FACULDADE ESTÁCIO DE SÁ

CURSO: DESENVOLVIMENTO FULL STACK

3º SEMESTRE – MATRÍCULA 202302595341

ALAIM ALMEIDA DE OLIVEIRA

Criação das Entidades e Sistema de Persistência

Salvador – BA

2024

Herança

A herança é um conceito importante e fundamental na programação orientada a objetos, pois é amplamente utilizada na linguagem de programação Java.

É um mecanismo pelo qual uma classe(subclasse) pode herdar atributos e métodos de outra classe (classepai ou superclasse). A subclasse pode então, estender e modificar o comportamento da superclasse, permitindo a reutilização de código e a criação de hierarquias de classes.

Na utilização da herança no Java, uma das maiores vantagens seria no quesito de reutilização de código, pois a superclasse compartilha todo o seu comportamento, atributos para as outras classes, evitando a repetição de código, promovendo a reutilização e a modularidade do mesmo.

Com a herança, temos o polimorfismo, onde um objeto de uma subclasse pode ser tratado como um objeto da superclasse. Isso facilita a escrita de código genérico e flexível. Além do reaproveitamento de código, todas as classes herdadas, ficaram organizadas em uma hierarquia, representando relações de especialização e generalização entre objetos do mundo real. Isso facilita a compreensão e a manutenção do código.

Desvantagens da Herança em Java

Ao utilizar a herança, é criado um acoplamento forte entre a subclasse e superclasse, o que pode tornar o código mais difícil de entender e manter. Mudanças na superclasse podem afetar as subclasses e vice-versa.

Quando se trata de herança múltiplas, o Java não tem suporte.

Devido a hierarquias profundas ao ser utilizado a herança, o código pode se tornar complexo e difícil de gerenciar, pois pode levar a uma estrutura excessivamente complicada e difícil de entender.

A herança em Java é uma poderosa ferramenta de programação orientada a objetos que oferece vantagens significativas, como reutilização de código, extensibilidade e polimorfismo. No entanto, é importante usar a herança com cuidado e considerar suas potenciais desvantagens, como acoplamento forte e herança frágil, ao projetar sistemas de software.

Interface Serializable

A interface Serializable em Java indica que uma classe pode ser convertida em uma sequência de bytes para armazenamento ou transmissão. Essa conversão é essencial para persistir objetos em arquivos binários, permitindo que o Java os armazene e reconstrua quando necessário.

Paradigma Funcional e API Stream

A API Stream em Java, desde o Java 8, utiliza conceitos do paradigma funcional para operar em coleções de dados de maneira mais eficiente e concisa. Ela oferece operações de alto nível, como map, filter e reduce, que simplificam a manipulação dos elementos da coleção de forma declarativa, sem a necessidade de iteração manual. Além disso, as operações em Stream são geralmente imutáveis, o que significa que não modificam o estado original da coleção, tornando o código mais seguro e concorrente. A API Stream também suporta a composição de funções, permitindo encadear várias operações em uma única expressão para criar pipelines de processamento de dados eficientes e legíveis.

Padrão de Desenvolvimento na Persistência de Dados em Arquivos em Java

Para persistir dados em arquivos em Java, é comum seguir o padrão de desenvolvimento chamado Object Serialization. Este padrão envolve a serialização de objetos em bytes para armazenamento em arquivos e inclui os seguintes passos:

Implementar a interface Serializable nas classes desejadas.

Usar classes como ObjectInputStream e ObjectOutputStream para serializar e desserializar objetos para e a partir de arquivos.

Lidar com exceções apropriadas ao trabalhar com operações de entrada e saída de dados.

Garantir que todas as classes usadas na serialização tenham uma versão única para evitar problemas de compatibilidade.

Esse padrão oferece uma maneira eficiente e flexível de persistir objetos em arquivos binários em Java.