

UNIFAI - Centro Universitário de Adamantina

Ciência da Computação – POO II - Prof. Carlos Koyama - 2ª Prova bimestral

- 1) A POO não é apenas uma forma de programar, é também um jeito de pensar em um problema utilizando conceitos do mundo real. **Considerando os conceitos da POO**, analise as afirmações:
- I. O objeto tem determinadas propriedades que o caracterizam e que são armazenadas no próprio objeto. A herança permite o reaproveitamento dos atributos da classe pai nas classes filha.
 - II. A herança é um mecanismo para o compartilhamento de métodos e atributos entre as classes e subclasses, permitindo a criação de novas classes através da programação das diferenças entre a nova classe e a classe-pai.
 - III. Uma classe abstrata pode conter métodos abstratos com ou sem implementação
 - IV. A definição de comportamentos diferenciados de um determinado método em subclasses pode ser resolvida definindo-se um novo método (com um nome diferente daquele definido na classe pai)

É INCORRETO o que se afirma em: (1,0)

- a) I e III
- b) I e IV
- c) II e III
- X d) III e IV**
- e) II e IV

- 2) **Comente e justifique** as afirmações incorretas, demonstre e explique através de um exemplo de código uma situação correta para o erro da afirmação (2,0)

Resposta:

Item III - métodos abstratos não podem ter corpo de implementações, e sim só sua assinatura, seu corpo deverá ser codificado somente na classe derivada.

```
-referências
abstract class Ex2 {
    -referências
    public abstract void exemplo();
}

-Referências
class Exemplo2: Ex2
{
    -referências
    public override void exemplo()
    {
    }
}
```

Item IV - Não utiliza de forma correta os conceitos de POO, a forma correta de tratar a diferença entre um método entre uma classe pai e filha, é o sobrescrevendo com o override, assim o comportamento padrão irá se manter na classe pai, mas será alterado apenas nessa classe filha, e para isso colocamos também o método da classe pai como virtual

```
class Ex2 {
    -referências
    public virtual void exemplo()
    {
        Console.WriteLine("Bom Dia");
    }
}

-Referências
class Exemplo2: Ex2
{
    -referências
    public override void exemplo()
    {
        //modificando e aplicando diferenças
        Console.WriteLine("Boa noite");
    }
}
```

3) Seja o código abaixo:

```
public class Moto  
{  
    string modelo, fabricante;  
  
    int quilometragem;  
  
    public double TransformaKmEmMilhas( )  
    {  
        return quilometragem / 1.609;  
    }  
}  
  
public class MotoTriciclo : Moto  
{  
  
    public float TransformaKmEmMilhas( )  
    {  
        return quilometragem / 1.589;  
    }  
}
```

Considerando que o método TransformaKmEmMilhas para efeito ilustrativo, tenha o comportamento diferente na classe filha, analise o código, **identifique, justifique e comente** problemas existentes e quais seriam os ajustes necessários para tornar o código correto ? Obs: considere a necessidade do método nas duas classes. (2,0)

Resposta:

Problemas e correção

- quilometragem não é visível na classe filha, este atributo deveria se tornar um atributo protegido, assim a subclasse, conseguiria visualizar e acessá-lo
- Nome dos métodos idênticos na classe pai e filha causa divergência, uso errado da herança deveria ser feito o método da classe pai como virtual para manter seu comportamento padrão, feito um override na filha, para modificar seu comportamento particular

Prints da correção:

Alterações implementadas

```
-referências
class Moto
{
    string modelo, fabricante;
    protected int quilometragem;

    -referências
    public virtual double TransformaKmEmMilhas()
    {
        return quilometragem / 1.609;
    }
}
```

```
-referências
class MotoTriciclo : Moto
{
    -referências
    public override double TransformaKmEmMilhas()
    {
        return quilometragem / 1.589;
    }
}
```

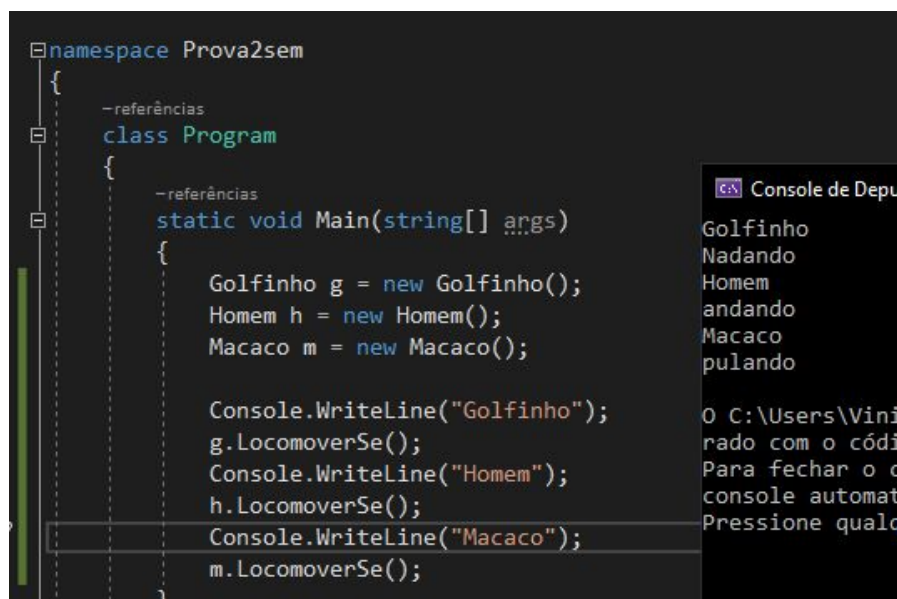
4) Codifique a **classe base, classes derivadas e método** de acordo com a descrição:

