Colégio Pedro II

PROGRAMAÇÃO O.O.

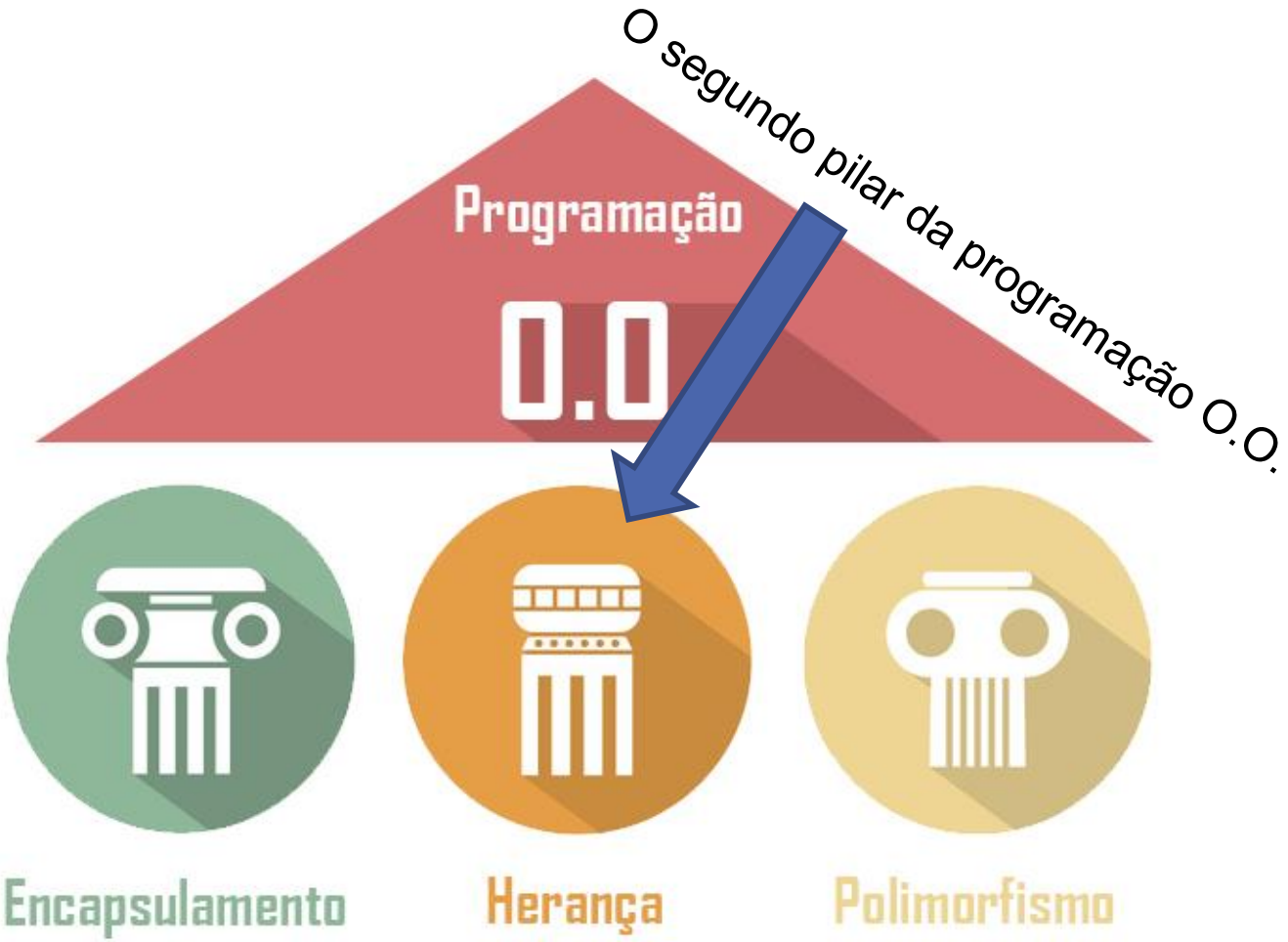
(*C#*)

{ O Segundo pilar da programação O.O.: Herança

Professor: João Luiz Lagôas }

Roteiro

Colégio Pedro II



Caso de uso:
produção de eventos

Colégio Pedro II

- Catarina está conseguindo gerenciar bem seus eventos e com a pronta resposta no atendimento aos clientes, a demanda só aumenta. Ela então pensa em modificar alguns valores....

Mudanças

Ela decide então começar a cobrar um pouco mais. O custo para cada pessoa nas festas agora será R$30 ao invés de R$25.

Ela decidiu também aumentar a taxa envolvida no custo da decoração. Ao invés de R$30 e R$50, ela mudará para R$45 e R$80, respectivamente.

Caso de uso:
produção de eventos

Colégio Pedro II

- Seria fácil realizar essas alterações?
  - Parece que sim!
- E se eu tivesse mais classes de festa? Umas 5? Ainda seria fácil fazer as alterações?
  - Daria um pouco mais de trabalho...
- Essa não é uma forma muito ineficiente de se codificar? Deve haver uma maneira melhor...

Caso de uso:
produção de eventos

Colégio Pedro II

- Seria fácil realizar essas alterações?
- Parece que sim!
- E se eu tive[r] Umas 5? Ainda seria fácil fa[zer]
- Daria uma pouco mais de trabalho...
- Essa não é uma forma muito ineficiente de se trabalhar? Deve haver uma maneira melhor...

Herança

Herança
Introdução

Colégio Pedro II

- Vamos começar observando o diagrama das duas classes que criamos para entender melhor o que está havendo.

DinnerParty
precoPorPessoa numeroDePessoas opcaoSaudavel opcaoChique
DinnerParty() CalcularPrecoDeBebida() CalcularPrecoDeDecoracao() CalcularCustoTotal()

BirthdayParty
precoPorPessoa numeroDePessoas boloGrande opcaoChique escrita
BirthdayParty() CalcularPrecoDaEscrita() CalcularPrecoDeDecoracao() CalcularCustoTotal()

Herança
Introdução

Colégio Pedro II

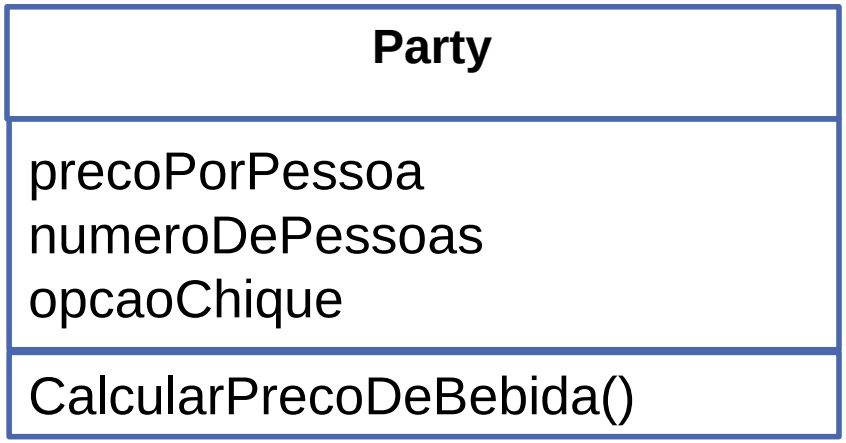
- O que há de semelhante?

