1- Polimorfismo:

. Um dos princípios fundamentais da Programação Orientada ao Objeto.

. É a capacidade de diferentes objetos, para um mesmo método ter uma resposta única para cada objeto.

. Permite escrever códigos mais genéricos e flexíveis.

. Permite lidar com uma variedade de tipos de objetos

Definição Formal de Polimorfismo

- O Polimorfismo é um dos princípios fundamentais da Programação Orientada a Objetos (POO).
- Em termos simples, o polimorfismo refere-se à capacidade de diferentes objetos responderem de maneira única a um mesmo método.
- Isso permite que você escreva código mais genérico e flexível, capaz de lidar com uma variedade de tipos de objetos.

Importância para POO:

. É essencial para criar sistemas de software flexíveis e adaptáveis.

. Permite escrever códigos sem precisar saber antecipadamente o tipo exato de objeto com o qual está lidando.

. O código pode confiar na capacidade do objeto de responder corretamente aos métodos necessários.

- O polimorfismo é essencial para criar sistemas de software flexíveis e adaptáveis.
- Ele permite que você escreva código que não precisa saber com antecedência o tipo exato de objeto com o qual está lidando.
- Em vez disso, o código pode confiar na capacidade do objeto de responder corretamente aos métodos necessários.

2 - Polimorfismo de Substituição e Polimorfismo de Sobrecarga

Tipos de Polimorfismo

Polimorfismo de Substituição ou Sobrescrita de Método (Override)

Polimorfismo de Sobrecarga de Método (Overload):

3 -

a) Polimorfismo de Sobrecarga.

b) Duas funções com o mesmo nome e para identificar qual tipo é por causa desses métodos terem parâmetros diferentes e estarem na mesma classe.

4 - Não, pois os parâmetros estão iguais e está em uma mesma classe, assim não é polimorfismo.

5 - Os dois métodos possuem o mesmo tipo na mesma posição,

causando um erro, pois estão na mesma classe.

6 -

```
public class Soma {
  public int doisValores(int a, int b) ——> m1
  {
      System.out.println("M1");
      return a + b;
  }
  public double doisValores(double a, int b) ——> m2
  {
      System.out.println("M2");
      return a + b;
  }
  public double doisValores(double a, double b) ——> m3
  {
      System.out.println("M3");
      return a + b;
  }
}
```

```
System.out.println(soma.doisValores(b, s)); m1
System.out.println(soma.doisValores(i, s)); m1
System.out.println(soma.doisValores(i, i)); m1
System.out.println(soma.doisValores(l, b)); m2
System.out.println(soma.doisValores(f, s)); m2
System.out.println(soma.doisValores(d, b)); m2
System.out.println(soma.doisValores(b, d)); m3
System.out.println(soma.doisValores(i, l)); m3
System.out.println(soma.doisValores(l, l)); m3
System.out.println(soma.doisValores(d, f)); m3
System.out.println(soma.doisValores(i, d)); m3
```