Programação Orientada a Objetos

Instituto Politécnico do Cávado e do Ave

Escola Técnica Superior Profissional

CTeSP – Tecnologia e Inovação Informática

Ano Letivo: 2021/2022

DOCENTE: EDUARDO PEIXOTO

AGENDA

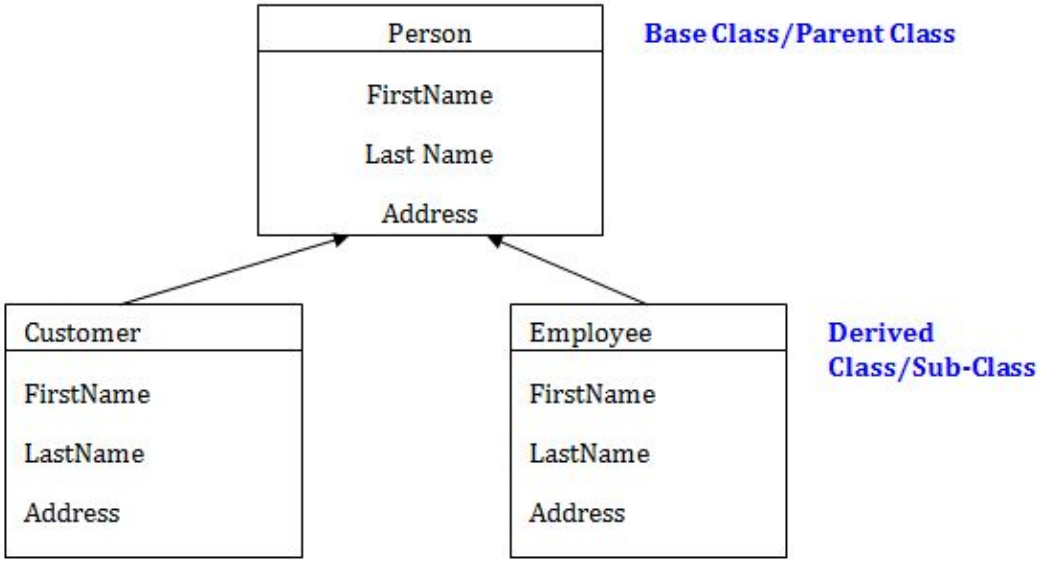
ORGANIZAÇÃO DA APRESENTAÇÃO

1. Herança e Polimorfismo

Herança e Polimorfismo(1/3)

- As classes são estruturadas hierarquicamente em superclasses, classes e subclasses por recurso à herança. As classes herdam as propriedades e os métodos das superclasses que derivam, acrescentam mais propriedades e métodos, e passam tudo às subclasses que delas derivarem. Esta hierarquia de classes é representada graficamente. Uma vez que uma subclasse é uma especialização de superclasse de que deriva, as árvores de classes mostram os diferentes níveis de generalização.

Herança e Polimorfismo (2/3)



Herança e Polimorfismo (3/3)

- A **herança** como veículo transmissor da especialização de subclasses é reforçada pelo polimorfismo. O **polimorfismo** é a transposição para a programação de polimorfia, que é a propriedade das substâncias que se apresentam em várias formas. O polimorfismo permite ao programador definir métodos genéricos em classes ou interfaces, que podem ser implementadas de diversas formas nas respectivas subclasses. O mesmo método pode, portanto, oferecer funcionalidades diferentes para cada classes que o implementa.

Derivação de uma subclasse

- Uma subclasse deriva de uma classe, designada por classe base, herdando todas as propriedades e métodos.
- Exemplo:
  - public class Circulo: Centro {
    - //variáveis de instância
    - //variáveis estáticas
    - //construtores
    - //acessores
    - //métodos
  - }

Construtores de uma classe

- Os construtores de uma classe constroem os objetos, colocando as suas propriedades - as especializadas definidas nessa subclasse e as genéricas que herdam da classe base - num estado inicial.
- Exemplo:
  - public Circulo (int x, int y, double R): base (x, y){
    - Raio = R;
  - }

Variáveis de instância das classes e subclasses

- As variáveis de instância e as variáveis estáticas da classe base podem ser acedidas por métodos das subclasses se forem definidas com o atributo *protected*.
- Exemplo:
  - public class Pai{
    - protected string NomePai;
  - }

Invocação de métodos da subclasse e da classe base(1/2)

