Ciências ULisboa
Faculdade de Ciências da Universidade de Lisboa

2019/2020

Construção de Sistemas de Software

SalesSys

Gateways — RDGW e TDGW

Ciências ULisboa | Informática

CSS (2019/2020)
Caderno de Exercícios
Folha 3

Padrões para organização dos dados

1. Suponha que se pretende desenhar a camada de acesso aos dados do SalesSys recorrendo ao padrão *Row Data Gateway* e que essa camada vai ser responsável por realizar a comunicação da aplicação com o SGBD escolhido — Derby — através de um conector JDBC. Adicionalmente deve ter em conta que se pretende usar esta camada por baixo dos *transaction scripts* concebidos anteriormente (folha 1).

   (a) Apresente um diagrama que explique a organização em camadas desta solução do SalesSys.

   (b) Que vantagens temos por ter todo o código responsável pelo acesso à base de dados numa camada separada?

   (c) Explique em que consiste o padrão *Row Data Gateway* focando a sua explicação na forma como é feita a abstração da base de dados e nas responsabilidades atribuídas a cada objeto do padrão.

   (d) Que vantagens advém de ter o código responsável pelas pesquisas numa tabela numa classe separada?

   (e) Identifique as classes *gateway* de que vai precisar, os seus atributos e métodos.

   (f) Desenhe uma classe `DataSource` com uma única instância que abstrai e encapsula a ligação à base de dados (que se considera ser a fonte única de dados).

   (g) Tendo em conta os atributos definidos para a classe `CustomerRowDGW`, desenhe o comportamento do método `insert()`. Identifique as constantes que devem ser definidas nesta classe para confinar ao máximo as dependências do modelo físico de dados.

   (h) Complete a classe `CustomerRowDGW` desenhando o comportamento dos métodos de classe

   - `getCustomerByVATNumber(int vat):CustomerRowDGW`
   - `getCustomerById(int saleId):CustomerRowDGW`.

   (i) Defina os atributos da classe `SaleRowDGW` e desenhe o comportamento do método `insert()`. Identifique as constantes que devem ser definidas nesta classe para confinar ao máximo as dependências do modelo físico de dados.

   (j) Complete a classe `SaleRowDGW` desenhando o comportamento do método de classe `getSaleById(int saleId):SaleRowDGW`.

   (k) Defina os atributos da classe `SaleProductRowDGW` e desenhe o comportamento do método `getProductSalesBySaleId(int id): Iterable<SaleProductRowDGW>`. Identifique as constantes que devem ser definidas nesta classe para confinar ao máximo as dependências do modelo físico de dados.

   (l) Repita as tarefas anteriores para as restantes classes da camada de acesso aos dados.

   (m) Suponha que quer combinar a camada de negócio organizada seguindo o *Transaction Script* desenhada no exercício 2 da folha 1 com a camada de acesso aos dados do exercício anterior. Identifique, no desenho da operação `addProductToSale`, que alterações são necessárias e que serviços adicionais é preciso que a classe *DataSource* forneça.

2. Suponha que se pretende desenhar a camada de acesso aos dados do SalesSys recorrendo ao padrão *Table Data Gateway* e que essa camada vai ser responsável por realizar a comunicação da aplicação com o SGBD escolhido — Derby — através de um conector JDBC. Apesar de existir atualmente uma única fonte de dados, o desenho da camada deve prever a possibilidade de serem usadas várias fontes de dados. Adicionalmente o desenho da camada deve ter em conta que se pretende usar esta camada por baixo dos *table modules* concebidos anteriormente (folha 2).

   (a) Explique em que consiste o padrão *Table Data Gateway* focando a sua explicação na forma como é feita a abstração da base de dados e nas responsabilidades atribuídas a cada objeto do padrão.

   (b) Identifique as classes *gateway* de que vai precisar, os seus atributos e métodos.

   (c) Desenhe uma classe *Persistence* cujos objetos abstraem formas de realizar a persistência e as respetivas fontes dos dados, e que são responsáveis por criar os *gateways* para as tabelas e suporte a execução de transações. Sugestão: Adapte a classe *DataSource* definida no exercício 1 de forma a poder ter várias instâncias.

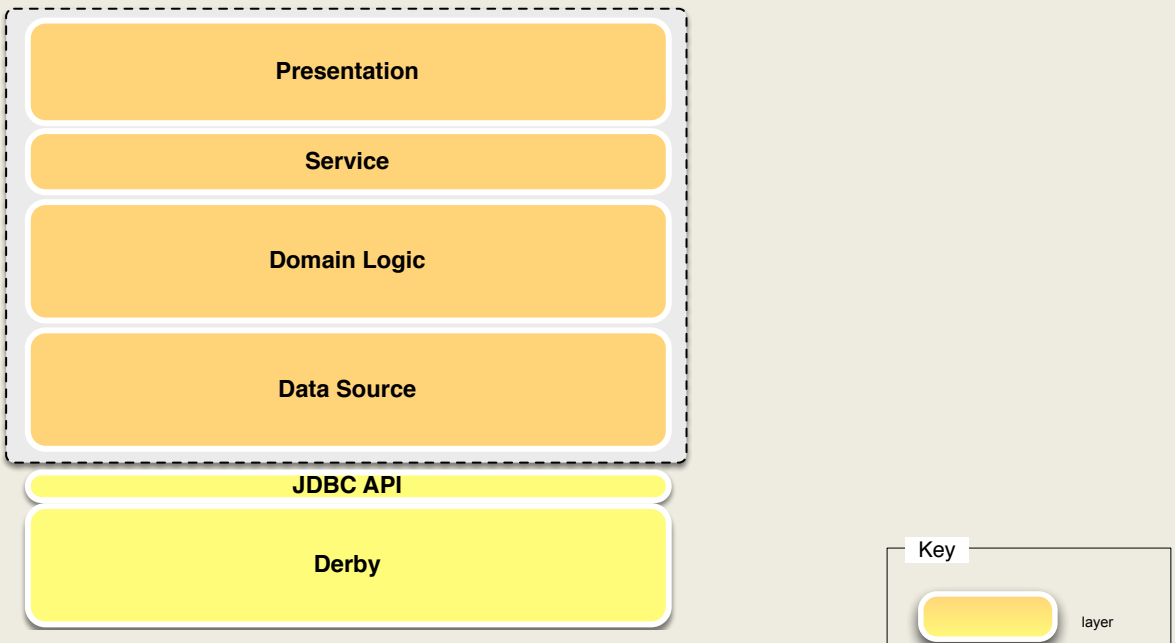
   (d) Defina os atributos da classe `CustomerTableDGW` e desenhe o comportamento dos métodos

