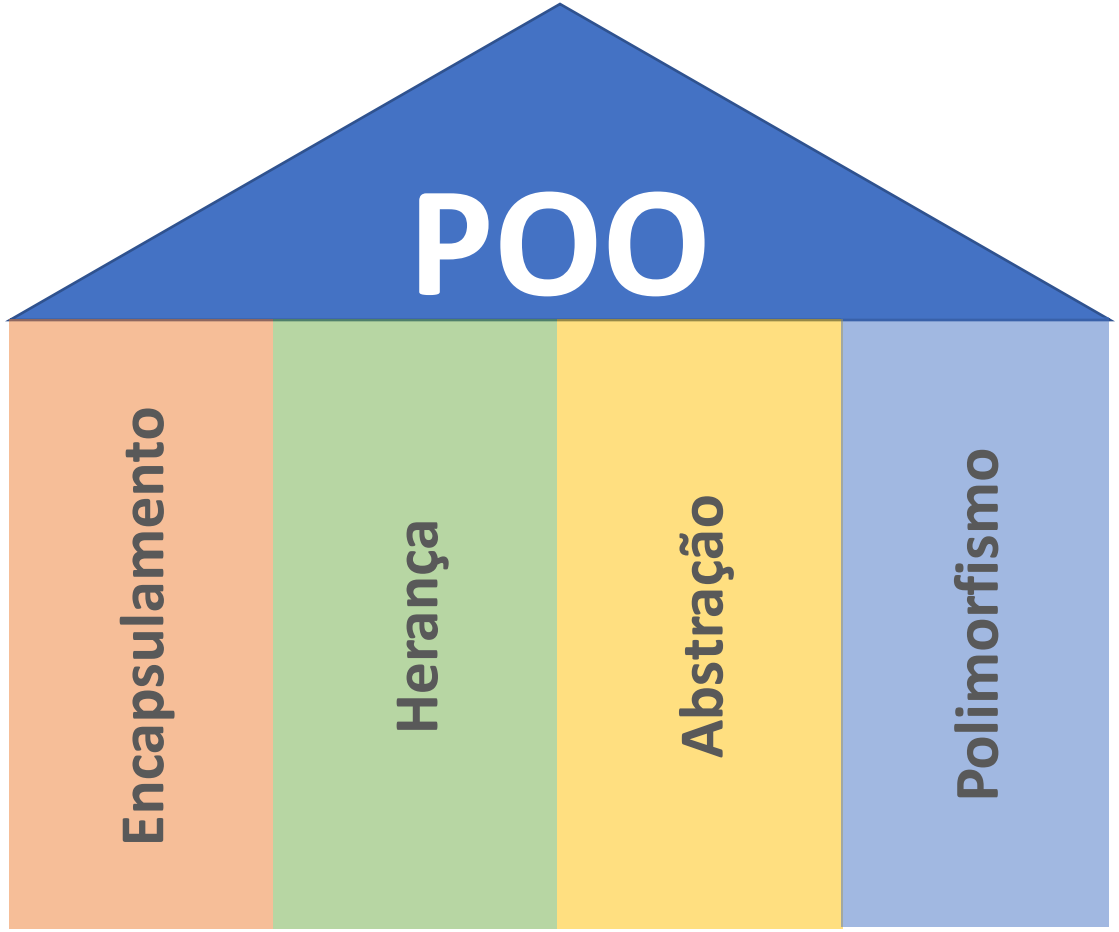
SENAI

CODIFICAÇÃO BACK-END

Pilares POO

Pilares de POO



Abstração

- Atributo/método que se repete
- Super classe ou classe abstrata
- Conceitual (acontece na construção do código)
- Não tem instância

```
public class ContaCorrente
{
    public double Saldo { get; set; }
}
```

```
public class ContaPoupanca
{
    public double Saldo { get; set; }
}
```

```
// classe Conta.cs

namespace OPP
{
    public abstract Conta
    {
      public double Saldo { get; set; }
    }
}
```

Namespace é uma palavra-chave usada para agrupar e organizar o código.

Herança

- **Como não podemos instanciar a super classe, instanciamos a classe filha da super classe, que herda todos os atributos e métodos.**

```
// Conta.cs

namespace OPP
{
    public abstract class Conta
    {
      public double Saldo {get; set;}
    }
}
```

```
// ContaCorrente.cs

namespace OOP
{
    public class ContaCorrente : Conta
    {

    }
}
```

Herança

- Para acessar o saldo, instanciamos a classe ContaCorrente.

```
// Program.cs

static void Main(string[] args)
     {
       ContaCorrente c = new ContaCorrente();
       Console.WriteLine(c.saldo);
     }
```

Polimorfismo

- **Sobrescrita → mesmo nome, comportamentos diferentes**
- **Classe filha sobrepõe (*override*) o comportamento da classe pai**
- **Sempre usado na herança**

```
class Compra
{
 public virtual void CalculaDesconto(){}
}

class CompraDebito : Compra
{
 public override void CalcularDesconto()
 {
     // cálculo do desconto no débito
 }
}

class CompraCredito : Compra
{
 public override void CalcularDesconto()
 {
     // cálculo do desconto no crédito
 }
}
```

Polimorfismo

- **Sobrecarga → métodos de mesmo nome com cargas, argumentos e parâmetros diferentes**
- **Sempre dentro da mesma classe**

```
class CompraCreditos : Compra
{
    public virtual void CalcularJuros()
    {
        // cálculo dos juros no crédito para compras à vista
    }

    public virtual void CalcularJuros (int parcelas)
    {   // cálculo do juros no crédito conforme o número de parcelas

    }
}
```

Encapsulamento

Encapsular = proteger

- Private set (só altera o valor dentro da mesma classe)
- Para alterar o saldo, criamos método para depositar ou sacar
- this.saldo += valorDepositado (acessa saldo e soma com o valor depositado)

```
public class Conta
    {
        public double saldo { get; private set; }

        public void Depositar(double valorDepositado)
        {
            this.saldo += valorDepositado;
        }
        public void Sacar(double valorSolicitado)
        {
            this.saldo -= valorSolicitado;
        }
    }
```

Bons estudos!