=======================================================

***Desenvolvimento de Aplicações para WEB***

=======================================================

# **Persistência**

A Persistência de dados é um meio para que um aplicativo persista e recupere informações de um sistema de armazenamento. Sem essa capacidade, os dados só existiriam na RAM, e seriam perdidos quando a RAM parasse.



# **JDBC**

Possibilita que uma aplicação construída na linguagem consiga acessar um banco de dados configurado **Local** ou **Remotamente.**

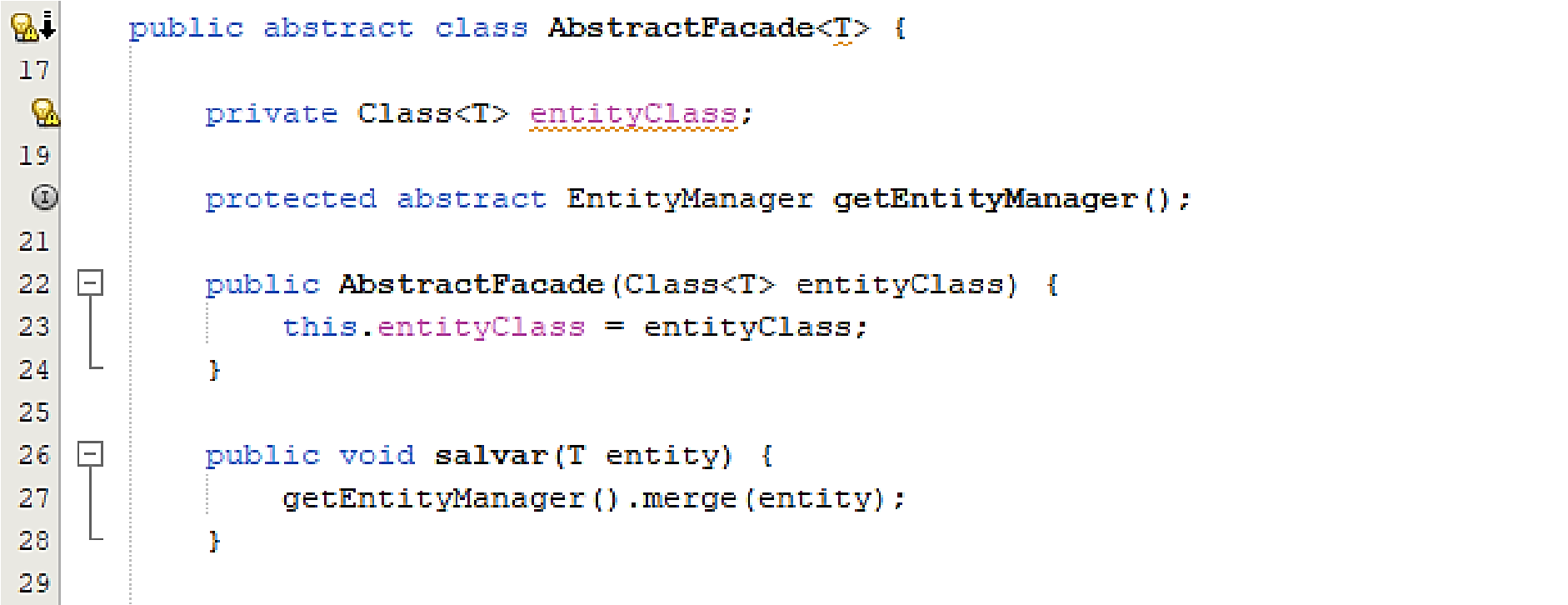
# **Hibernante com Persistência**

O Hibernante simplifica a persistência de dados em **Java**, fornecendo um mapeamento transparente entre Objetos **Java** e **Tabelas** de **Banco de Fados**, gerenciamento do Ciclo de Vida dos Objetos, recursos avançados de consulta e suporte a transações.

Simplificando o processo de armazenar e recuperar Objetos **Java** em um banco de dados relacional, abstraindo a complexidade do acesso ao banco de dados e fornecendo um mecanismo eficiente para a persistência dos dados.

# **Classe Genérica**

As Classes Genéricas encapsulam operações que não são específicas de um determinado tipo de dados. Permitem escrever códigos mais seguros e fáceis de ler pelo fato de utilizarem menos conversões de um Objeto em outro.





Também é responsável por realizar o CRUD, Sigla que consiste nas quatro operações básicas de todo banco de dados/aplicação de cadastro, que são: Create, Read, Update, Delete.

# **EJB**

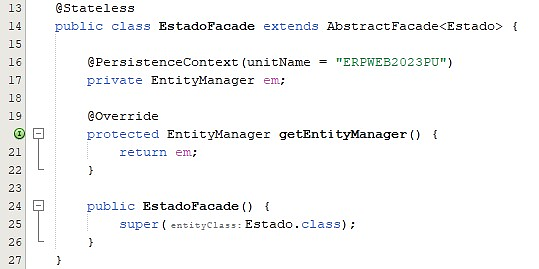
O **EJB**, ou **Enterprise JavaBeans**, é um componente que roda em um container de servidor de aplicação. É utilizado para facilitar o desenvolvimento de aplicações **Java**, disponibilizando componentes distribuídos, transacionais, seguros e portáteis.

Em resumo ele Instancia o Objeto.

@EJB: Incrementa as funcionalidades do **Enterprise JavaBeans** na aplicação.

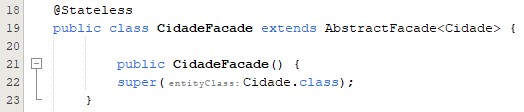
# **Facade**

O Facade é um padrão estrutural que atua na redução da complexidade da interface em um sistema com várias classes, bibliotecas ou frameworks, além de mover dependências indesejadas para um só local.



# **Método Construtor**

Os Métodos Construtores são responsáveis por criar o Objeto em memória, instanciando a classe que foi definida. Ele deve possuir o mesmo nome da classe, correspondendo às letras maiúsculas e minúsculas.



# **Converter**

O Converter permite a transformação de Objeto em String e a transformação de String em Objeto.

Objeto para String 🡪 (getAsString)

String para Objeto 🡪 (getAsObject)

# **Uso das Classes (Herança)**

Para usar a Classe tem é necessário fazer um Filho (Facade), e utilizar o getEntityManager, sendo capaz de gerenciar adequadamente os métodos e operações relacionados às entidades Pai e Filhas em um relacionamento de herança, independentemente da estratégia de mapeamento adotada.

Faz se uso de super para chamar o construtor da Classe Pai.

**Ex:**