Exercícios

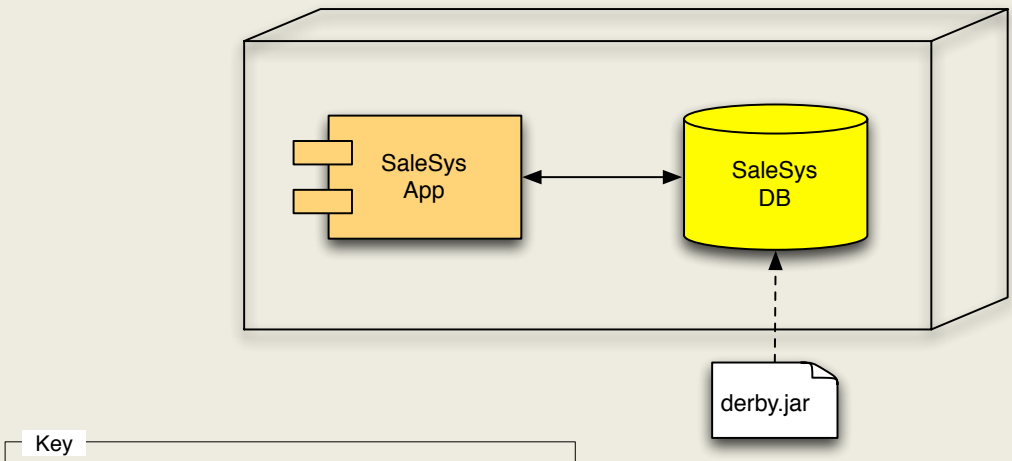
Suponha que se pretende desenhar a camada de acesso aos dados do SalesSys recorrendo ao padrão *Row Data Gateway* e que essa camada vai ser responsável por realizar a comunicação da aplicação com o SGBD escolhido — Derby — através de um conector JDBC. Adicionalmente deve ter em conta que se pretende usar esta camada por baixo dos *transaction scripts* concebidos anteriormente (folha 1).

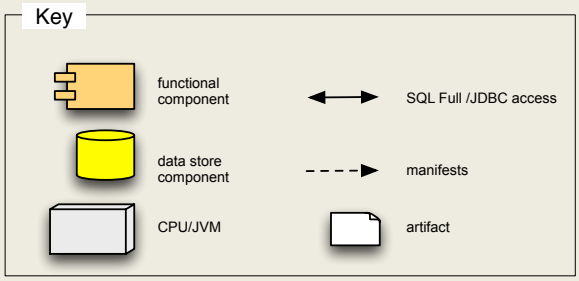
(a) Apresente um diagrama que explique a organização em camadas desta solução do SalesSys.

SalesSysApp: Organização em camadas (do código)



SalesSys: Arquitetura (BD embebida)





Apache Derby

Apache Derby is a relational database management system developed by the Apache Software Foundation that can be embedded in Java programs and used for online transaction processing. It has a 3.5 MB disk-space footprint. Apache Derby is

Exercícios

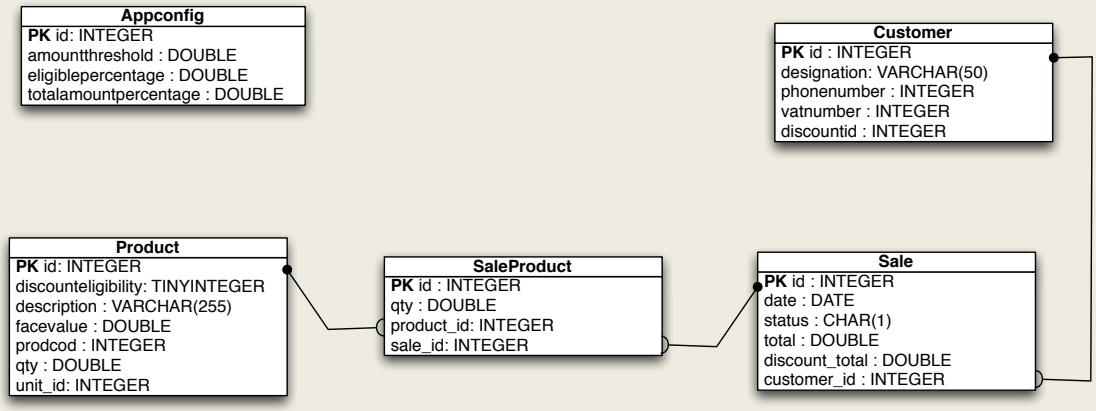
(b) Que vantagens temos por ter todo o código responsável pelo acesso à base de dados numa camada separada?