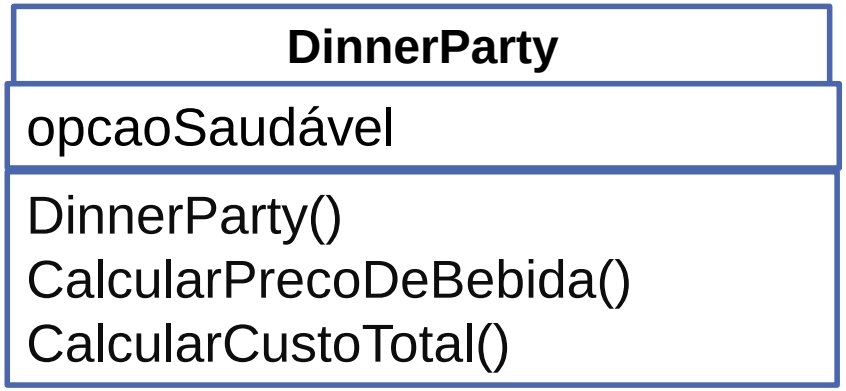
DinnerParty
precoPorPessoa numeroDePessoas opcaoSaudavel opcaoChique
DinnerParty() CalcularPrecoDeBebida() CalcularPrecoDeDecoracao() CalcularCustoTotal()

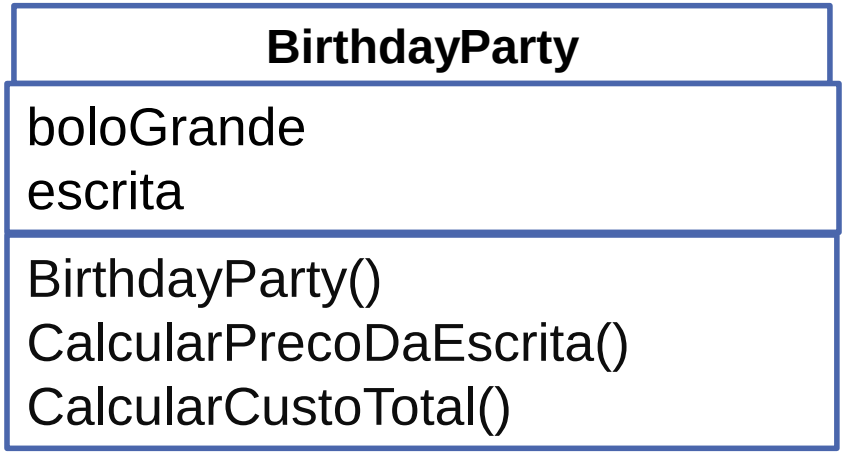
BirthdayParty
precoPorPessoa numeroDePessoas boloGrande opcaoChique escrita
BirthdayParty() CalcularPrecoDaEscrita() CalcularPrecoDeDecoracao() CalcularCustoTotal()

