1

Polimorfismo

Modelagem II

Modelagem II

Principais conceitos da Programação Orientada a Objetos

Polimorfismo

Em UML, polimorfismo é um conceito da programação orientada a objetos que permite que objetos de diferentes classes respondam a mesma mensagem de maneiras distintas, adaptando-se às características de cada classe. É um conceito chave para a flexibilidade e reutilização de código em sistemas complexos.

Modelagem II

Principais conceitos da Programação Orientada a Objetos

Polimorfismo

Em UML, polimorfismo é um conceito da programação orientada a objetos que permite que objetos de diferentes classes respondam a mesma mensagem de maneiras distintas, adaptando-se às características de cada classe. É um conceito chave para a flexibilidade e reutilização de código em sistemas complexos.

O polimorfismo, derivado do grego "poly" (muitos) e "morphos" (formas), significa "muitas formas". Em UML, ele se manifesta através da capacidade de objetos de diferentes classes responderem de maneira única a uma mesma operação ou método.

Modelagem II

Principais conceitos da Programação Orientada a Objetos

Como o polimorfismo é representado em UML?

O polimorfismo em UML é geralmente representado através de:

- **Herança**

  As classes derivadas (subclasses) herdam atributos e métodos da classe base (superclasse), podendo sobrescrever ou estender esses métodos para se comportarem de forma diferente.

Modelagem II

Principais conceitos da Programação Orientada a Objetos

Como o polimorfismo é representado em UML?

O polimorfismo em UML é geralmente representado através de:

- **Interfaces**:

  Interfaces definem um contrato de comportamento, especificando um conjunto de métodos que as classes que implementam essa interface devem fornecer. Diferentes classes podem implementar a mesma interface, fornecendo implementações diferentes para os métodos definidos, demonstrando polimorfismo.

- **Classes abstratas**:

  Classes abstratas podem definir métodos abstratos que devem ser implementados pelas subclasses. Isso permite que subclasses com diferentes implementações compartilhem uma interface comum.

Modelagem II

Principais conceitos da Programação Orientada a Objetos

Como o polimorfismo é representado em UML?

Exemplo Prático

Imagine um sistema de gerenciamento de formas geométricas. Temos uma classe base chamada Forma com um método abstrato chamado **calcularArea().** Classes derivadas como **Circulo**, **Retangulo** e **Triangulo** podem implementar o método **calcularArea()** de maneiras diferentes, cada uma calculando a área de acordo com sua forma específica. Isso é polimorfismo em ação.

Modelagem II

Principais conceitos da Programação Orientada a Objetos

Benefícios do Polimorfismo em UML:

- **Flexibilidade:**

  Permite que o código trate objetos de diferentes classes de forma uniforme, através de uma interface comum.

- **Reutilização de código:**

  Facilita a reutilização de código, evitando a duplicação de lógica para diferentes tipos de objetos.

Modelagem II

Principais conceitos da Programação Orientada a Objetos

Benefícios do Polimorfismo em UML:

- **Extensibilidade:**

  Facilita a adição de novas classes e comportamentos ao sistema sem a necessidade de modificar o código existente.

- **Manutenção:**

  Torna o código mais fácil de manter e evoluir, pois as mudanças em uma classe geralmente não afetam outras classes que compartilham a mesma interface.

Referências

- Texto gerado por IA