- O código que depende do JDBC (e q envolve SQL), do esquema da BD e escolha do SGDB está todo nesta camada, separado portanto do código que trata da lógica de negócio
- Facilita a divisão de trabalho
- Facilita o trabalho dos DBAs (afinar e fazer evoluir a base de dados)
- Facilita mudanças ao nível de como é feita a persistência (mudança de SGBD ou mesmo a forma como é feita a persistência)

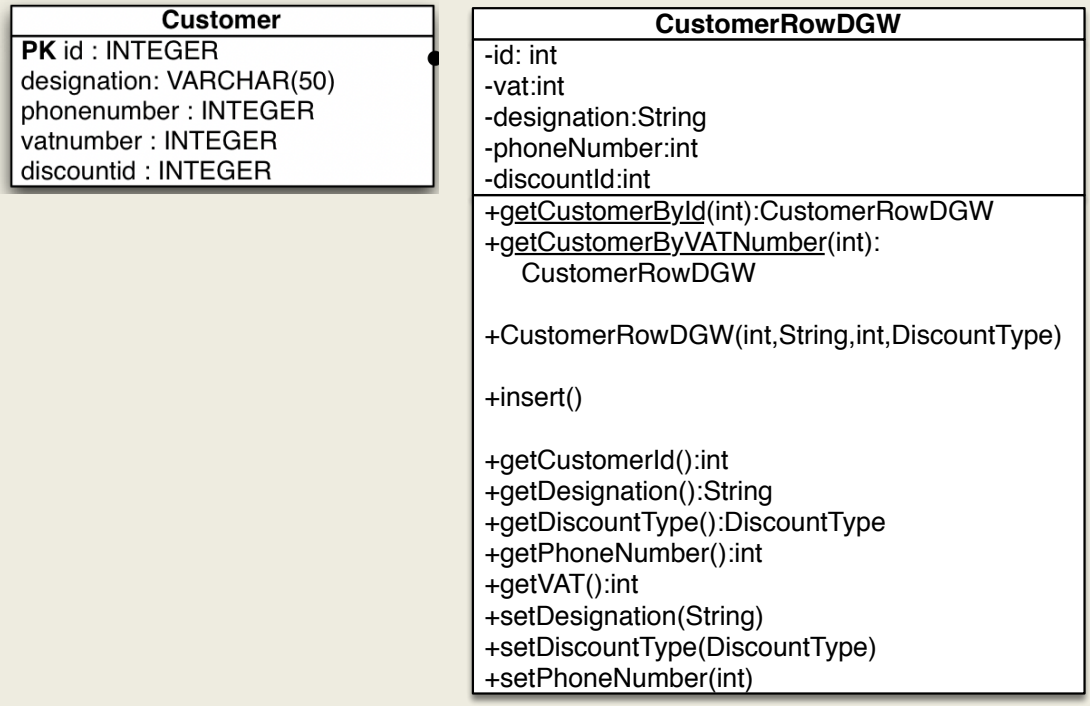
Exercícios

(c) Explique em que consiste o padrão *Row Data Gateway* focando a sua explicação na forma como é feita a abstração da base de dados e nas responsabilidades atribuídas a cada objeto do padrão.

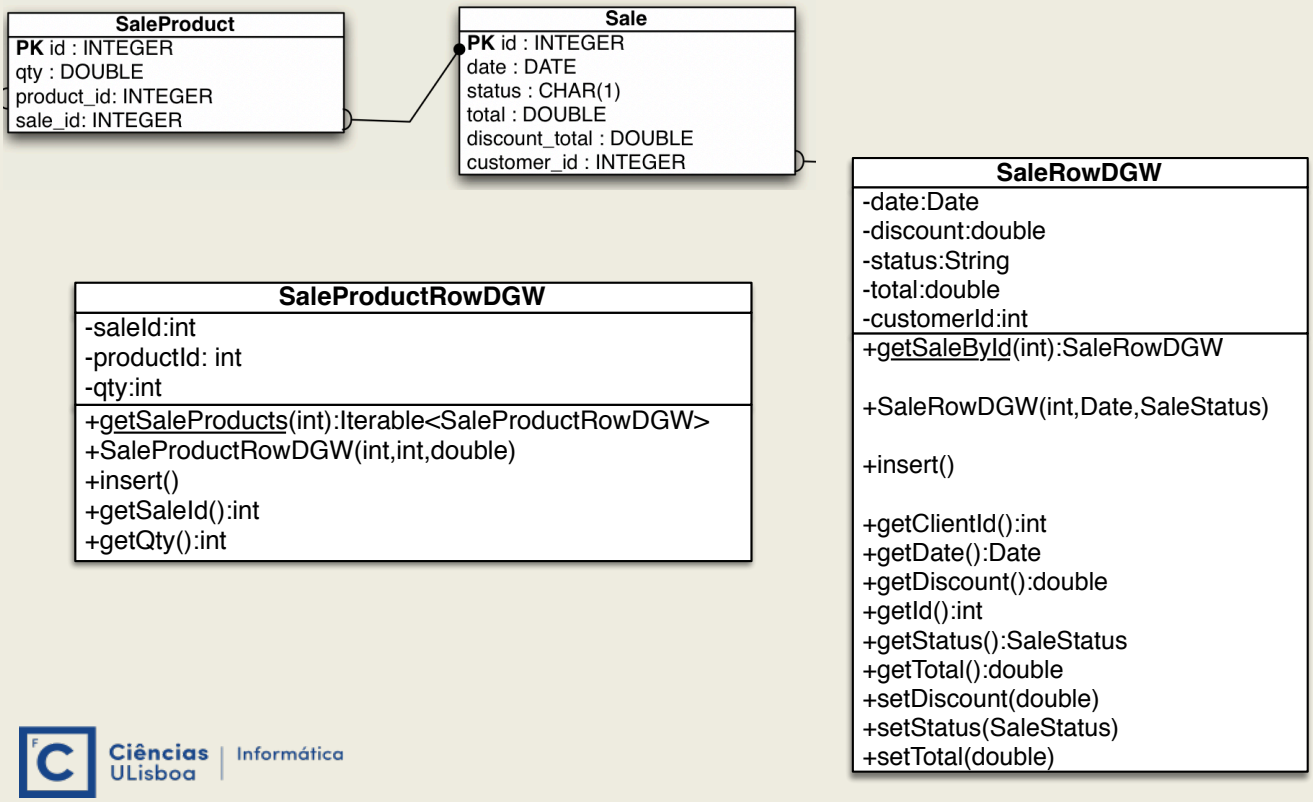
(d) Identifique as classes *gateway* de que vai precisar, os seus atributos e métodos.