Herança
Introdução

Colégio Pedro II

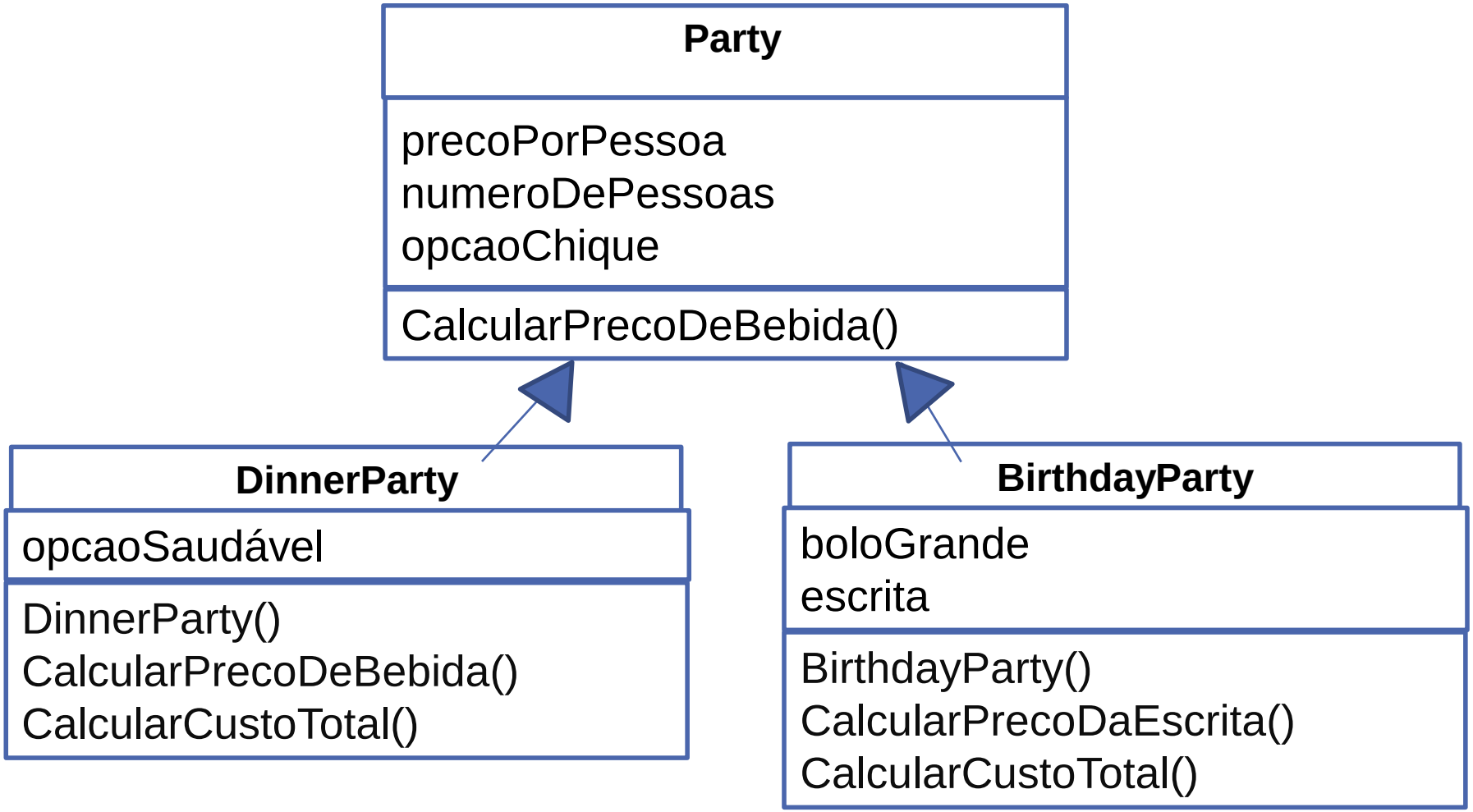






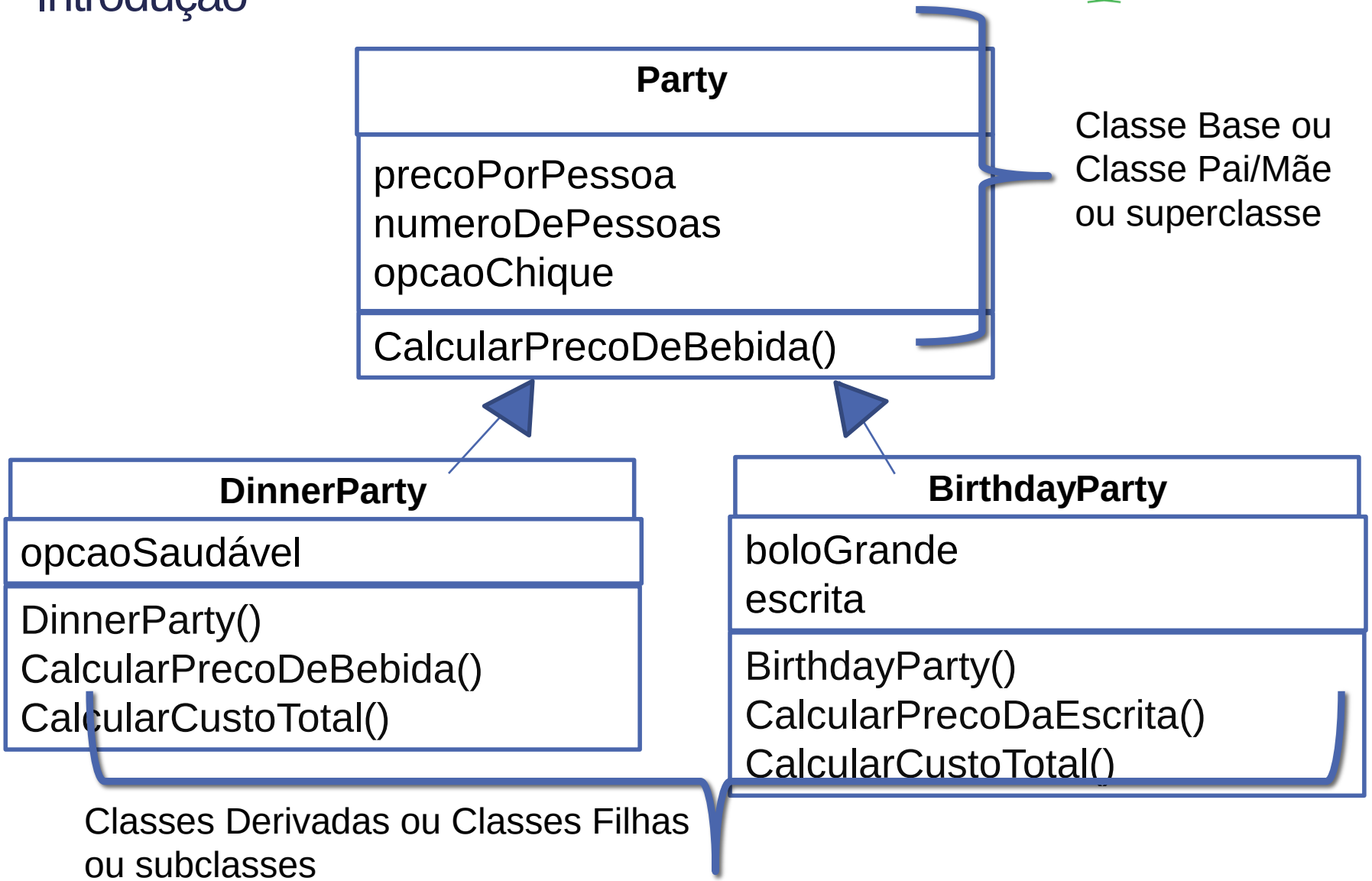
Herança
Introdução

Colégio Pedro II



Herança
Introdução

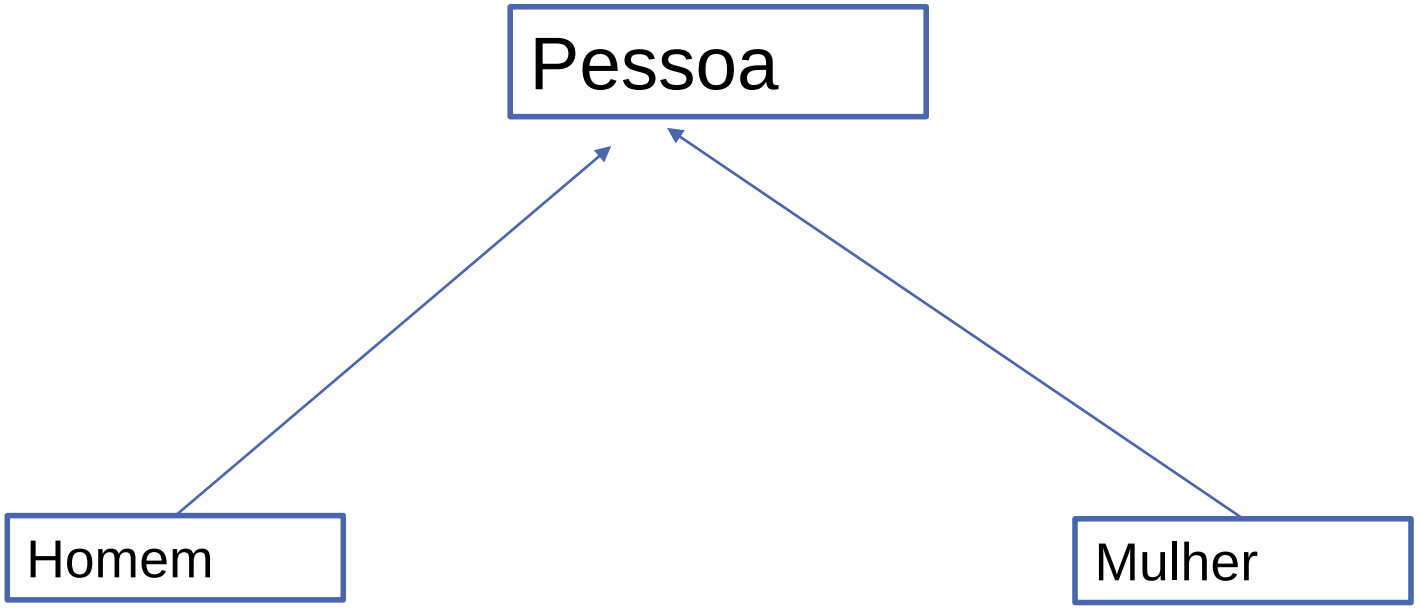
Colégio Pedro II



Herança

Exemplos de relação geral/específico

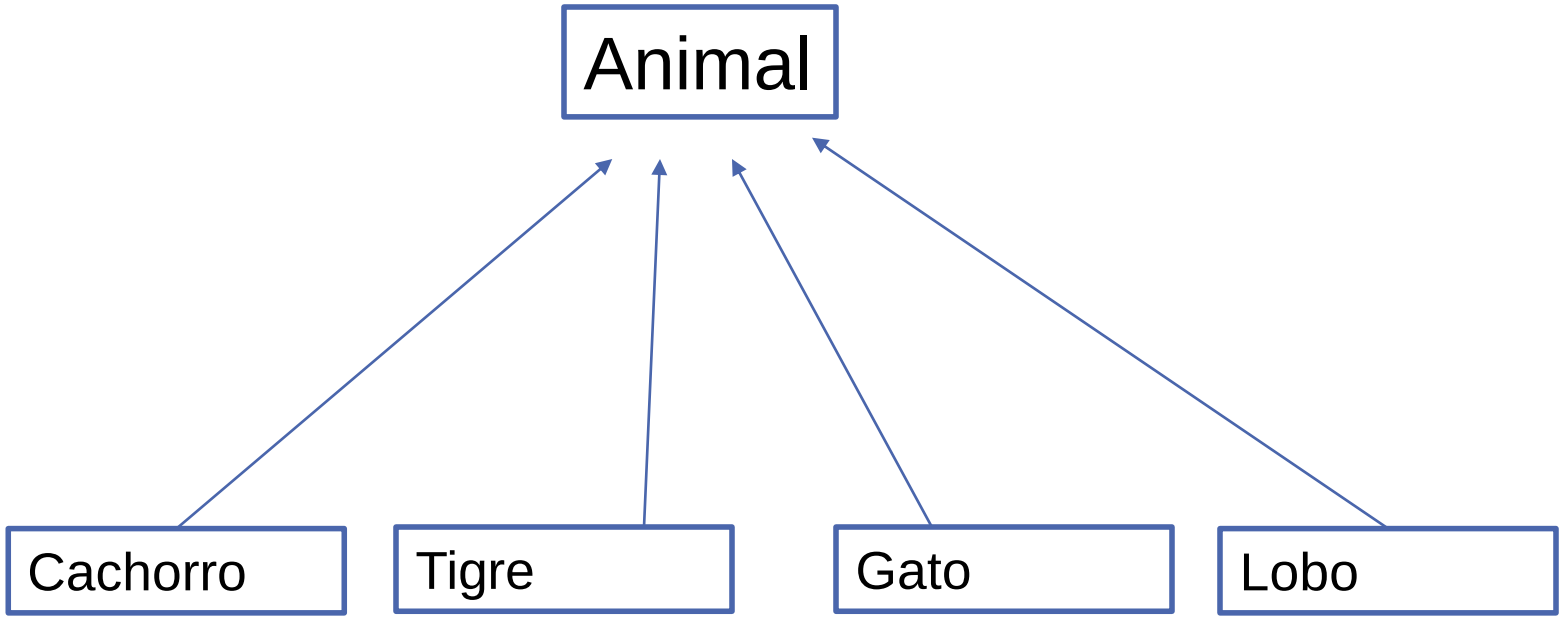