- Considere a classe Mamífero (classe base) , as classes derivadas Golfinho, Homem, Macaco, todos eles com um método em comum “Locomover-se” porém com comportamentos distintos (coloque uma mensagem qualquer). Com base SOMENTE nas considerações feitas, além de realizar a codificação, justifique a forma como a classe base Mamífero foi criada (é uma classe normal ?, sim/não, por que ?) (3,0)

Resposta:

Não, a classe base mamífero, trata-se de uma classe abstrata, que não é instanciada, e sim usada somente como base para suas derivadas, tem um método abstrato padrão que contém apenas sua assinatura, e que tem seu comportamento definido dentro de cada classe filha aplicando suas particularidades e diferenças

Classe program chamando os métodos das classes derivadas



```
namespace Prova2sem
{
    -referências
    class Program
    {
        -referências
        static void Main(string[] args)
        {
            Golfinho g = new Golfinho();
            Homem h = new Homem();
            Macaco m = new Macaco();

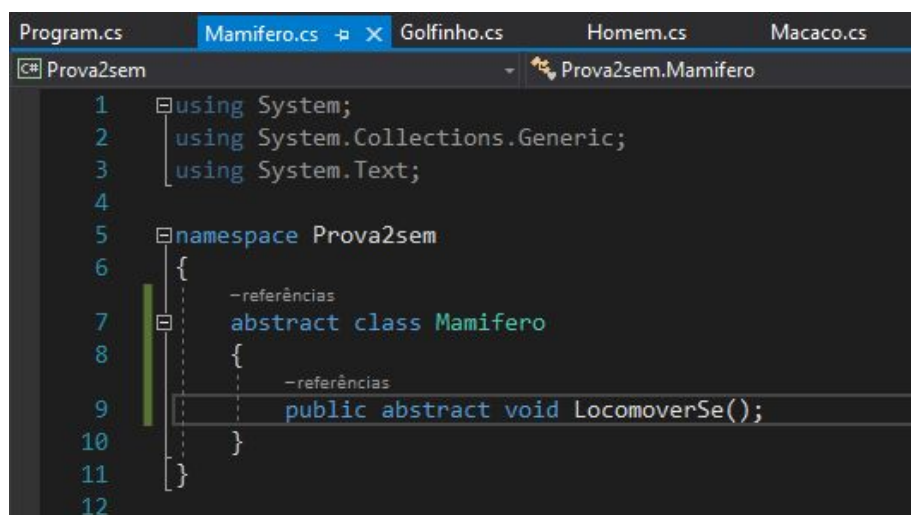
            Console.WriteLine("Golfinho");
            g.LocomoverSe();
            Console.WriteLine("Homem");
            h.LocomoverSe();
            Console.WriteLine("Macaco");
            m.LocomoverSe();
        }
    }
}
```

Console de Depuração

Golfinho
Nadando
Homem
andando
Macaco
pulando

O C:\Users\Vini
rado com o código
Para fechar o d
console automat
Pressione qualq

Classe base



```
Program.cs  Mamifero.cs  Golfinho.cs  Homem.cs  Macaco.cs
Prova2sem
1  using System;
2  using System.Collections.Generic;
3  using System.Text;
4
5  namespace Prova2sem
6  {
7      -referências
8      abstract class Mamifero
9      {
10         -referências
11         public abstract void LocomoverSe();
12     }
13 }
```

classes derivadas

```
Program.cs  Mamifero.cs  Golfinho.cs  Homem.cs  Macaco.cs
C# Prova2sem  Prova2sem.Golfinho

5 namespace Prova2sem
6 {
7     -referências
8     class Golfinho : Mamifero
9     {
10         -referências
11         public override void LocomoverSe()
12         {
13             Console.WriteLine("Nadando");
14         }
15     }
```

```
Program.cs  Mamifero.cs  Golfinho.cs  Homem.cs  Macaco.cs
C# Prova2sem  Prova2sem.Homem

2 using System.Collections.Generic;
3 using System.Text;
4
5 namespace Prova2sem
6 {
7     -referências
8     class Homem: Mamifero
9     {
10         -referências
11         public override void LocomoverSe()
12         {
13             Console.WriteLine("andando");
14         }
15     }
```

```
Program.cs  Mamifero.cs  Golfinho.cs  Homem.cs  Macaco.cs
C# Prova2sem  Prova2sem.Macaco

2 using System.Collections.Generic;
3 using System.Text;
4
5 namespace Prova2sem
6 {
7     -referências
8     class Macaco:Mamifero
9     {
10         -referências
11         public override void LocomoverSe()
12         {
13             Console.WriteLine("pulando");
14         }
15     }
```