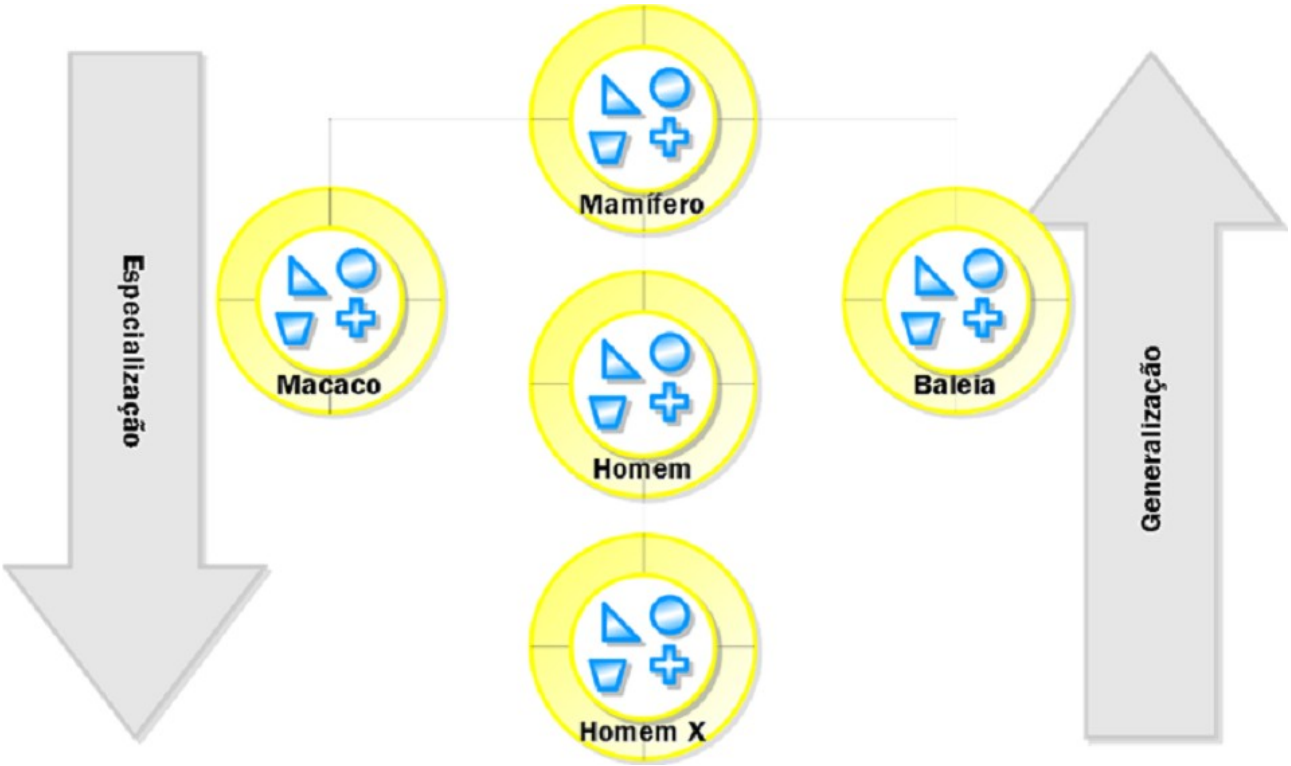
POLIMORFISMO – O TERCEIRO PILAR DA POO

O encapsulamento é o primeiro pilar da programação orientada a objetos, a herança é o segundo e o polimorfismo é o terceiro pilar. **O polimorfismo permite criar software que facilite a sua futura alteração e manutenção.**

Polimorfismo **significa muitas formas.** Em programação orientada a objetos o polimorfismo permite que um único nome de classe ou nome de método represente um código diferente, selecionado por algum mecanismo automático. Assim, como o mesmo nome pode representar código diferente, o mesmo nome pode representar muitos comportamentos diferentes.

Outra forma de definir o **polimorfismo** é dizer que ele é a propriedade de duas ou mais classes derivadas de uma mesma superclasse responderem a mesma mensagem, cada uma de uma forma diferente. Ocorre quando uma subclasse redefine um método existente na superclasse, ou seja, quando temos os **métodos sobrescritos (overriding).**

Por exemplo, na imagem abaixo, temos uma hierarquia da classe Mamífero:



Vamos supor que a classe Mamífero tem o método locomoverSe():

Superclasse Mamífero:

```
locomoverSe() {
    // A ser implementado em alguma subclasse.
}
```

As subclasses Macaco, Homem e Baleia herdam e sobrescrevem o método citado da seguinte forma:

Subclasse Macaco:

```
locomoverSe() {
    pulandoDeGalhoEmGalho;
}
```

Subclasse Homem:

```
locomoverSe() {
    andando;
}
```

Subclasse Baleia:

```
locomoverSe() {
    nadando;
}
```

Os objetos de cada classe responderão à mesma mensagem (locomoverSe()), mas cada um de uma forma diferente. O que caracteriza o principal conceito do polimorfismo, veja como poderia ser o código da aplicação lá no método main():

```
Mamifero variavelObjeto;

variavelObjeto = new Macaco();
variavelObjeto.locomoverSe(); // pulando de galho em galho

variavelObjeto = new Homem();
variavelObjeto.locomoverSe(); // andando

variavelObjeto = new Baleia();
variavelObjeto.locomoverSe(); // nadando
```

Com a mesma variável-objeto do tipo Mamifero, podemos usá-la para receber objetos dos tipos das subclasses de Mamifero. O que torna o código bem enxuto e limpo.

Recapitulando: o principal conceito do polimorfismo é a propriedade de duas ou mais classes derivadas (Macaco, Homem e Baleia) de uma mesma superclasse (Mamifero) responderem a mesma mensagem (locomoverSe()), cada uma de uma forma diferente (pulando de galho em galho, andando e nadando, respectivamente).

Questões de Concurso

[IFSE 2016 IFSE – Analista de Tecnologia da Informação – Desenvolvimento] Analise o trecho de código abaixo:

```
animal a = new animal( );
a.locomover( );
a = new cobra( );
a.locomover( );
a = new gato( );
a.locomover( );
```

É um exemplo de:
[A] Encapsulamento
[B] Polimorfismo
[C] Troca de mensagens
[D] Classe abstrata

Comentários:
O código da questão é bem parecido com o exemplo que foi dado no artigo. Há uma superclasse animal e duas subclasses: cobra e gato. A variável-objeto a é do tipo da superclasse e pode apontar para objetos das subclasses.

Cada subclasse pode sobrescrever o método locomover(), herdado da superclasse animal.

O polimorfismo é uma consequência da herança e o principal conceito vem da sobrescrita dos métodos.
Gabarito: letra B.

[FCC 2016 TRT 14ª Região – Técnico Judiciário – Tecnologia da Informação – Questão 49] São, dentre outros, recursos essenciais em uma aplicação orientada a objetos para se obter polimorfismo:

[A] Herança e sobrescrita de métodos.
[B] Classes estáticas, com métodos protegidos.
[C] Interfaces, contendo métodos não abstratos e implementados.
[D] Classes abstratas, sem subclasses.
[E] Arrays unidimensionais ou multidimensionais.

Comentários:
O polimorfismo é uma consequência da herança e o principal conceito vem da sobrescrita dos métodos.

Gabarito: letra A.