Colégio Pedro II



Herança

Exemplos de relação geral/específico

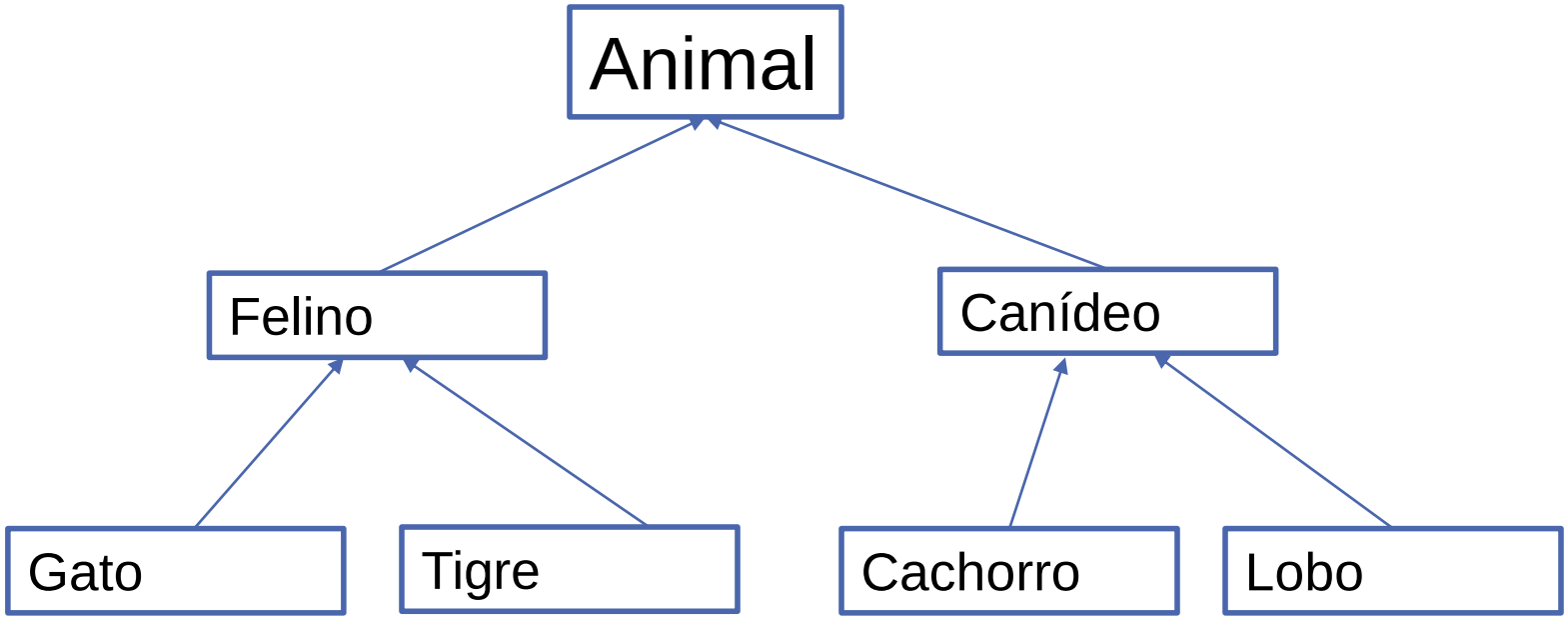
Colégio Pedro II



Herança

Exemplos de relação geral/específico

Colégio Pedro II



Herança
Introdução

Colégio Pedro II

- Não é coincidência que suas classes DinnerParty e BirthdayParty tenham muito código em comum.

