@Override

**public** Order getOrderDetails(String orderId) **throws** Exception {

Connection connection = **null**;

Order order = **null**;

**try** {

*exceptionProps* = PropertiesLoader.*loadProperties*(***EXCEPTION\_PROPERTIES\_FILE***);

*goProps* = PropertiesLoader.*loadProperties*(***GO\_PROPERTIES\_FILE***);

DbConnection.*getInstance*();

connection = DbConnection.*getConnection*();

PreparedStatement statement = connection.prepareStatement(QuerryMapper.***IS\_ORDER\_PRESENT***);

statement.setString(1, orderId);

ResultSet resultSet = statement.executeQuery();

resultSet.next();

order = resultSet.getObject(1, Order.**class**);

String orderID=order.getOrderId();

String userID=order.getUserId();

} **catch** (DatabaseException e) {

GoLog.***logger***.error(*exceptionProps*.getProperty("orderId\_not\_found\_failure"));

}

**return** order;

}

@Override

**public** **boolean** checkSalesRepId(String userId) **throws** Exception{

Connection connection = **null**;

**boolean** checkSalesRepIdFlag = **false**;

**try** {

*exceptionProps* = PropertiesLoader.*loadProperties*(***EXCEPTION\_PROPERTIES\_FILE***);

*goProps* = PropertiesLoader.*loadProperties*(***GO\_PROPERTIES\_FILE***);

DbConnection.*getInstance*();

connection = DbConnection.*getConnection*();

PreparedStatement statement = connection.prepareStatement(QuerryMapper.***IS\_SALES\_REP\_ID\_PRESENT***);

statement.setString(1, userId);

ResultSet resultSet = statement.executeQuery();

resultSet.next();

String userID= resultSet.getString("1");

**if**(userID!=**null**) {

checkSalesRepIdFlag=**true**;

**return** checkSalesRepIdFlag;

}

}**catch**(Exception e) {

GoLog.***logger***.error(*exceptionProps*.getProperty("userId\_not\_found\_failure"));

}

**return** checkSalesRepIdFlag;

}

@Override

**public** Product getProductDetails(String orderId, String productId) **throws** Exception {

Connection connection = **null**;

Product product = **null**;

**try** {

*exceptionProps* = PropertiesLoader.*loadProperties*(***EXCEPTION\_PROPERTIES\_FILE***);

*goProps* = PropertiesLoader.*loadProperties*(***GO\_PROPERTIES\_FILE***);

DbConnection.*getInstance*();

connection = DbConnection.*getConnection*();

PreparedStatement statement = connection.prepareStatement(QuerryMapper.***IS\_PRODUCT\_PRESENT***);

statement.setString(1, productId);

ResultSet resultSet = statement.executeQuery();

product = resultSet.getObject(1, Product.**class**);

} **catch** (DatabaseException e) {

GoLog.***logger***.error(*exceptionProps*.getProperty("productId\_not\_found\_failure"));

}

**return** product;

}

@Override

**public** **boolean** checkDispatchStatusForCancelling(String orderId) {

**boolean** checkDispatchStatusFlag = **false**;

Connection connection = **null**;

PreparedStatement statement;

ResultSet resultSet;

**int** index = 0;

**try** {

*exceptionProps* = PropertiesLoader.*loadProperties*(***EXCEPTION\_PROPERTIES\_FILE***);

*goProps* = PropertiesLoader.*loadProperties*(***GO\_PROPERTIES\_FILE***);

connection = DbConnection.*getInstance*().*getConnection*();

statement = connection.prepareStatement(QuerryMapper.***CHECK\_ORDER\_DISPATCH\_STATUS***);

statement.setString(1, orderId);

resultSet = statement.executeQuery();

**while** (resultSet.next()) {

index = resultSet.getInt(1);

}

**if** (index == 1) {

checkDispatchStatusFlag = **true**;

}

} **catch** (DatabaseException | IOException | SQLException se) {

// **TODO** Add Logger

se.printStackTrace();

}

**return** checkDispatchStatusFlag;

}

@Override

**public** List<OrderProductMap> getOrderProductMapForCancelling(String orderId) {

OrderProductMap opm = **null**;

Connection connection = **null**;

List<OrderProductMap> list = **null**;

list = **new** ArrayList<OrderProductMap>();

**try** {

*exceptionProps* = PropertiesLoader.*loadProperties*(***EXCEPTION\_PROPERTIES\_FILE***);

*goProps* = PropertiesLoader.*loadProperties*(***GO\_PROPERTIES\_FILE***);

connection = DbConnection.*getInstance*().*getConnection*();

PreparedStatement statement = connection.prepareStatement(QuerryMapper.***GET\_PRODUCT\_MAP***);

statement.setString(1, orderId);

ResultSet resultSet = statement.executeQuery();

**while** (resultSet.next()) {

**int** productStatus = resultSet.getInt("PRODUCT\_STATUS");

**if** (productStatus != 1) {

// log4j.logger.error(exceptionProps.getProperty("order\_return\_failure"));

**throw** **new** SalesRepresentativeException(*exceptionProps*.getProperty("product\_cancel\_failure"));

} **else** {

String productId = resultSet.getString("PRODUCT\_ID");

String productUIN = resultSet.getString("PRODUCT\_UIN");

opm = **new** OrderProductMap(orderId, productId, productUIN, (productStatus == 0 ? **false** : **true**),

