Programação Orientada a Objetos

Herança

★ Conceito teórico

É a reutilização de código, sendo um tipo de mecanismo em que uma classe tem atributos e métodos de uma outra classe, sendo a Classe Base e a Classe Derivada.

- **Classe Base -** tem as características e métodos herdados por outras classes
- **Classe Derivada -** contém os atributos e métodos da Classe base.

★ Tipos de Heranças

- **Herança Simples -** quando a classe derivada herda de uma única classe base.
- **Herança Múltipla -** quando a classe herda de mais de uma classe base, porém nem todas as linguagens vão suportá-la devido a sua complexidade.

★ Exemplos práticos

```python
# Herança
class Itens_Jogo:
    def __init__(self, magia, forca, protecao):
        self.magia = magia
        self.forca = forca
        self.protecao = protecao

    def atingir(self):
        print("Você atingiu o seu inimigo!")

    def defender(self):
        print("Você conseguiu se denfender do seu inimigo!")

class Tijolo(Itens_Jogo):
    def fazer_barrulho(self):
        print("Você fez barrulho, conseguiu atrair o inimigo!")

class Garrafa(Itens_Jogo):
    def fazer_moloto(self):
        print("Você fez um molotov, jogue-o nos inimigos!")
```

Polimorfismo

★ Conceito teórico

É a modificação de objetos de diferentes classes que sejam tratados e compartilhem uma mesma interface ou Classe Base.

★ Tipos de Polimorfismo

- **Polimorfismo Estático** - acontece quando o programa é compilado, o mesmo método é implementado várias vezes na mesma classe, só com parâmetros diferentes, contudo a escolha do método a ser chamado vai depender do parâmetro passado.
- **Polimorfismo dinâmico** - acontece quando o programa é executado, com o mesmo método sendo implementado várias vezes nas classes derivadas, com os mesmo parâmetros. A escolha do método depende do objeto que o chama, que a classe pode implementá-lo.
- **Sobrecarga -** permite a criação de vários métodos com o mesmo nome, mas com diferentes parâmetros.
- **Substituição** - permite que uma subclasse forneça uma implementação específica de um método que já existe em sua superclasse.

★ Exemplos práticos

```python
# Polimorfismo
class Animal:
    def emitir_som(self):
        print("O animal faz um som.")

class Cachorro(Animal):
    def emitir_som(self):
        print(" Au au!")

class Gato(Animal):
    def emitir_som(self):
        print("Miau!")


animais = [Cachorro(), Gato(), Animal()]

for animal in animais:
    animal.emitir_som()
```

★ Vantagens
- **Reutilização de código:** as subclasses herdam os membros da superclasse, o que reduz a duplicação de código.
- **Extensibilidade:** novas funcionalidades podem ser adicionadas facilmente às subclasses, sem modificar a superclasse.
- **Organização:** a hierarquia de classes facilita a compreensão e a manutenção do código.

★ Diferenças

Herança explica a ação de herdar outra classe, enquanto o Polimorfismo explica o que os objetos ganham se tiverem uma Classe Base.

A herança permite que a classe derivada reutilize os atributos e métodos de uma classe base. Já o polimorfismo permite que métodos com o mesmo nome tenham funções diferentes.

★ Semelhanças

Ambos melhoram a organização do código e facilitam a reutilização de código. Utilizam o conceito de hierarquia entre classes.

Referências:

https://www.devmedia.com.br/conceitos-e-exemplos-heranca-programacao-orientada-a-objetos-parte-1/18579

https://www.estrategiaconcursos.com.br/blog/heranca-polimorfismo-programacao-orientada-objetos/

https://medium.com/lets-do-it-pl/what-is-polymorphism-and-what-are-the-advantages-of-it-8947a79125e8

https://imasters.com.br/carreira-dev/a-linguagem-de-programacao-orientada-a-objetos-e-seus-beneficios

https://dev.to/fabianoflorentino/programacao-orientada-a-objetos-heranca-1pc3

https://youtu.be/B9AOG7RFKzY?si=DJg2MmlPXfUOZVuI

https://youtu.be/SPpu-a65g1E?si=DGYKIzrAc4Mj7LZS