- Quando você escreve programas C#, frequentemente cria classes que representam coisas do mundo real – e essas coisas geralmente se relacionam entre si.

Quando você tem duas classes que são **casos mais específicos** de **algo mais geral**, você pode utilizar o conceito de Herança para reusar código e evitar duplicação.

Herança
Introdução

Colégio Pedro II

Quando você tem classes que são **casos mais específicos** de **algo mais geral**, você pode utilizar o conceito de Herança para reusar código e evitar duplicação.

Party

DinnerParty
BirthdayParty

Isso é feito através da criação de uma classe mais geral (**classe base**) que passa todos os seus atributos e métodos para as classes específicas (**classes derivadas**).

Herança
Exemplo prático

Colégio Pedro II

Suponha que estamos implementando um jogo de **Bomberman**.

Sabemos que neste jogo todos os personagens (denominados Bomberman) apresentam uma quantidade de vida, uma velocidade, uma cor e a capacidade de deixar uma bomba.

Como seria o modelo de nossa classe Bomberman?!?

Herança
Exemplo prático

Colégio Pedro II

Bomberman
Vida Velocidade Cor
Bomberman() ColocarBombar()

Atributos

Métodos

Herança

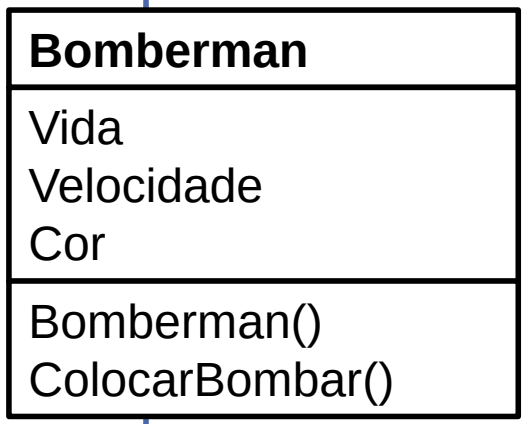
Exemplo prático

Colégio Pedro II

```
class Bomberman
    {
        public int vida = 10;
        public int velocidade = 5;
        public string cor = "branco";

        public Bomberman(int v, int vel, string c)
        {
            vida = v;
            velocidade = vel;
            cor = c;
        }

        public void ColocarBomba()
        {
            Console.WriteLine("Colocando Bomba!");
        }
    }
```



Herança
Exemplo prático

Colégio Pedro II

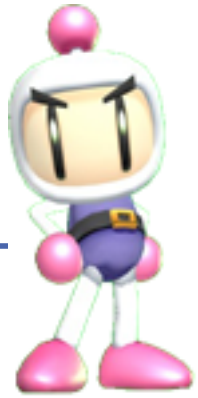
```
class Program
    {
        static void Main(string[] args)
        {
            Bomberman whitebomber = new Bomberman(10, 5, "branco");
            Bomberman blackbomber = new Bomberman(10, 5, "preto");
            Bomberman bluebomber = new Bomberman(10, 5, "azul");

            whitebomber.ColocarBomba();
            blackbomber.ColocarBomba();
            bluebomber.ColocarBomba();

            Console.ReadLine();
        }
    }
```

Herança
Exemplo prático

Colégio Pedro II



```
class Program
    {
        static void Main(string[] args)
        {
            Bomberman whitebomber = new Bomberman(10, 5, "branco");
            Bomberman blackbomber = new Bomberman(10, 5, "preto");
            Bomberman bluebomber = new Bomberman(10, 5, "azul");

            whitebomber.ColocarBomba();
            blackbomber.ColocarBomba();
            bluebomber.ColocarBomba();

            Console.ReadLine();
        }
    }
```

Herança

Exemplo prático

Colégio Pedro II

Seu jogo funciona e você decide então adicionar outros personagens ao game além dos Bombermans. Você então pensa em 4 novos personagens, todos compartilhando as mesmas características e comportamentos dos Bombermans mas com poderes específicos exclusivos de cada um personagem. Os personagens que você pensou são:

- Fire Bomber (é capaz de Explodir),
- Aqua Bomber (é capaz de Nadar),
- Cyclone Bomber (é capaz de Voar),
- Earth Bomber (é capaz de Rolar).




Herança

Exemplo prático

Colégio Pedro II

Como se tratam de personagens novos com funcionalidades novas, não tem jeito... Teremos que criar uma classe para cada um deles. Mas muitos dos comportamentos e características já foram implementados na classe Bomberman. Por que não aproveitá-las?

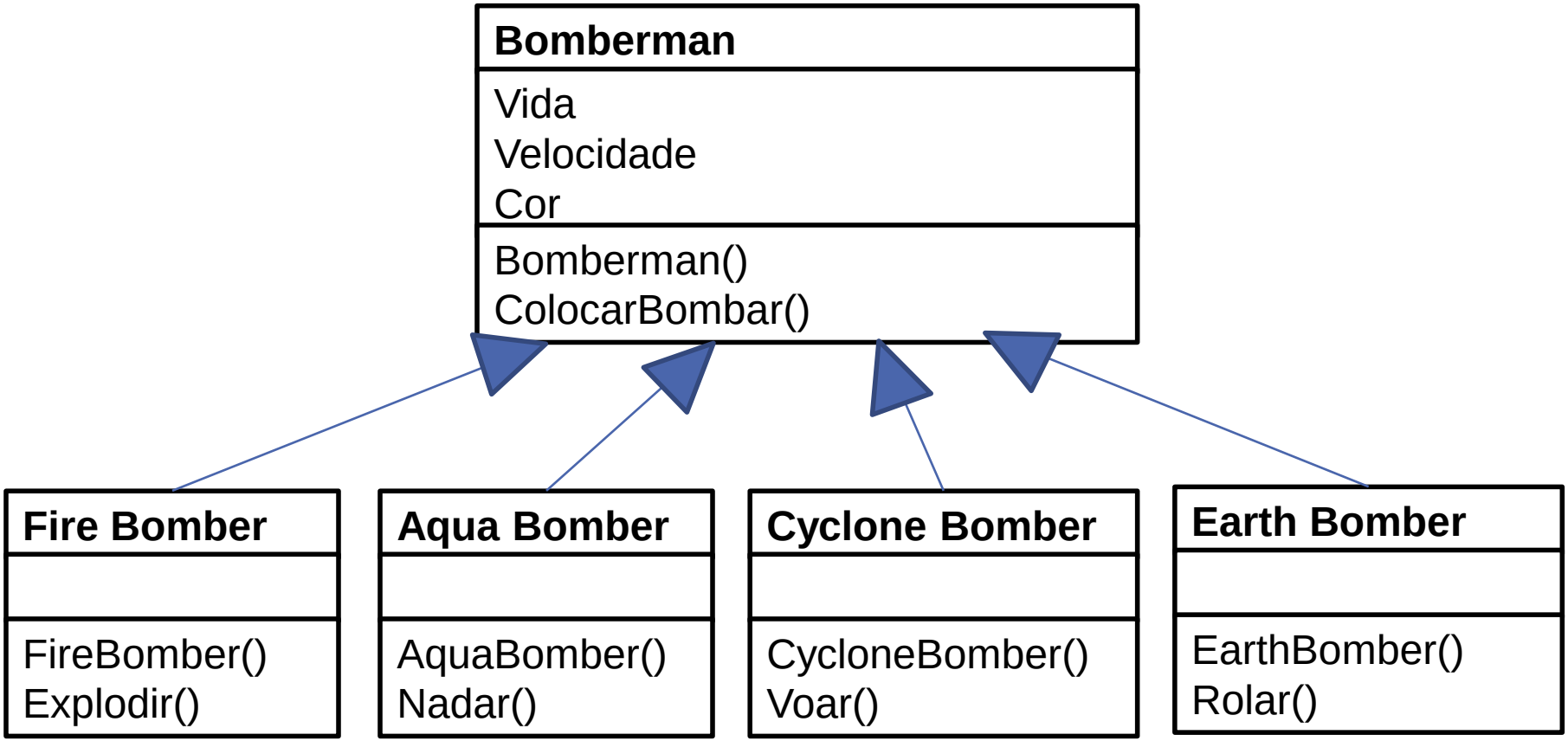
Seu jogo f adicionar outros personag ns. Você então pensa e mpartilhando as mesmas s Bombermans mas cor de cada um personag são:

