/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* - Function Name : cancelOrder - Input Parameters : String orderId - Return

\* Type : - Throws :SalesRepresentativeException - Author : CAPGEMINI - Creation

\* Date : 23/09/2019 - Description : Cancel Order to database calls dao method

\* getOrderDetails(sr)

\*

\* **@throws** Exception

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**public** String cancelOrder(String orderId, String userId) **throws** Exception {

Order order = **null**;

OrderProductMap opm = **null**;

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

String statusOrderCancel = **null**;

System.***out***.println("Cancelling of order is being processed");

**if** (salesRep.getOrderDetails(orderId) == **null**) {

**throw** **new** OrderNotFoundException("No such order id exists");

} **else** **if**(salesRep.checkSalesRepId(userId)) {

**throw** **new** SalesRepresentativeException("Sales Representaive id is invalid");

}

**else** **if** ((salesRep.checkDispatchStatusForCancelling(orderId)) == **true**) {

**throw** **new** OrderNotFoundException("Order cant be cancelled as it is dispatched! Return the order");

} **else** **if** (salesRep.getOrderProductMapForCancelling(orderId).isEmpty() == **true**) {

**throw** **new** OrderNotFoundException("Order cant be mapped with products");

} **else** {

list = salesRep.getOrderProductMapForCancelling(orderId);

Iterator<OrderProductMap> itr = list.iterator();

Date dt = **new** Date();

**int** i = 0;

**while** (itr.hasNext()) {

OrderCancel oc = **new** OrderCancel(orderId, userId, list.get(i).getProductId(),

list.get(i).getProductUIN(), dt, 0);

String statusOrderReturn = salesRep.cancelOrder(oc);

i++;

itr.next();

}

statusOrderCancel = "Order has been cancelled";

}

**return** statusOrderCancel;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* - Function Name : cancelProduct - Input Parameters : String orderId, String

\* productID, int qty - Return Type : - Throws :SalesRepresentativeException -

\* Author : CAPGEMINI - Creation Date : 23/09/2019 - Description : Cancel Order

\* to database calls dao method getOrderDetails(sr)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

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

Order order = **null**;

OrderProductMap opm = **null**;

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

List<Order> listOrder = **null**;

String statusProductCancel = **null**;

**if** (salesRep.getOrderDetails(orderId) == **null**) {

**throw** **new** OrderNotFoundException("No such order id exists");

} **else** **if**(salesRep.checkSalesRepId(userId)) {

**throw** **new** SalesRepresentativeException("Sales Representaive id is invalid");

}

**else** **if** ((salesRep.checkDispatchStatusForCancelling(orderId)) == **true**) {

**throw** **new** OrderNotFoundException(

"Selected Product cant be cancelled as it is dispatched! Return the Product");

} **else** **if** (salesRep.getOrderProductMapForCancelling(orderId).isEmpty() == **true**) {

**throw** **new** OrderNotFoundException("Order cant be mapped with products");

} **else** {

list = salesRep.getOrderProductMapForCancelling(orderId);

Iterator<OrderProductMap> itr = list.iterator();

**int** totalQuantityOrdered = list.size();

Date dt = **new** Date();

**int** i = 0;

**int** productQtyOrdered = salesRep.getProductQuantityOrdered(orderId, productId);

**if** (productQtyOrdered == 0) {

System.***out***.println("Invalid product id");

} **else** {

**if** (productQtyOrdered < qty) {

statusProductCancel = "Quantity of product cancelled cant be greater than Quantity of product ordered";

**return** statusProductCancel;

} **else** **if** (productQtyOrdered == qty) {

// while (itr.hasNext()) {

statusProductCancel = salesRep.cancelProduct(orderId, productId, productQtyOrdered, qty);

// itr.next();

// }

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

// while (itr.hasNext()) {

statusProductCancel = salesRep.cancelProduct(orderId, productId, productQtyOrdered, qty);

// i++;

// itr.next();

// }

}

}

**return** statusProductCancel;

}

String cancelOrder(String orderId, String userId) **throws** Exception;

String cancelProduct(String orderId, String userId, String productId, **int** qty)

**throws** Exception;

SELECT COUNT(\*) FROM ORDER\_PRODUCT\_MAP WHERE ORDER\_ID = OR157 AND PRODUCT\_ID = PROD06;