**false**);

list.add(opm);

}

}

} **catch** (SalesRepresentativeException | DatabaseException | SQLException | IOException e) {

GoLog.***logger***.error(*exceptionProps*.getProperty("orderId\_not\_found\_failure"));

}

**return** list;

}

@Override

**public** String cancelOrder(OrderCancel oc) {

Connection connection = **null**;

String cancelOrderStatus = "Order cant be cancelled";

**int** i = 0;

**try** {

*exceptionProps* = PropertiesLoader.*loadProperties*(***EXCEPTION\_PROPERTIES\_FILE***);

*goProps* = PropertiesLoader.*loadProperties*(***GO\_PROPERTIES\_FILE***);

connection = DbConnection.*getInstance*().*getConnection*();

PreparedStatement statement = connection.prepareStatement(QuerryMapper.***ADD\_CANCEL\_ORDER***);

statement.setString(1, oc.getOrderId());

statement.setString(2, oc.getUserId());

statement.setString(3, oc.getProductId());

statement.setString(4, oc.getProductUIN());

java.util.Date utilDate = oc.getOrderCancelTime();

java.sql.Date sqlDate = **new** java.sql.Date(utilDate.getTime());

statement.setDate(5, sqlDate);

statement.setInt(6, 1);

i = statement.executeUpdate();

System.***out***.println("The order-cancel table's " + i + " columns has been updated");

PreparedStatement statement2 = connection.prepareStatement(QuerryMapper.***UPDATE\_ORDER\_PRODUCT\_MAP***);

statement2.setString(1, oc.getOrderId());

**int** j = statement2.executeUpdate();

System.***out***.println("The order-product-map table's " + j + " columns has been updated");

cancelOrderStatus = "The product with the uin" + oc.getProductUIN() + "has been cancelled";

} **catch** (SQLException | IOException | DatabaseException e) {

GoLog.***logger***.error(*exceptionProps*.getProperty(" return\_order\_failure"));

}

**return** cancelOrderStatus;

}

@Override

**public** **int** getProductQuantityOrdered(String orderId, String productId) **throws** Exception {

**int** productQuantity = 0;

Connection connection = **null**;

**try** {

*exceptionProps* = PropertiesLoader.*loadProperties*(***EXCEPTION\_PROPERTIES\_FILE***);

*goProps* = PropertiesLoader.*loadProperties*(***GO\_PROPERTIES\_FILE***);

connection = DbConnection.*getInstance*().*getConnection*();

PreparedStatement statement = connection.prepareStatement(QuerryMapper.***GET\_PRODUCT\_QUANTITY***);

statement.setString(1, orderId);

statement.setString(2, productId);

ResultSet resultSet = statement.executeQuery();

resultSet.next();

productQuantity = resultSet.getInt(1);

} **catch** (SQLException e) {

GoLog.***logger***.error(*exceptionProps*.getProperty(" product\_quantity\_failure"));

}

**return** productQuantity;

}

@Override

**public** String cancelProduct(String orderId, String productId, **int** productQty, **int** qty) **throws** Exception {

String cancelProductStatus = "Product cant be cancelled";

Connection connection = **null**;

**int** rowsChanged = 0;

**try** {

*exceptionProps* = PropertiesLoader.*loadProperties*(***EXCEPTION\_PROPERTIES\_FILE***);

*goProps* = PropertiesLoader.*loadProperties*(***GO\_PROPERTIES\_FILE***);

connection = DbConnection.*getInstance*().*getConnection*();

**if** (productQty == qty) {

PreparedStatement statement = connection

.prepareStatement(QuerryMapper.***UPDATE\_ORDER\_PRODUCT\_MAP\_CANCEL\_PROD\_EQUAL\_QUANTITY***);

statement.setString(1, orderId);

statement.setString(2, productId);

rowsChanged = statement.executeUpdate();

} **else** **if** (productQty > qty) {

PreparedStatement statement = connection

.prepareStatement(QuerryMapper.***UPDATE\_ORDER\_PRODUCT\_MAP\_CANCEL\_PROD\_LESS\_QUANTITY***);

statement.setString(1, orderId);

statement.setString(2, productId);

statement.setInt(3, qty);

rowsChanged = statement.executeUpdate();

}

Integer index = rowsChanged;

String rows = Integer.*toString*(index);

cancelProductStatus = "The given quantity of product has been cancelled and" + rows

+ "rows has been changed";

System.***out***.println(cancelProductStatus);

**return** cancelProductStatus;

} **catch** (SQLException e) {

GoLog.***logger***.error(*exceptionProps*.getProperty(" product\_quantity\_failure"));

}

**return** cancelProductStatus;

}

Order getOrderDetails(String orderId) **throws** Exception;

**boolean** checkSalesRepId(String userId) **throws** Exception;

Product getProductDetails(String orderId, String productId) **throws** Exception;

List<OrderProductMap> getOrderProductMapForCancelling(String orderId);

**boolean** checkDispatchStatusForCancelling(String orderId);

String cancelOrder(OrderCancel oc);

**int** getProductQuantityOrdered(String orderId, String productId) **throws** Exception;

String cancelProduct(String orderId, String productId, **int** productQty, **int** qty) **throws** Exception;

product\_quantity\_failure=Product is not present in the order