- Fire Bom
- Aqua Bo paz de Nadar),
- Cyclone Bomber (é capaz de Voar),
- Earth Bomber (é capaz de Rolar).




Herança
Exemplo prático

Colégio Pedro II



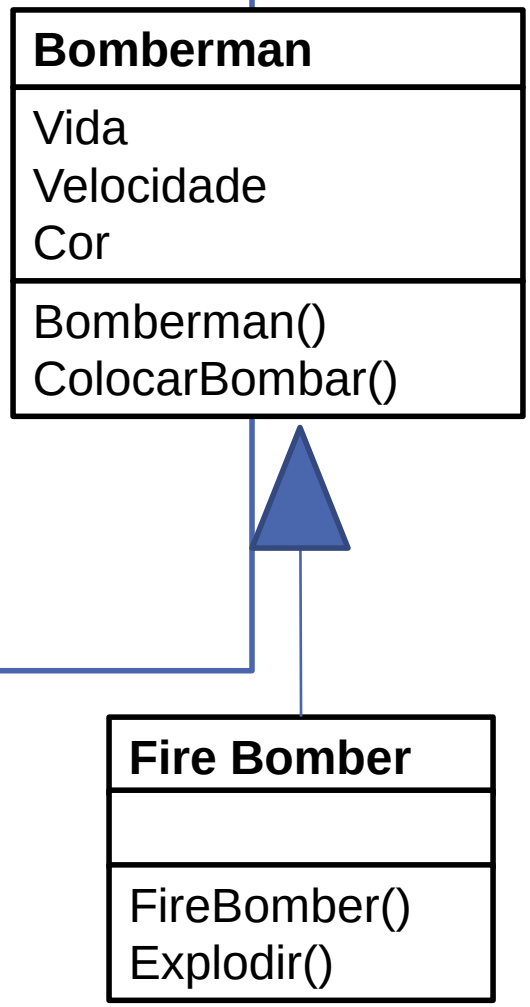
Herança

Exemplo prático

Colégio Pedro II

```
class FireBomber : Bomberman
{
        public FireBomber(int v, int vel, string c)
                :base(v, vel, c)
        {
        }

        public void Explodir()
        {
            Console.WriteLine("Explodindo!");
        }
    }
```



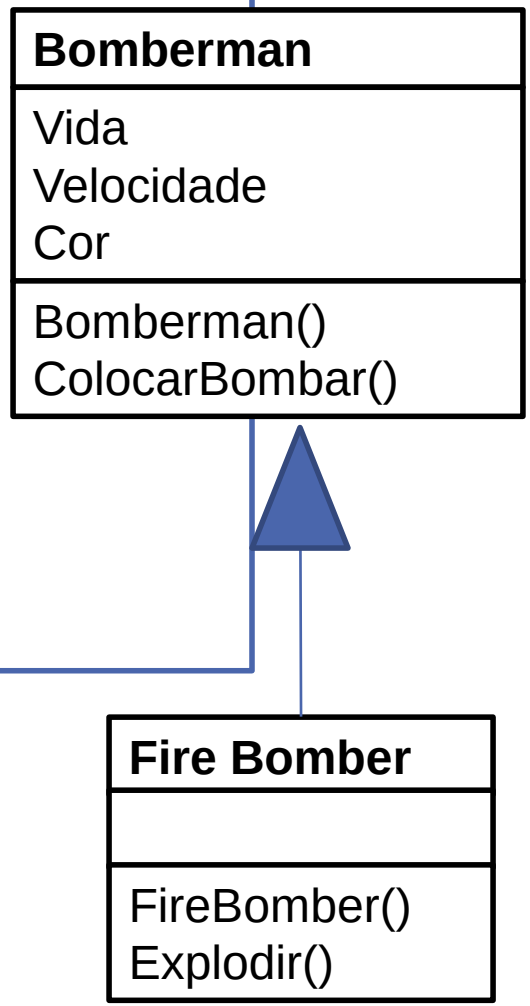
Herança

Exemplo prático

Os **dois pontos** indicam que a classe FireBomber é classe filha da classe Bomberman.

```
class FireBomber : Bomberman
{
        public FireBomber(int v, int vel, string c)
                :base(v, vel, c)
        {
        }

        public void Explodir()
        {
            Console.WriteLine("Explodindo!");
        }
    }
```