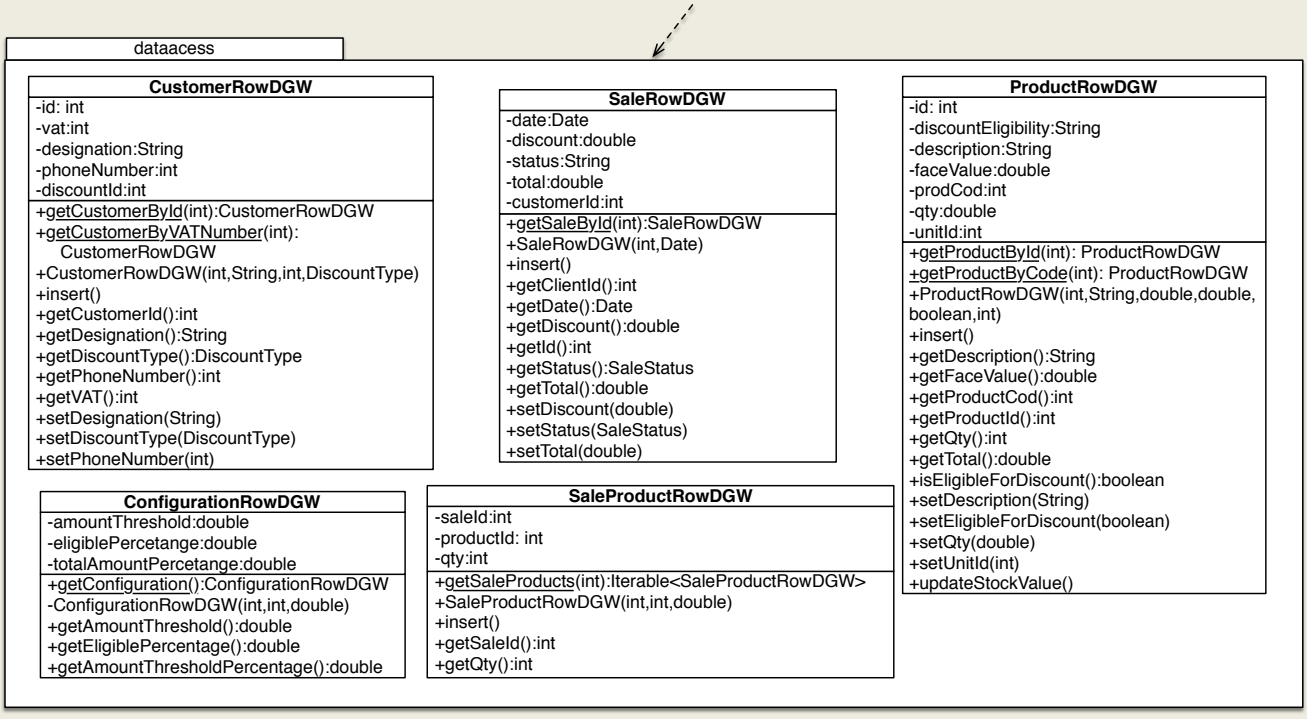
Camada de Acesso aos Dados com RDGW



Camada de Acesso aos Dados com RDGW

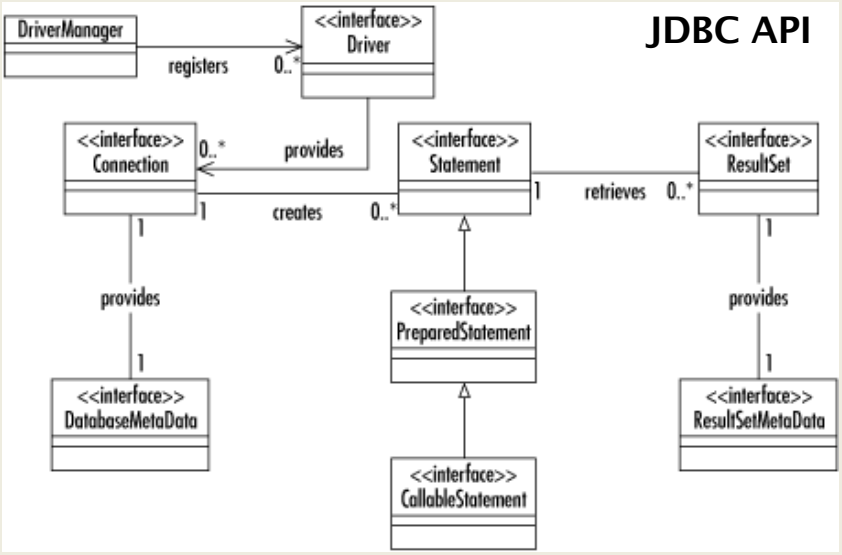


Camada de Acesso aos Dados com RDGW



Exercícios

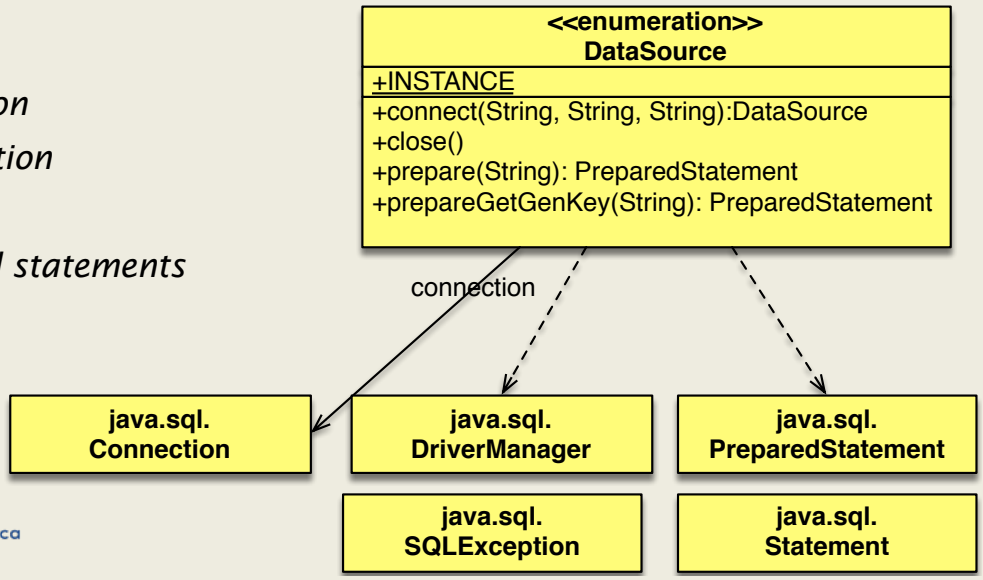
(e) Desenhe uma classe `DataSource` com uma única instância que abstrai e encapsula a ligação à base de dados.



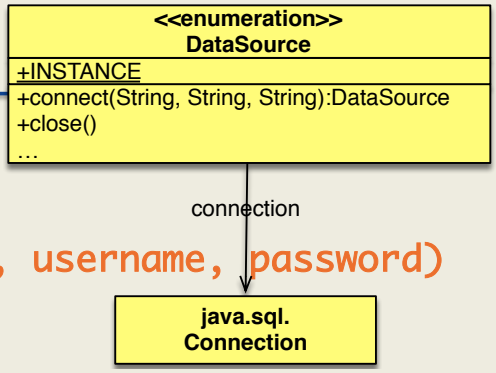
Fonte de Dados e JDBC

Classe `DataSource` com uma única instância que

- abstrai a ligação à base de dados (a fonte de dados) através de um conector JDBC
- é responsável por estabelecer a ligação e por todas as ações sobre essa ligação
  - Abrir *connection*
  - Fechar *connection*
  - Criar *prepared statements*



Fontes de Dados e JDBC



Na classe `DataSource`

- o método `connect` usa

```
DriverManager.getConnection(url, username, password)
```

- o método `close` usa

```
connection.close()
```

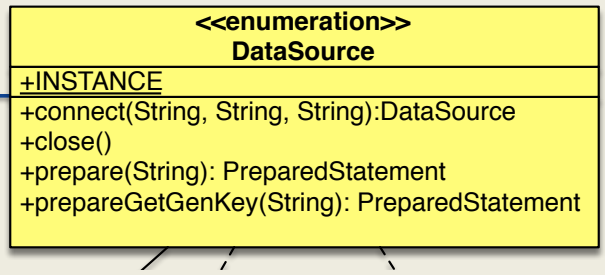
- **Na classe** `StartUp` **do SaleSys**
  - no arranque da aplicação

```
DataSource.INSTANCE.connect(
    "jdbc:derby:data/derby/cssdb;create=false",
    "SaleSys",
    "");
```

- e no fecho da aplicação

```
DataSource.INSTANCE.close()
```


Fontes de Dados e JDBC



Na classe `DataSource`

- Os métodos que permitem obter objetos `PreparedStatement` a partir de Strings com frases SQL usam

