1

Herança

Modelagem II

Modelagem II

Principais conceitos da Programação Orientada a Objetos

Herança

Em UML, herança é um relacionamento entre classes onde uma classe (subclasse/filha) herda atributos e métodos de outra classe (superclasse/pai). Isso permite a reutilização de código e a criação de uma hierarquia de classes, onde subclasses especializam a superclasse. A herança é representada por uma seta com ponta vazia apontando da subclasse para a superclasse.

Modelagem II

Principais conceitos da Programação Orientada a Objetos

Herança

Conceito:

- **Superclasse/Classe Pai:** A classe que fornece os atributos e métodos que serão herdados.
- **Subclasse/Classe Filha:** A classe que herda os atributos e métodos da superclasse.
- **Reutilização de Código:** A herança permite que atributos e métodos comuns sejam definidos em uma única classe (superclasse), evitando repetição em subclasses.
- **Hierarquia:** A herança cria uma estrutura hierárquica entre classes, facilitando a organização e compreensão do sistema.

Modelagem II

Principais conceitos da Programação Orientada a Objetos

Herança

Exemplo:

Imagine uma classe **Veículo como superclasse** e **Carro, Moto e Caminhão como subclasses**. A classe Veículo pode ter atributos como cor, marca, modelo e métodos como acelerar, frear. As subclasses herdariam esses atributos e métodos, e cada uma poderia adicionar seus próprios atributos específicos (por exemplo, númeroDePortas para Carro, cilindradas para Moto).

Modelagem II

Principais conceitos da Programação Orientada a Objetos

Herança

Representação em UML:

A herança é representada por uma seta com ponta vazia apontando da subclasse para a superclasse. Por exemplo, a seta ligando Carro a Veículo indica que Carro herda de Veículo.

Modelagem II

Principais conceitos da Programação Orientada a Objetos

Vantagens da Herança:

- **Reutilização de código:**
  Evita a duplicação de código em subclasses, economizando tempo de desenvolvimento.

- **Organização hierárquica:**
  Cria uma estrutura clara e organizada, facilitando a compreensão do sistema.

- **Facilidade de manutenção:**
  Alterações na superclasse são refletidas em todas as subclasses, simplificando a manutenção.

- **Flexibilidade:**
  Permite que as subclasses adicionem novos atributos e métodos, especializando o comportamento da superclasse.

Referências

- Texto gerado por IA