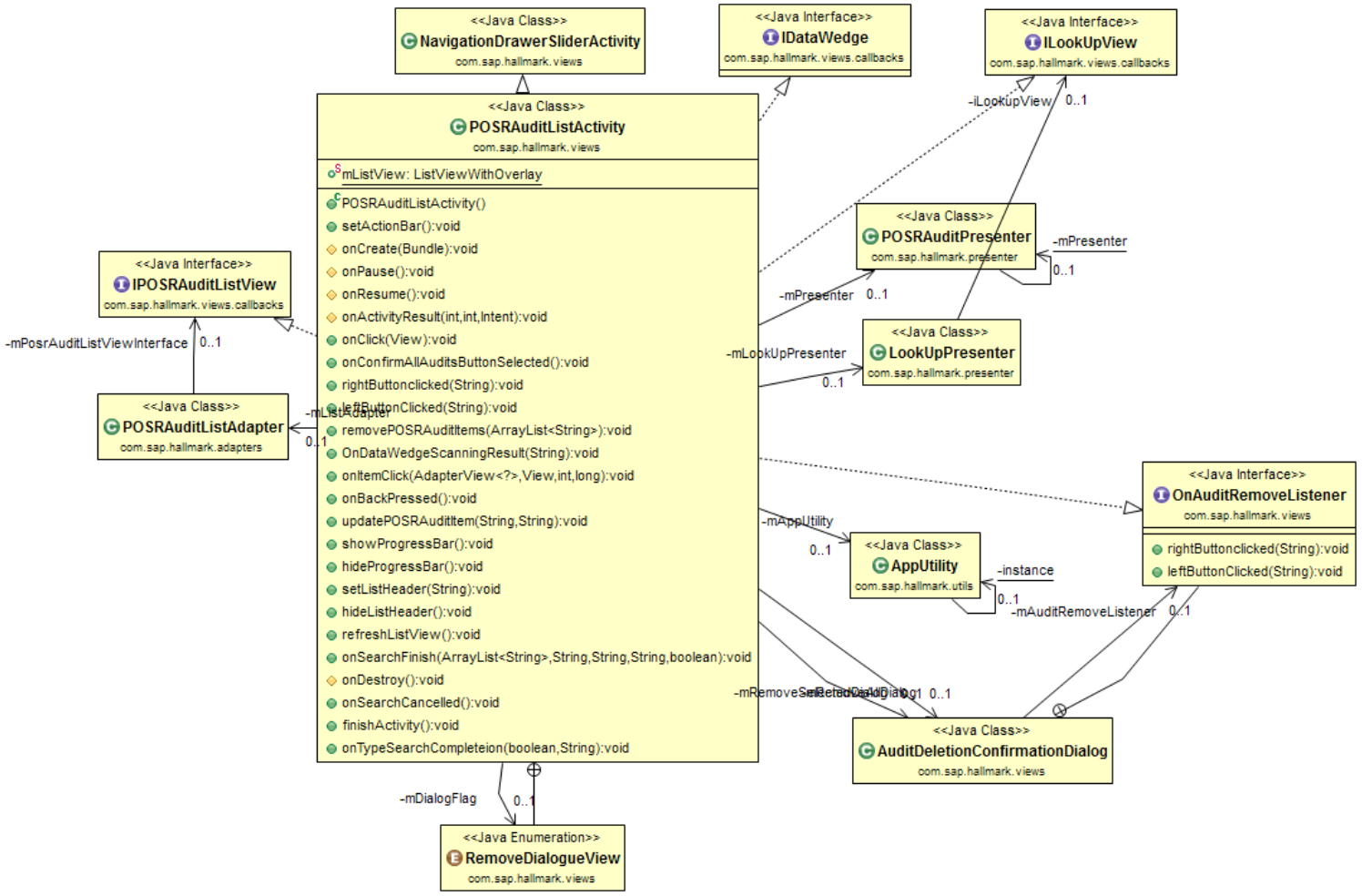
# POSRAudit

Priyanka S N
SEP 21, 2015

# Agenda

- General Program Structure
- Interfaces, Classes
- Packages
- POSR Audit
- Reorder