- Os métodos de instância da classe base ou das subclasses são invocadas para os respectivos objetos.
- Exemplo:
  - //Moradia é uma subclasse de Propriedades
  - Moradia M = new Moradia(“Filipe”, 500100100, “Braga”);
  - Console.WriteLine(“{0} Renda Minima={1}”, ImpressaoProp(), M.RendaMinima());

Invocação de métodos da subclasse e da classe base(2/2)

- Os métodos estáticos da classe base ou das subclasses são invocados para a classe que os contém.
- Exemplo:
  - //Moradia é uma subclasse de Propriedades
  - Console.WriteLine(Moradia.MensagemM());
  - Console.WriteLine(Propriedades.MensagemP());

Destruidores(1/2)

- Os destruidores são métodos que removem os objetos. A destruição de um objeto tem de ser efetuada a partir do seu subobjeto com maior profundidade.
- Exemplo:
  - ~Escola(){
    - Console.WriteLine(“Destruí escola: {0}”, Nome);
  - }

Destruidores(2/2)

- A destruição de objetos pode ser seguida pela sua remoção física, através da inovação da respectiva facilidade de gestão de memória - o Garbage Collection do *.Net Framework*. O programador pode, assim, forçar a recolha de objetos que deixaram se ser referenciados.
- Exemplo:
  - A = null;
  - System.GC.Collect();

Virtual e override(1/3)

- Uma subclasse pode implementar de forma diferente os métodos virtuais que foram definidos na sua classe base. Por exemplo, a implementação do método virtual *NotaFinal*, que foi definido na classe *AvaliaçãoGeral*, foi posteriormente sobreposta por outra implementação definida na subclasse *AvaliacaoEspecifica*. A implementação que sobrepõe o método virtual tem de ser qualificada com o atributo *override*.

Virtual e override(2/3)

Exemplo

- public class AvaliacaoGeral{
  - Public virtual void NotalFinal(){
    - Console.WriteLine(“Regime Geral:”)
    - CFinal = (int)Math.Round(T1, 0);
  - }
- …}

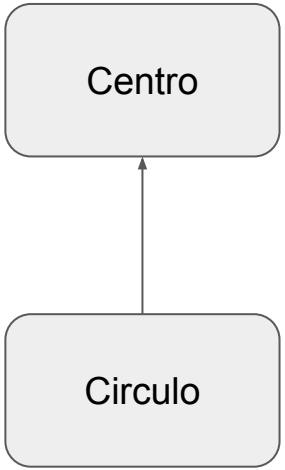
- public class AvaliacaoEspecifica: AvaliacaoGeral{
  - Public virtual void NotalFinal(){
    - Console.WriteLine(“Regime específico para Informática:”)
    - base.CFinal = (int)Math.Round((NotaT1 + T2)/2, 0);
  - …}
- }

Virtual e override(3/3)

- Um método virtual tem as seguintes caraterísticas:
  - Tem de ser explicitamente qualificado com o atributo virtual;
  - Não pode ser qualificado com os atributos *static*, *abstract*, *private* ou *override*;
  - Pode ser, ou não, implementado de forma diferente pelas subclasses;
  - A sua implementação pode ser sobreposta sucessivamente por todas as subclasses, isto é, um método virtual pode ser sobreposto por outro definido numa subclasse ou nas subclasses que derivam desta, e assim sucessivamente.
  - A sua invocação recai sobre a implementação mais próxima do objeto, dada a hierarquia de classes.

Exercício(1/2)

- Define a classe Centro e a sua subclasse Circulo:



Exercício(2/2)

- A classe Centro encapsula:
  - Duas variáveis de instância, x e y, que são as coordenadas de um ponto;
  - Um construtor;
  - Um método de impressão das coordenadas.
- A subclasse Circulo encapsula:
  - A variável de instância Raio;
  - Um construtor;
  - Um método para calcular a área do círculo.

Exercício (resolução)

```
public class Centro
{
        private int x, y;
        public Centro()
        {
                x = 0;
                y = 0;
        }
        public Centro(int x, int y)
        {
                this.x = x;
                this.y = y;
        }
        public void CoodCentro()
        {
                Console.WriteLine("Centro = ({0}, {1})", x, y);
        }
}
```

```
public class Circulo : Centro
{
        private double Raio;
        public Circulo()
        {
        }
        public Circulo(int x, int y, double R): base(x, y)
        {
                Raio = R;
        }
        public double Area()
        {
                return Math.Round(Math.PI * Math.Pow(Raio, 2), 2);
        }
}
```

Questões