**CS-499 Milestone Three  
John Kirven**

**About this artifact:**This artifact is an Android application built with Java whose primary purpose is to manage the inventory/stock of a warehouse. It supports simple (local) account creation and uses a local database to hold item information and user accounts. The app uses a model-view-viewmodel (MVVM) architectural pattern to separate the UI logic from the database data. The database is handled via Room, a library for implementing SQLite in Android and currently houses two tables, Account and InventoryItem. The app itself consists of three Activities; login/account creation, adding new items, and browsing the list of all items.

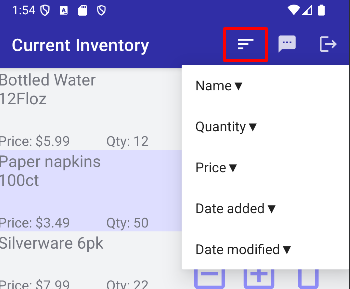
**About the enhancement:**The original implementation of the InventoryItem data structure was functional, but simple and lacking fields that it probably should’ve had such as a price and SKU. Additionally, the List that the database table populated was in an unspecified order and had no methods to sort the data by any meaningful metric. My plan was to remedy both by enhancing the data structure and implementing sorting algorithms so the user can choose a metric and direction to sort the List. Additionally, I wanted to clean up the existing code base and properly comment the code, two things I failed to do in the original due to time constraints.

**The process:**This process was not nearly as smooth as the process for the first artifact, I am nowhere near as familiar with Android development as I am with Java as a whole, so I had to consult the Android Developers website throughout the process. With my enhancements planned, my first step was to add the additional fields into the InventoryItem data structure along with getters and setters to be used by the view model and then ensure that the database schema matched the updated InventoryItem class which I accomplished by wiping the existing database from my emulated device. While still working in InventoryItem I also updated the method that adds dummy items into the database to include the new fields. From there, I had to update the layout for the new item activity so users can populate those new fields when adding a new item, along with data validation for those new fields in the AddInventoryActivity class.

A screenshot of a computer

AI-generated content may be incorrect.

With the data structure properly expanded, it was time to add sorting. I already had a menu above the List so I added a new item to that menu for sorting. I then created a PopupMenu that is generated when the user presses the sort button (highlighted in red), the items in the popup menu make calls to various sorting algorithms that sort and refresh the item list. The arrows indicate if the sort metric is currently ascending or descending.



As all the fields in the InventoryItem class use data types that already have a compareTo method, sorting was done by utilizing the Comparator class as well as the Collections class to reverse the order of the list when descending is selected.

**Outcomes:**

Design and evaluate computing solutions that solve a given problem using algorithmic principles and computer science practices and standards appropriate to its solution, while managing the trade-offs involved in design choices (data structures and algorithms)

Demonstrate an ability to use well-founded and innovative techniques, skills, and tools in computing practices for the purpose of implementing computer solutions that deliver value and accomplish industry-specific goals (software engineering/design/database)

Develop a security mindset that anticipates adversarial exploits in software architecture and designs to expose potential vulnerabilities, mitigate design flaws, and ensure privacy and enhanced security of data and resources