# General Program Structure



# Interfaces, Classes

| Object Type | Object | Short Text/Comment |
|---|---|---|
| Class | POSRAuditListActivity | Common Activity for POSR Audit; it displays the list of POSR products that have been audited for a particular customer, and allows users to edit the audit quantities |
| Class | POSRAuditListAdapter | Adapter for the POSR Audit list |
| Class | POSRAuditListABO | POSRAudit ABO class, it integrates with the MBOs The data can be saved locally in the device, and the class provides an interface between the MBO layer and the view layer for POSR Audit module. |
| Class | POSRAuditPresenter | Presenter class to connect the ABO layer to the POSR Audit UI |
| Class | ListViewWithOverlay | Listview with an overlay at the implemented position in {@link #dispatchDraw(Canvas)} |
| Interface | OnListItemsChangedListener | Callback when list items are selected |
| Interface | ILookupView | Callback to search for product by UPC/reorder ticket number |

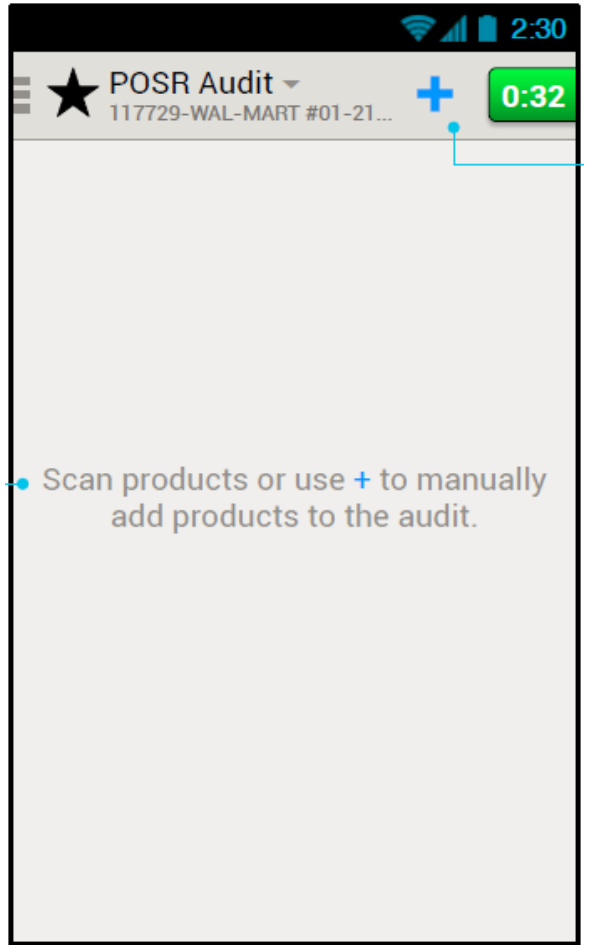
# Packages

The following packages are created to support this functionality:

com.sap.hallmark.presenter
com.sap.hallmark.views.callbacks
com.sap.hallmark.views
com.sap.hallmark.adapters

