Table of contents:

Android Environment Setup

Android Build Tools: 26.0.2
Targeted SDK: 26
Java Version: 7
Minimum Device Support: API 22

Command Line Interface To Test And Run

Building

Use the Build Variants window in Android Studio to choose which version of the app you want to install, or alternatively choose one of the following tasks from the command line:

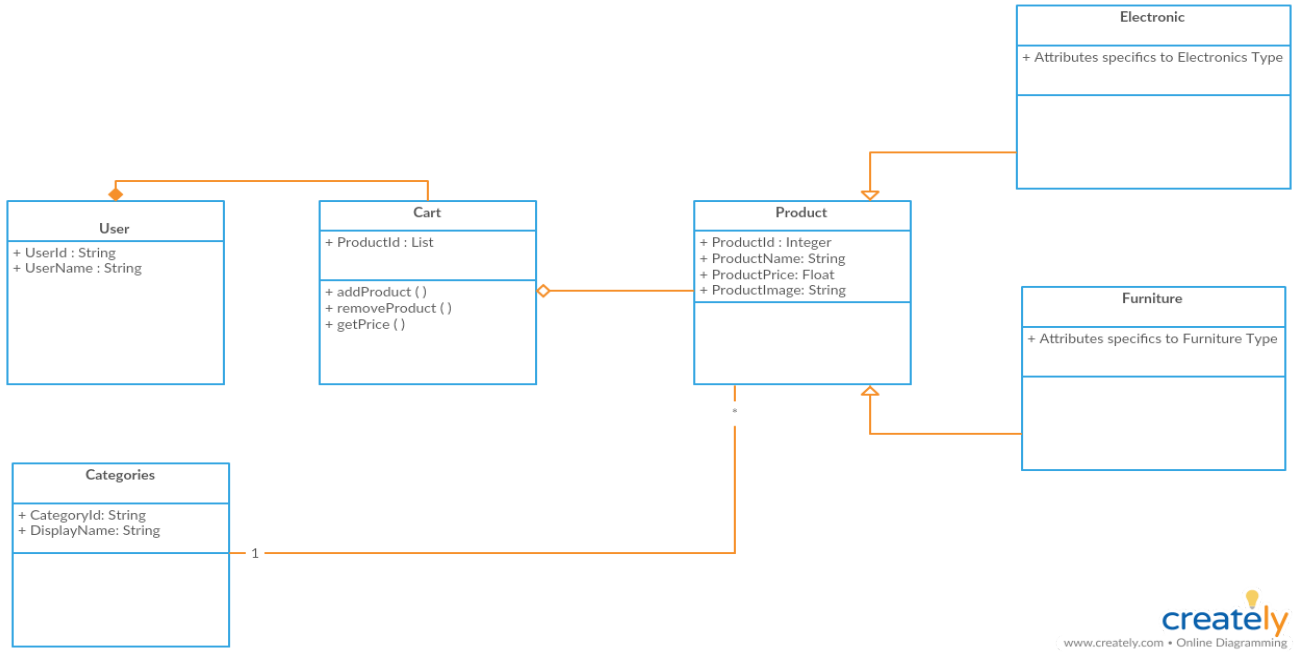
Unit Test

./gradlew testProdReleaseUnitTest

Run (Note: Ensure emulator is up and running)

./gradlew installProdDebug

Object Oriented Design



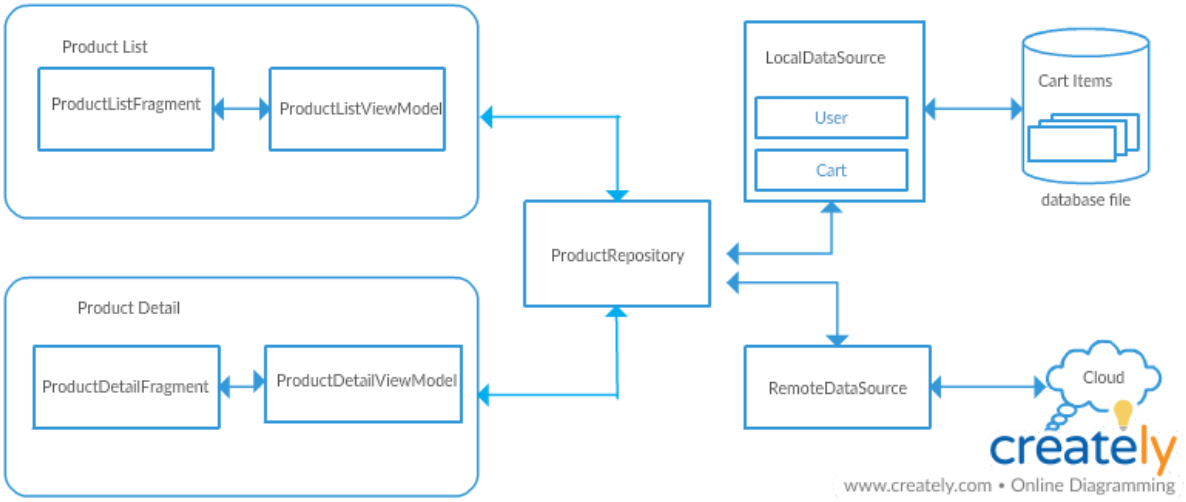
Relationship:

1. Electronic and Furniture product types inherit from Product abstract class.
2. User and Cart shares composition relationship as Cart can't outlive the User's lifetime and as well User owns the Cart.
3. Cart and Product shares aggregation relationship as Product can outlive the Cart but they can't own the Product
4. Category and Product share associate relationship as each won't own either way and each can outlive others.

Reasoning:

1. Having all the behavior of the business that is likely to change encapsulated in a single part of our software decreases the amount of time we need to perform a change because it will all be performed in one place.

Application Architecture Diagram



Reasoning:

1. By isolating that code as much as possible, we decrease the likelihood of changes in other places causing it to break, thus decreasing the time it takes to stabilize the system.

External Libraries

- Butterknife - reduces dev time, code readability

Naming Convention

- File name: <domain_name>_<usage>_<andriod_component_name>
- Variable name: <domain_name>_<framework_or_component_name>

What Next?

- Unit testing domain entity
- Automated UI tests
- Unit testing Local Database
- Clarification on Error screen behaviour

Code Snippets

1. Passing Event Triggered from View to ViewModel:
2. Passing Event Triggered from background Operation to View:
3. Passing Event from Fragment to Activity or Vice Versa:

are performed view ObservableList or MutableLiveData which are ***observable*** pattern.

ViewModel.java

```
public final ObservableList<Product> items = new ObservableArrayList<>();
….
….
public void getProducts(new ProductsDatasource.LoadProductsCallback() {
     @Override public void onProductsLoaded(List<Product> products) {
      items.addAll(products);
 }
```

Fragment.java

```
ViewModel.getItems().observe(this, events -> {
        Log.d(TAG, "Events Changed:" + events);
        //”events” - hold all data, if populated
    });
```