INTRODUÇÃO AO JDBC E O PADRÃO FACTORY

* Para acessar o banco de dados, precisamos de um driver
  + Um driver nada mais é do que uma biblioteca (JAR)
* JDBC significa Java DataBase Conectivity
  + JDBC define uma camada de abstração entre a sua aplicação e o driver do banco de dados
  + Essa camada possui, na sua grande maioria, interfaces que o driver implementa
* Para abrir uma conexão, devemos usar o método getConnection, da classe DriverManager
  + O método getConnection recebe uma string de conexão JDBC, que define a URL, usuário, senha, etc

EXECUTANDO COMANDOS SQL NO JAVA

* Para simplificar e encapsular a criação da conexão, devemos usar uma classe ConnectionFactory
  + A classe ConnectionFactory segue o padrão de criação Factory Method
  + O Factory Method encapsula a criação de um objeto
* Para executar um comando SQL, podemos usar a interface java.sql.Statement
  + O método execute envia o comando para o banco de dados
  + Dependendo do comando SQL, podemos recuperar a chave primária ou os registros selecionados

EVITANDO SQL INJECTION

* Ao executar SQL como Statement, temos um risco de segurança, chamado de SQL Injection
  + SQL Injection nada mais é do que passar um novo comando SQL como parâmetro
* Para evitar SQL Injection, devemos usar a interface PreparedStatement
  + Diferentemente do Statement, o PreparedStatement trata (sanitiza) cada parâmetro do comando SQL

CONTROLE DE TRANSAÇÃO

* O banco de dados oferece um recurso chamado de transação, para juntar várias alterações como unidade de trabalho
  + Se uma alteração falha, nenhuma alteração é aplicada (é feito um rollback da transação)
  + Todas as alterações precisam funcionar para serem aceitas (é feito um commit)
* commit e rollback são operações clássicas de transações
* Para garantir o fechamento dos recursos, existe no Java uma cláusula try-with-resources
  + O recurso em questão deve usar a interface Autoclosable

ESCALABILIDADE COM POOL DE CONEXÕES

* É boa prática usar um pool de conexões
* Um pool de conexões administra/controla a quantidade de conexões abertas
  + Normalmente tem um mínimo e máximo de conexões
* Como existe uma interface que representa a conexão (java.sql.Connection), também existe uma interface que representa o pool de conexões (javax.sql.DataSource)
* C3PO é uma implementação Java de um pool de conexão

CAMADA DE PERSISTÊNCIA COM DAO

* Para cada tabela de domínio, temos uma classe de domínio
  + Por exemplo, a tabela produtos tem uma classe Produto associada
  + Objetos dessa classe representa um registro na tabela
* Para acessar a tabela, usaremos um padrão chamado Data Access Object (DAO)
  + Para cada classe de domínio, existe um DAO. Por exemplo, a classe Produto possui um ProdutoDao
  + Todos os métodos JDBC relacionados com o produto devem estar encapsulados no ProdutoDao

EVITANDO QUERIES N+1

* Que quando temos um relacionamento, é preciso ter cuidado para não cair no problema de queries N + 1
  + N + 1 significa executar uma query e mais uma nova query (N) para cada relacionamento
  + Queries N + 1 podem gerar um problema no desempenho
  + Queries N + 1 podem ser evitadas através de joins no SQL
* A criar a nossa própria camada de persistência