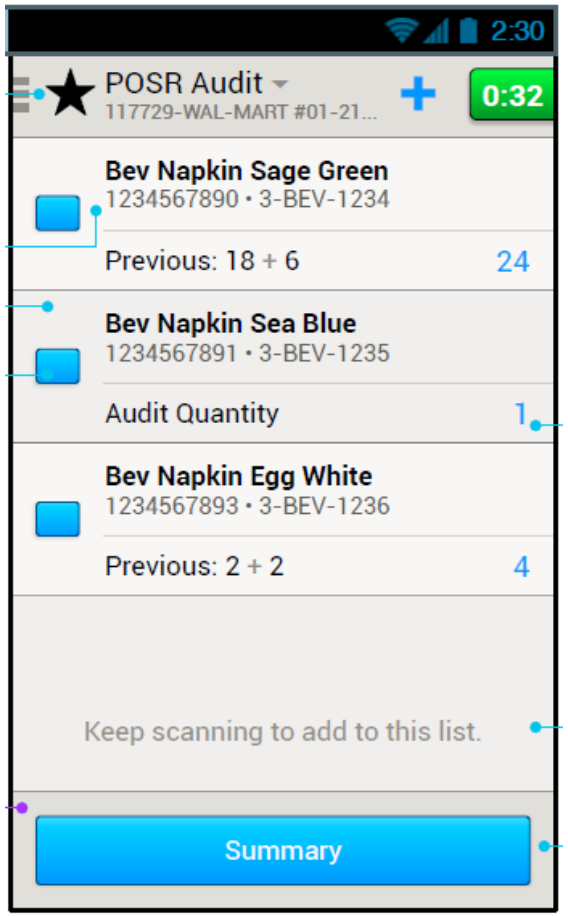
# POSRAudit

The initial POSR Audit screen instructs the user how to begin using the feature.



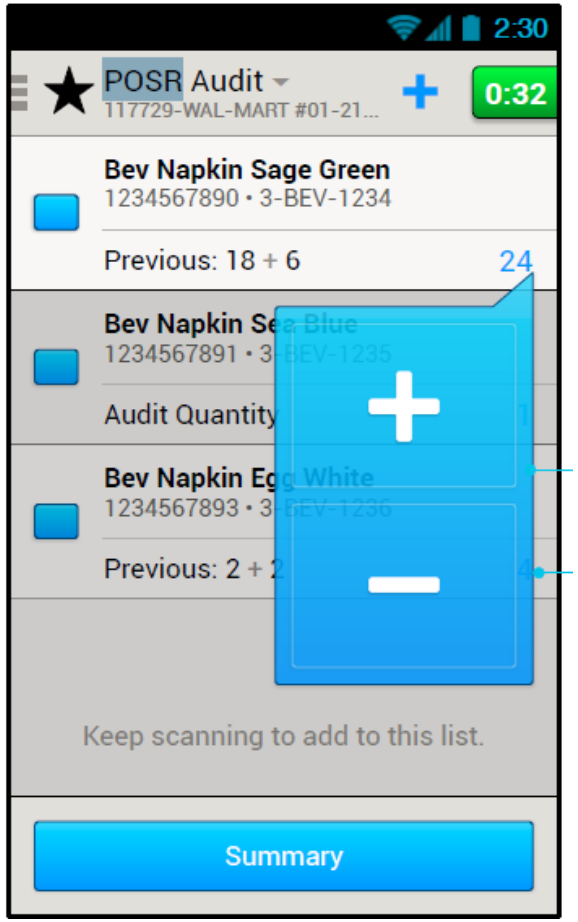
# POSRAudit

- The user is able to scan multiple items in a row. Individual products are listed, with multiple scans of the same product being collected together to increase the quantity.
- Quantity can be increased by either scanning an item multiple times or by manual change.
- If a previously scanned product is scanned again, it moves to the top and changes the "Audit Quantity“.