```
connection.prepareStatement(String sql)
connection.prepareStatement(String sql,
                            int autoGeneratedKeys)
```

e constantes

```
Statement.RETURN_GENERATED_KEYS
Statement.NO_GENERATED_KEYS
```

Exercícios

(f) Defina os atributos da classe `CustomerRowDGW` e desenhe o comportamento do método `insert()`.

(g) Complete a classe `CustomerRowDGW` desenhando o comportamento dos métodos de classe

`getCustomerByVATNumber(int vat): CustomerRowDGW`

`getCustomerById(int saleId): CustomerRowDGW.`

SalesSys: Esquema da base de dados (Derby)

```
CREATE TABLE CUSTOMER (
        ID INTEGER GENERATED BY DEFAULT AS IDENTITY (START WITH 1)
                PRIMARY KEY NOT NULL,
        DESIGNATION VARCHAR(50) NOT NULL,
        PHONENUMBER INTEGER,
        VATNUMBER INTEGER NOT NULL UNIQUE,
        DISCOUNT_ID INTEGER NOT NULL
)

CREATE TABLE PRODUCT (
        ID INTEGER GENERATED BY DEFAULT AS IDENTITY (START WITH 1)
                PRIMARY KEY NOT NULL,
        DESCRIPTION VARCHAR(255),
        DISCOUNTELIGIBILITY SMALLINT DEFAULT 0,
        FACEVALUE DOUBLE,
        PRODCOD INTEGER ,
        QTY DOUBLE CHECK (QTY>=0),
        UNIT_ID INTEGER
)
```

Classe *CustomerRDGW*

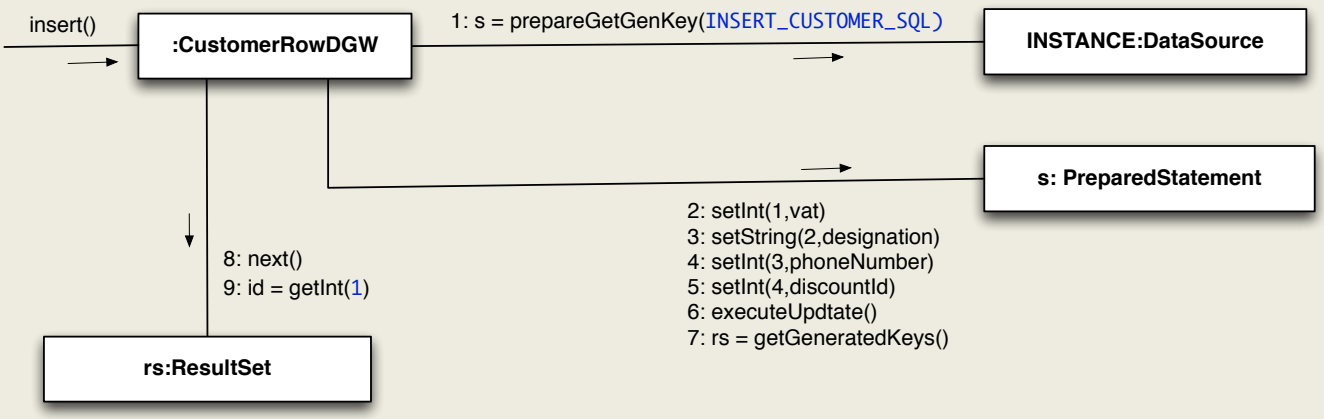
Localizar o mais possível as dependências do modelo físico da fonte dos dados, definindo constantes

