double a, double b);
        1 reference
        double Dividir(double a, double b);
    }
}
```

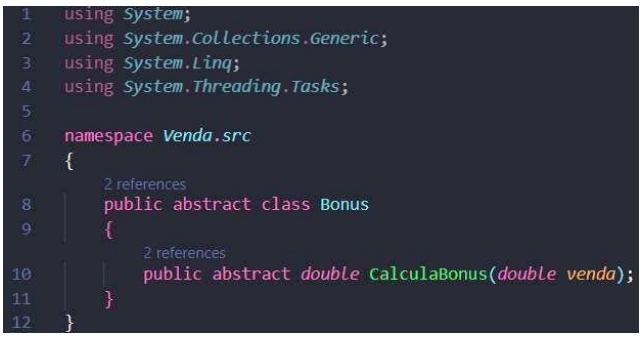
IOperacoes, é uma:

✗ Uma interface, que possui apenas a assinatura dos métodos

B Classe abstrata, que possui apenas a assinatura dos métodos

C Classe normal, que possui apenas a assinatura dos métodos

9.



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace Venda.src
{
    2 references
    public abstract class Bonus
    {
        2 references
        public abstract double CalculaBonus(double venda);
    }
}
```

Em uma classe abstrata é correto afirmar que:

✗ Os métodos podem ser de assinatura e implementados

B Os métodos podem ser somente de assinatura

C Os métodos podem ser somente implementados

10. Para quais casos é possível utilizar uma classe abstrata e uma interface, respectivamente.

✗ Para casos onde é definido um conjunto de assinatura de métodos que outras classes devem implementar e casos que precisem se preocupar com o comportamento padrão

B Para casos que precisem se preocupar com o comportamento padrão e casos onde é definido um conjunto de assinatura de métodos que outras classes devem implementar

C Para quaisquer casos, na realidade não é importante o conceito de abstrato e interface