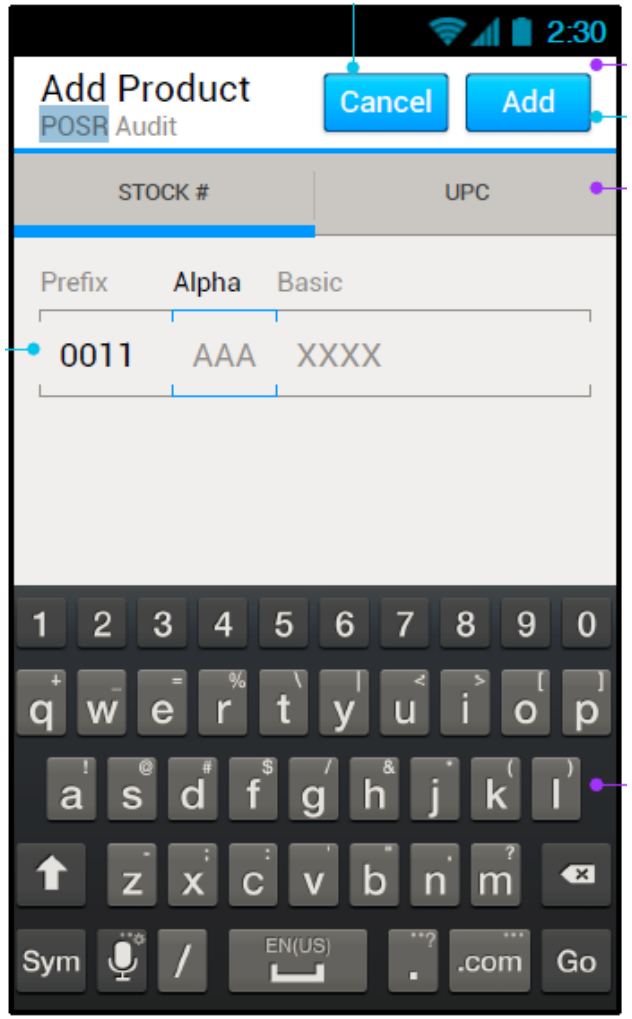
# POSRAudit

- When the user scans a product, an overlay appears allowing them to increase/decrease the quantity amount quickly.
- After ~15 seconds of inactivity the overlay automatically fades away. It can also be dismissed by  tapping any of the dulled out area.



# POSRAudit

Unlike the Lookup function, the user must enter a complete valid pocket ID or stock number to return a single result. They cannot perform searches for partial information.



## POSR Audit

```
private void addProductToPOSRAuditList(ArrayList<String> productIds) {
mInProgress = false;
ArrayList<ProductModel> products = mPresenter.getPOSRProductList(
productIds, mCustomerId);
for (ProductModel product : products) {
if (product != null) {

addPOSRProduct(product);
if (products.size() == 1) {
mListView.smoothScrollToPosition(0);
mListView.setShowOverlay(true);
mListAdapter.setShowListPopup(true);
}
}
}
}
```

## POSR Audit

```
public void addPosrAuditItem(
final HashMap<String, String> productAuditQuantityValues,
final String customerId, final String enterpriseId,
final String status,
final IHallmarkUncaughtExceptionHandler exceptionHandler) {

new Thread(new Runnable() {

@Override
public void run() {
// TODO Auto-generated method stub
try {
new HallmarkExceptionHandler().init(mContext, Thread.currentThread(), exceptionHandler);
int count = 0;
ProductABO productABO = new ProductABO();
ProductModel product = new ProductModel();
// To get the timestamp of the POSR Audit specific to a
// store
Timestamp timestamp = mAppUtility.getAuditTimestamp(
customerId, enterpriseId);
```

## POSR Audit

```
if (timestamp == null) {
// Create new POSR audit if order items is created for
// the
// first time
TimeStampUtility timeStampUtility = TimeStampUtility
.getTimeStampInstance(mContext);
timestamp = timeStampUtility
.converTimestampToUTC(new Timestamp(System
.currentTimeMillis()));
createPOSRAudit(customerId, enterpriseId, timestamp,
null, false);
String key = IAppConstants.POSRAuditScreenCons.POSR_AUDIT_TIMESTAMP_KEY
+ customerId
+ IAppConstants.HYPHEN
+ enterpriseId;
AppConfig appConfig = AppConfig
.getAppConfigInstance(mContext);
appConfig.saveLongPreference(key, timestamp.getTime());
}
```