CustomerRowDGW
-id: int -vat:int -designation:String -phoneNumber:int -discountId:int
+getCustomerById(int):CustomerRowDGW +getCustomerByVATNumber(int): CustomerRowDGW +CustomerRowDGW(int,String,int,DiscountType) +insert() +getCustomerId():int +getDesignation():String +getDiscountType():DiscountType +getPhoneNumber():int +getVAT():int +setDesignation(String) +setDiscountType(DiscountType) +setPhoneNumber(int)

```
CREATE TABLE CUSTOMER (
ID INTEGER GENERATED BY DEFAULT AS IDENTITY
(START WITH 1) PRIMARY KEY NOT NULL,
DESIGNATION VARCHAR(50) NOT NULL,
PHONENUMBER INTEGER,
VATNUMBER INTEGER NOT NULL UNIQUE,
DISCOUNT_ID INTEGER NOT NULL
)
```

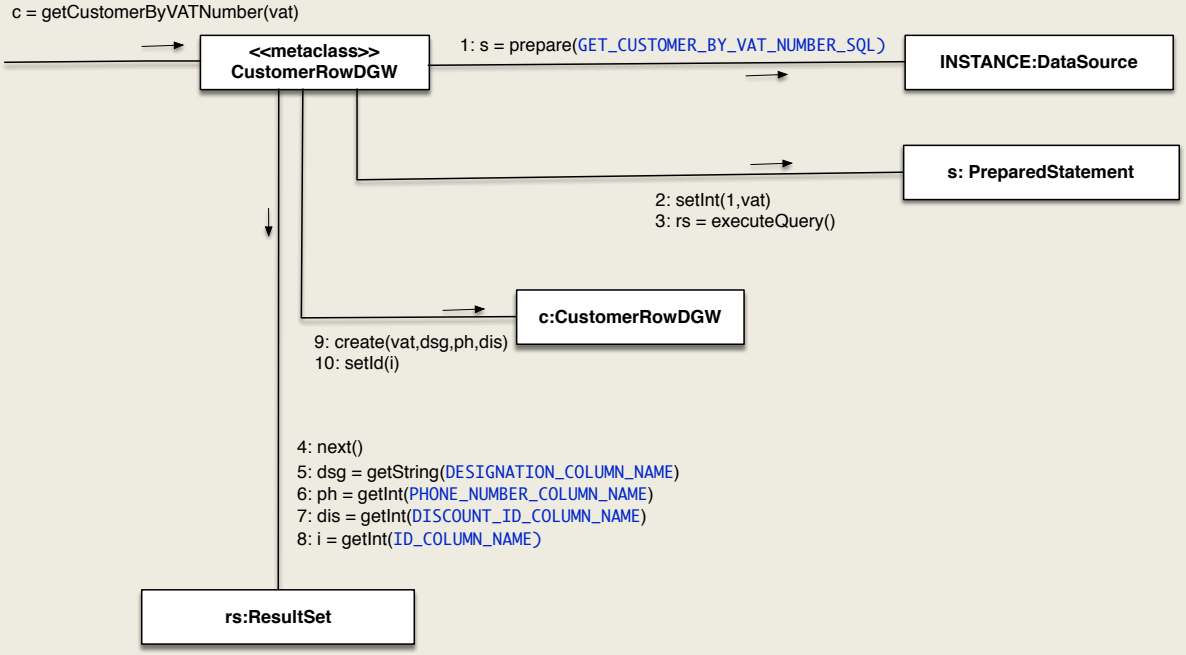
```
TABLE_NAME = "customer";

ID_COLUMN_NAME = "id";
VAT_NUMBER_COLUMN_NAME = "vatnumber";
DESIGNATION_COLUMN_NAME = "designation";
PHONE_NUMBER_COLUMN_NAME = "phonenumber";
DISCOUNT_ID_COLUMN_NAME = "discount_id";
```