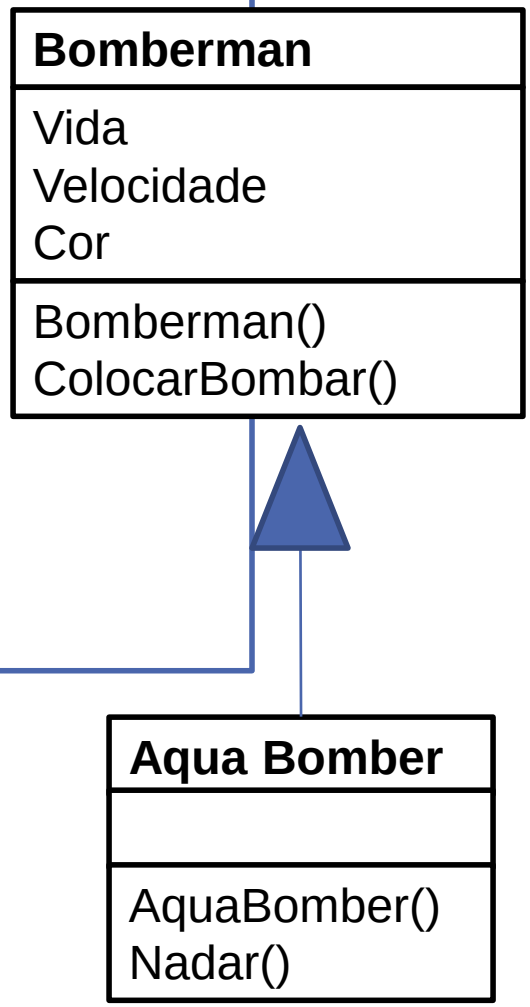
Herança

Exemplo prático

Colégio Pedro II

```
class AquaBomber : Bomberman
    {
        public AquaBomber(int v, int vel, string c)
        :base(v, vel, c)
        {
        }

        public void Nadar()
        {
            Console.WriteLine("Nadando!");
        }
    }
```

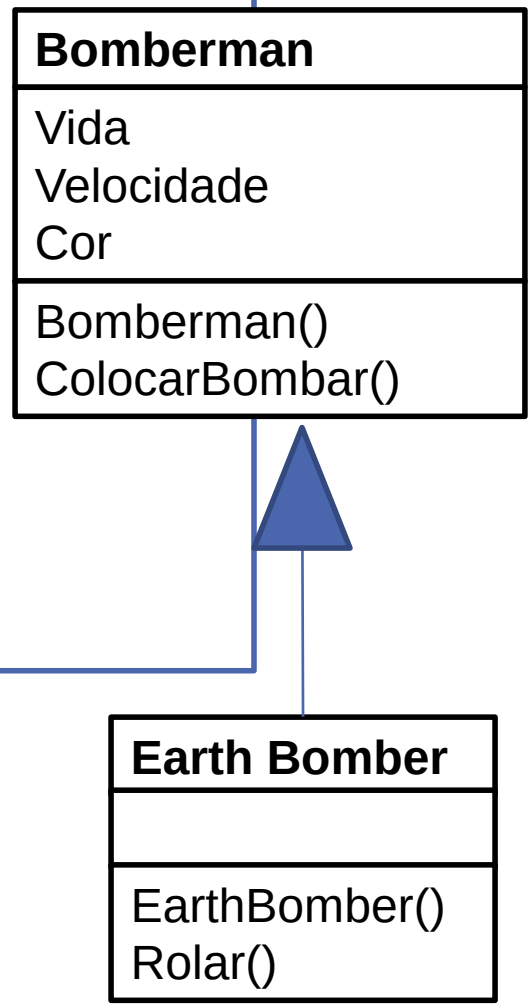


Herança
Exemplo prático

Colégio Pedro II

```
class EarthBomber : Bomberman
    {
        public EarthBomber(int v, int vel, string c)
        :base(v, vel, c)
        {
        }

        public void Rolar()
        {
            Console.WriteLine("Rolando!");
        }
    }
```



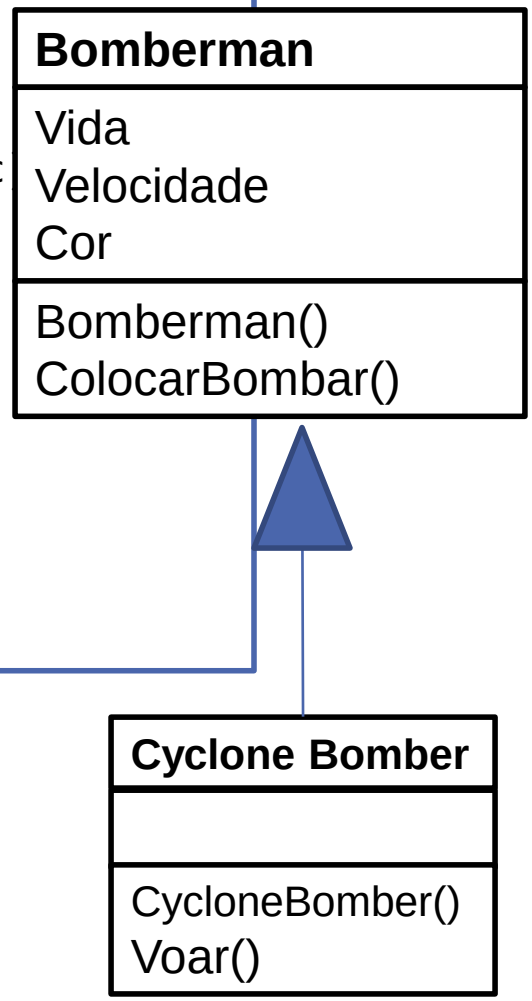
Herança

Exemplo prático

Colégio Pedro II

```
class CycloneBomber : Bomberman
{
        public CycloneBomber(int v, int vel, string c
                :base(v, vel, c)
        {
        }

        public void Voar()
        {
            Console.WriteLine("Voando!");
        }
    }
```



Herança
Exemplo prático

Colégio Pedro II

```
class Program
   {
       static void Main(string[] args)
       {
           Bomberman whitebomber = new Bomberman(10, 5, "branco");
           Bomberman blackbomber = new Bomberman(10, 5, "preto");
           Bomberman bluebomber = new Bomberman(10, 5, "azul");

           whitebomber.ColocarBomba();
           blackbomber.ColocarBomba();
           bluebomber.ColocarBomba();

           FireBomber fireBomber = new FireBomber(20, 2, "vermelho");

           fireBomber.ColocarBomba();
           fireBomber.Explodir();

           AquaBomber aquaBomber = new AquaBomber(8, 14, “verde");

           aquaBomber.ColocarBomba();
           aquaBomber.Nadar();

           Console.ReadLine();
       }
   }
```

Herança
Exemplo prático

Colégio Pedro II

```
class Program
   {
        static void Main(string[] args)
        {
            Bomberman whitebomber = new Bomberman(10, 5, "branco");
            Bomberman blackbomber = new Bomberman(10, 5, "preto");
            Bomberman bluebomber = new Bomberman(10, 5, "azul");

            whitebomber.ColocarBomba();
            blackbomber.ColocarBomba();
            bluebomber.ColocarBomba();

            FireBomber fireBomber = new FireBomber(20, 2, "vermelho");

            fireBomber.ColocarBomba();
            fireBomber.Explodir();

            AquaBomber aquaBomber = new AquaBomber(8, 14, “verde");

            aquaBomber.ColocarBomba();
            aquaBomber.Nadar();

            Console.ReadLine();
        }
   }
```