## POSRAudit

```
if (timestamp != null) {
for (String productId : productAuditQuantityValues
.keySet()) {
++count;
POSRAuditItem auditLine = POSRAuditItem
.findByPrimaryKey(customerId, enterpriseId,
productId, timestamp.toString());
if (auditLine != null) {
auditLine
.setAudit_Qty(productAuditQuantityValues
.get(productId));
auditLine.setStatus(status);
auditLine.update();
}
```

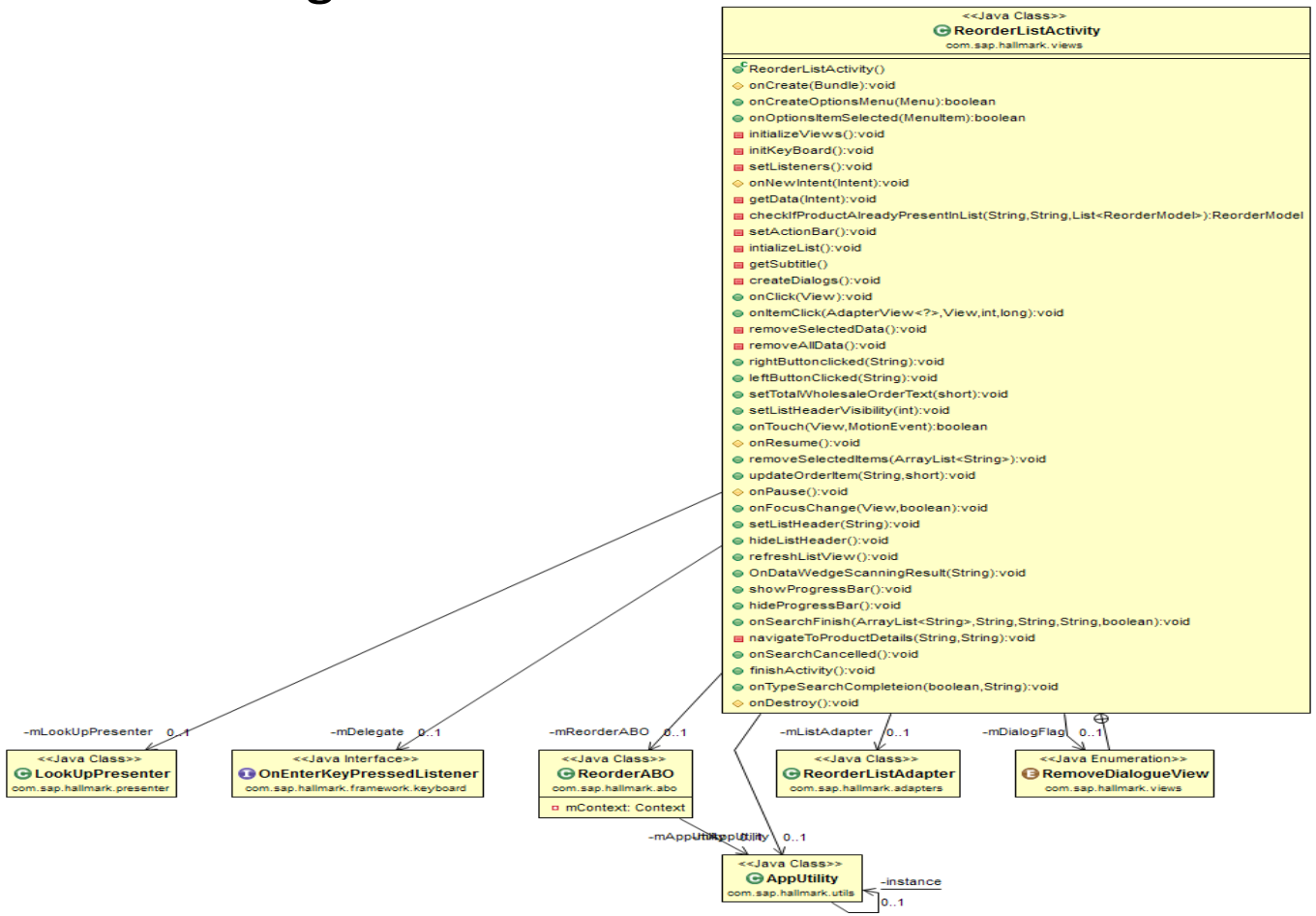
## POSRAudit

```
else {
product = productABO.getProductDetailsById(
productId, customerId);
savePOSRAuditItem(product, customerId,
enterpriseId,
productAuditQuantityValues
.get(productId), status, count,
timestamp);
}
}
}
} catch (Exception ex) {
AppLogs.e(IAppConstants.LoggingConstants.LOG_TYPE_ERROR,
ex.getMessage());
}

}

}).start();
}
```