Classe *CustomerRDGW: insert()*

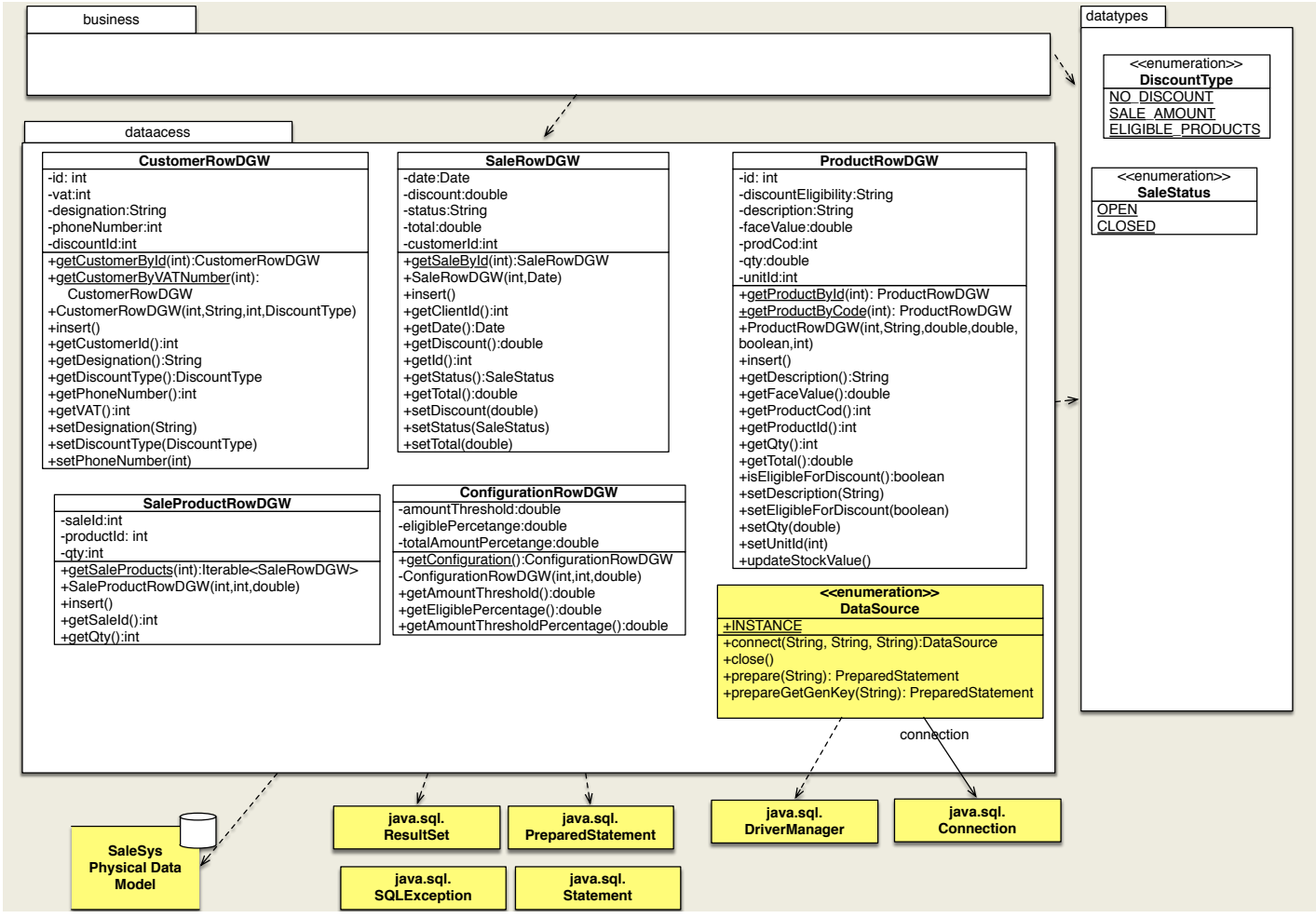



```
INSERT_CUSTOMER_SQL = "insert into " + TABLE_NAME + " " + "(" + ID_COLUMN_NAME + ", " + VAT_NUMBER_COLUMN_NAME + ", " + DESIGNATION_COLUMN_NAME + ", " +
                      PHONE_NUMBER_COLUMN_NAME + ", " + DISCOUNT_ID_COLUMN_NAME + ") " + "values (DEFAULT, ?, ?, ?, ?)";
```

Classe *CustomerRDGW: getCustomerByVATNumber(vat)*




```
GET_CUSTOMER_BY_VAT_NUMBER_SQL = "select " + ID_COLUMN_NAME + ", " + VAT_NUMBER_COLUMN_NAME + ", " + DESIGNATION_COLUMN_NAME + ", " +
                                 PHONE_NUMBER_COLUMN_NAME + ", " + DISCOUNT_ID_COLUMN_NAME + " " +
                                "from " + TABLE_NAME + " " + "where " + VAT_NUMBER_COLUMN_NAME + " = ?";
```



Exercícios

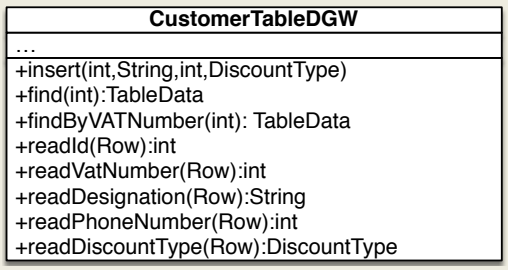
Suponha que se pretende desenhar a camada de acesso aos dados do SalesSys recorrendo ao padrão *Table Data Gateway* e que essa camada vai ser responsável por realizar a comunicação da aplicação com o SGBD escolhido — Derby — através de um conector JDBC. Apesar de existir atualmente uma única fonte de dados, o desenho da camada deve prever a possibilidade de serem usadas várias fontes de dados. Adicionalmente o desenho da camada deve ter em conta que se pretende usar esta camada por baixo dos *table modules* concebidos anteriormente (folha 2).

(a) Explique em que consiste o padrão *Table Data Gateway* focando a sua explicação na forma como é feita a abstração da base de dados e nas responsabilidades atribuídas a cada objeto do padrão.

(b) Identifique as classes *gateway* de que vai precisar, os seus atributos e métodos.