Herança

Exemplo p

Colégio Pedro II

Note que as classes filhas têm o método ColocarBomba() mesmo nós não implementando ele na classe! Isso acontece porque os dois pontos (:) criaram uma relação de herança entre as classe Bomberman e as classes FireBomber e AquaBomber. Sendo assim, as classes filhas HERDARAM todos os ATRIBUTOS e MÉTODOS da classe pai.

```
class Program
    {
        static voi
        {
            Bomber
            Bomber
            Bomber

            whiteb
            blackb
            bluebomber.ColocarBomba();

            FireBomber fireBomber = new FireBomber(20, 2, "vermelho");

            fireBomber.ColocarBomba();
            fireBomber.Explodir();

            AquaBomber aquaBomber = new AquaBomber(8, 14, “verde");

            aquaBomber.ColocarBomba();
            aquaBomber.Nadar();

            Console.ReadLine();
        }
    }
```

Herança
Exemplo prático

Colégio Pedro II

- Observe o seguinte trecho do código em todas as classes filhas...

```
class FireBomber : Bomberman
{
        public FireBomber(int v, int vel, string c)
                          :base(v, vel, c)
        {
        }

        public void Explodir()
        {
            Console.WriteLine("Explodindo!");
        }
    }
```

Herança
Exemplo prático

Colégio Pedro II

- O que será que a palavra-chave **base** significa dentro do construtor de FireBomber?

```
class FireBomber : Bomberman
{
        public FireBomber(int v, int vel, string c)
                          :base(v, vel, c)
        {
        }

        public void Explodir()
        {
            Console.WriteLine("Explodindo!");
        }
    }
```

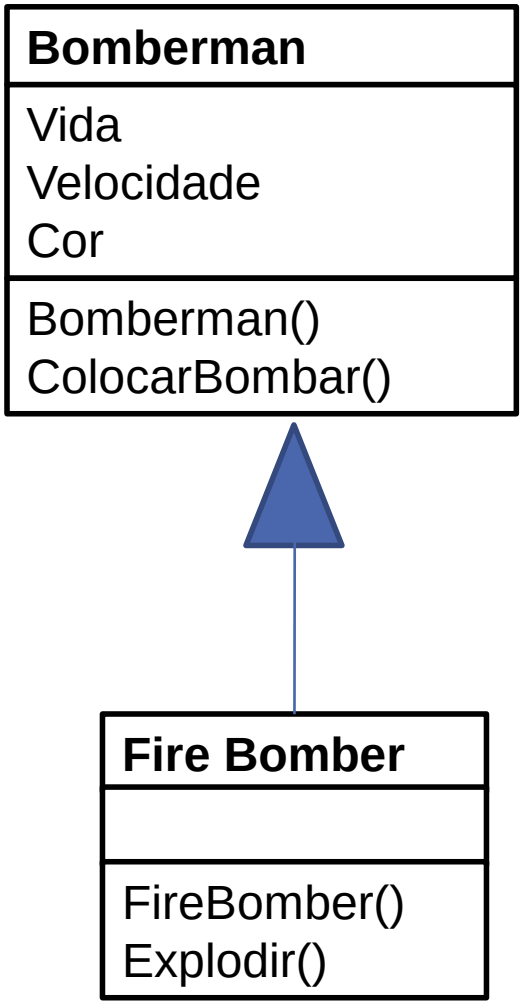
Herança

Processo de construção de objetos

Colégio Pedro II

- O construtor da classe base é sempre executado antes do construtor da subclasse. Isso faz total sentido já que para que uma subclasse é uma ESPECIALIZAÇÃO da classe pai.

- A palavra base é utilizada para CHAMAR o construtor da classe pai passando os devidos parâmetros. Ao término da sua construção, o código continua sua execução construindo a classe filha.



Herança

Processo de construção de objetos

Colégio Pedro II

```
class Bomberman
    {
        public int vida = 10;
        public int velocidade = 5;
        public string cor = "branco";

        public Bomberman(int v, int vel, string c)
        {
            vida = v;
            velocidade = vel;
            cor = c;

            Console.WriteLine("Construindo bomberman...");
        }

        public void ColocarBomba()
        {
            Console.WriteLine("Colocando Bomba!");
        }
    }
```

Herança

Processo de construção de objetos

Colégio Pedro II

```
class FireBomber : Bomberman
    {
        public FireBomber(int v, int vel, string c):base(v, vel, c)
        {
            Console.WriteLine("Construindo Firebomber...");
        }

        public void Explodir()
        {
            Console.WriteLine("Explodindo!");
        }
    }
```

Herança

Processo de construção de objetos

Colégio Pedro II

```
class Program
    {
        static void Main(string[] args)
        {
            Bomberman whitebomber = new Bomberman(10, 5, "branco");

            FireBomber fireBomber = new FireBomber(20, 2, "vermelho");

            Console.ReadLine();
        }
    }
```

Herança

Processo de construção de objetos

Colégio Pedro II

Console

```
Construindo bomberman...
```

```
class Program
    {
        static void Main(string[] args)
        {
            Bomberman whitebomber = new Bomberman(10, 5, "branco");

            FireBomber fireBomber = new FireBomber(20, 2, "vermelho");

            Console.ReadLine();
        }
    }
```

Herança

Processo de construção de objetos

Colégio Pedro II

```
class Program
    {
        static void Main(string[] args)
        {
            Bomberman whitebomber = new Bomberman(10, 5, "branco");

            FireBomber fireBomber = new FireBomber(20, 2, "vermelho");

            Console.ReadLine();
        }
    }
```

Console

```
Construindo bomberman...
Construindo firebomber...
```

Herança

Ideia básica e resumão

Colégio Pedro II

- Utilizamos os : os para fazer com que uma classe filha HERDE todos os atributos e métodos de uma classe pai.
- O processo de construção de objetos de classes filhas acontece sempre com a chamada do construtor da classe pai e DEPOIS o construtor da classe filha.
- Utilize a palavra reservada base precedida dos dois pontos no construtor de uma classe filha para indicar a chamada ao construtor pai.

Voltando ao problema da Catarina

Colégio Pedro II

- Quando deixamos Catarina pela última vez, ela estava satisfeita com seu sistema. No entanto, nós sabemos que há muita duplicação de código nas classes BirthdayParty e DinnerParty de modo que aplicar o conceito de Herança resolveria vários futuros problemas no sistema.

- Sua tarefa é abrir a solução contida no arquivo Projeto Catarina.rar, já com as classes Form1, DinnerParty e BirthdayParty e criar uma quarta classe Party para aplicar os conceitos de Herança.

- Para tal, siga o diagrama do próximo slide para te auxiliar.

- **Comece implementando a classe base Party.**

Voltando ao problema da Catarina

Colégio Pedro II