# General Program Structure



## Packages

The following packages are created to support this functionality:
com.sap.hallmark.views
com.sap.hallmark.adapters
com.sap.hallmark.views.callbacks

# Reorder

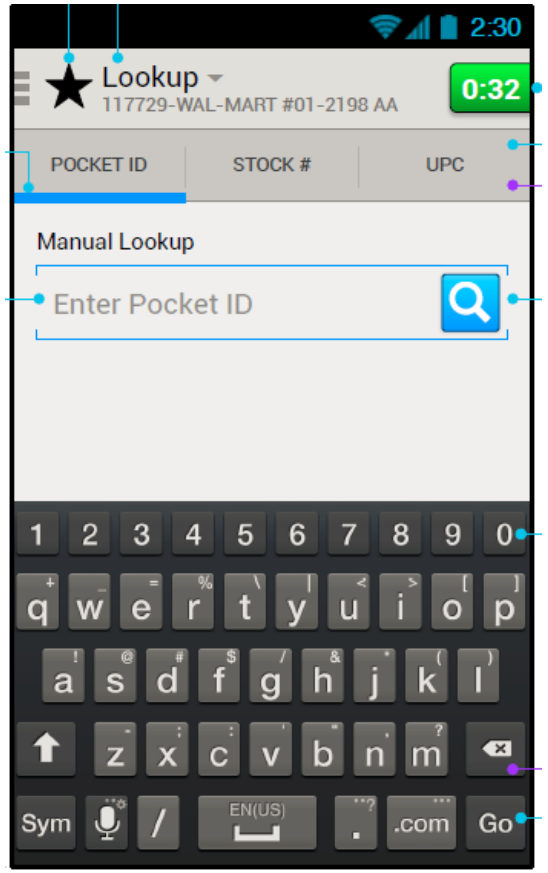
## Interfaces, Classes

| Object Type | Object | Short Text/Comment |
|---|---|---|
| Class | ReorderListActivity | Common Activity for Orders; it displays the list of products that have been ordered for a particular customer; and also allows users to edit the wholesale quantities |
| Class | ReorderListAdapter | Adapter for the orders list |
| Class | ReorderABO | Reorder ABO class that integrates with the MBOs. The data can be saved locally in the device. The class provides an interface between the MBO layer and the view layer for Reorder module. |
| Interface | OnListItemsChangedListener | Callback when list items are selected |
| Interface | ILookupView | Callback to search for product by UPC/reorder ticket number |

# Reorder

The user can scan reorder ticket number from anywhere in the Lookup Module, and the corresponding product is added to the list.

It is possible to edit wholesale quantities in the Order list



# Reorder

**ReorderListActivity:**

Using this activity:
It is possible to display the list of products that have been ordered for a particular customer, and also allows users to edit the wholesale quantities.
A search is triggered for the UPC and if search is successful then the user is navigated to Product details of the UPC.

## ReorderABO:

ABO creates the Order header for the customer only if there are OrderItems corresponding to header.

ABO updates or deletes the OrderItems.

# Thank You