Classe *CustomerTableDGW*

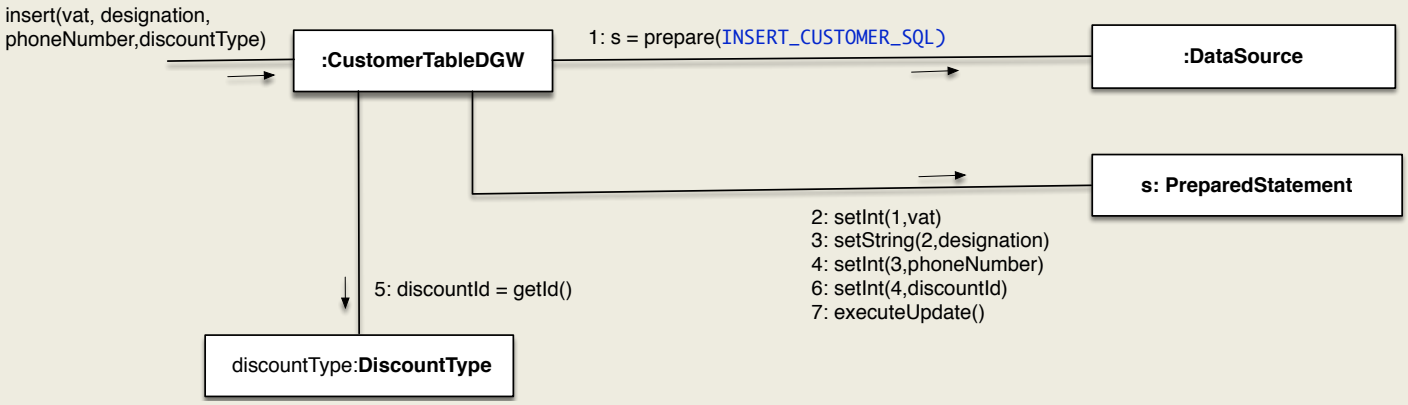
Localizar o mais possível as dependências do modelo físico da fonte dos dado, definindo constantes




```
CREATE TABLE CUSTOMER (
ID INTEGER GENERATED BY DEFAULT AS IDENTITY
(START WITH 1) PRIMARY KEY NOT NULL,
DESIGNATION VARCHAR(50) NOT NULL,
PHONENUMBER INTEGER,
VATNUMBER INTEGER NOT NULL UNIQUE,
DISCOUNT_ID INTEGER NOT NULL
)
```

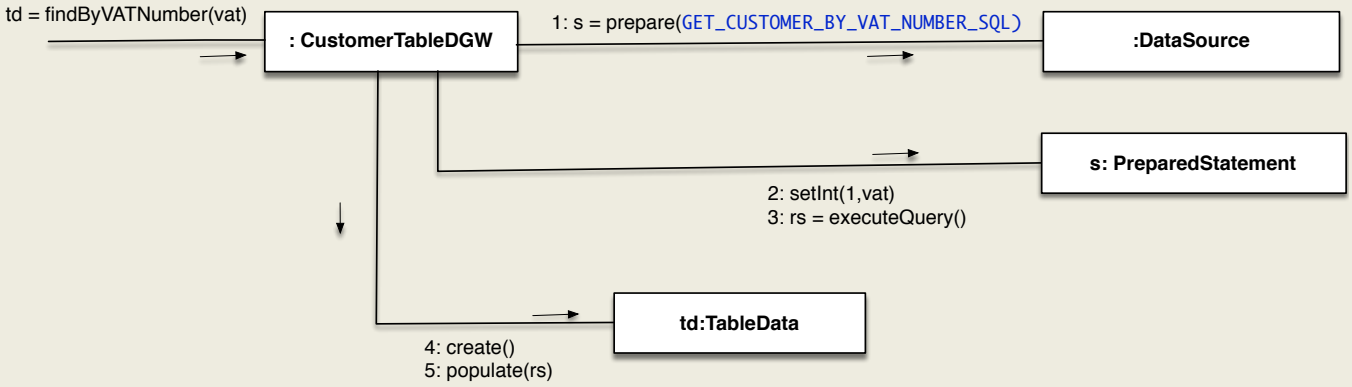
```
TABLE_NAME = "customer";

ID_COLUMN_NAME = "id";
VAT_NUMBER_COLUMN_NAME = "vatnumber";
DESIGNATION_COLUMN_NAME = "designation";
PHONE_NUMBER_COLUMN_NAME = "phonenumber";
DISCOUNT_ID_COLUMN_NAME = "discount_id";
```

Classe *CustomerTableDGW: insert(…)*




```
INSERT_CUSTOMER_SQL = "insert into " + TABLE_NAME + " " + "(" + ID_COLUMN_NAME + ", " + VAT_NUMBER_COLUMN_NAME + ", " + DESIGNATION_COLUMN_NAME + ", " +
                      PHONE_NUMBER_COLUMN_NAME + ", " + DISCOUNT_ID_COLUMN_NAME + ") " + "values (DEFAULT, ?, ?, ?, ?)";
```

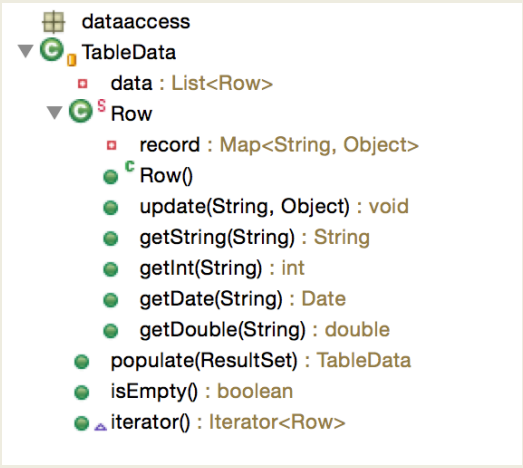

Classe *CustomerTableDGW: findByVATNumber(…)*




```
GET_CUSTOMER_BY_VAT_NUMBER_SQL = "select " + ID_COLUMN_NAME + ", " + VAT_NUMBER_COLUMN_NAME + ", " + DESIGNATION_COLUMN_NAME + ", " +
                                 PHONE_NUMBER_COLUMN_NAME + ", " + DISCOUNT_ID_COLUMN_NAME + " " +
                                "from " + TABLE_NAME + " " + "where " + VAT_NUMBER_COLUMN_NAME + " = ?";
```

Table Data

- `TableData` é uma classe cujos objetos representam dados tabulares (*record sets*)



- É um tipo semelhante a `java.sql.ResultSet` mas mais abstrato

```
public interface ResultSet
extends Wrapper, AutoCloseable
```

A table of data representing a database result set, which is usually generated by executing a statement that queries the database.

SalesSys: Solução com *Table Module* + *Table Data Gateway*