**Homework 3 (75 points)**

CSCI 1125-30: Object-Oriented Programming Using Java

Spring 2017

**Instructor**: Donald Sawyer

**Student**: [Your Name Here]

**What to turn in:**

1. Copy and paste your source code into a word document and paste screen shot(s) of your program output window.
   1. If you import special java libraries into your program, you MUST include which libraries, their version, and where you got them from.
2. Output file created (transaction\_summary.txt) after using retail\_store.txt as input.
3. All source code files (only .java files).
4. Compress all files into a SINGLE COMPRESSED ZIP file with the name: yournamehw3.zip
5. Place the compressed zip file into the D2L dropbox for Homework 3.
6. BONUS POINTS: You can earn up to 2.5 bonus points per method tested on RetailTransaction using JUnit. Fully tested functions will receive 2.5 points (will need multiple test cases for full 2.5 points). The methods that can earn points are:
   1. scanItem()
   2. calculateTransactionTotal()
   3. getNumberOfProductsSold()

*NOTE: If I am unable to compile or run your program because you are missing some source files, you will receive a grade of zero (0) for the assignment.*

*NOTE 2: I will be running your program with other input files, not just the one included, so make sure your program works.*

# Objectives

The learning objectives for this assignment are as follows:

1. Redesign a legacy Java program
2. Using the ArrayList class
3. Use inheritance for similar class structures
4. Use StringBuilder for building long messages
5. String operations

# Assignment Description: Refocusing the Business Model

The CEO for the Retail Store has decided that they want to get away from entertainment and focus on hand-held electronics. CEO has decided that they will no longer be selling Laptops, TVs and HDMI cables. As part of the new strategy, the business will focus largely on carrying many different cell phones, laptops, and accessories. During this transition, the CEO will focus on cell phone varieties. Customers have asked for receipts to be generated so they can see the items and totals for their transactions. The retail store software will need to be able to generate receipts with all required information.

You will be modifying the existing Retail Store application. In this assignment, you will use inheritance to be able to read a product catalog from a file with multiple types of cell phone products and accessories.

## Inputs

There will be one text file used as input to the program:

1. A text file called retail\_store.txt

The input file will contain multiple runs of transactions and sales in the application. See Appendix B for the format of the input file.

### Product Database

Your program will need to read the product catalog from the text file and load the products into a product catalog class. There will be two types of products that can be stored in the product catalog:

1. Cell phone
2. Headphones

### Retail Transactions

Your program will need to read retail transactions from the input file and process them. The new fields added to a transaction are listed below:

1. Transaction date/time in the format yyyy-mm-dd HH:MM:SS (example: 2017-11-20 17:53:01).

## Outputs

There will be two types of output produced as files:

1. The summary of each transaction (as in previous homework) will be written to transaction\_summary.txt.
2. A receipt for EACH transaction will need to be created

# Assignment Requirements

Below are the business requirements for the application for this second phase. All the old requirements are still valid, unless they are modified in the requirements below.

## New Application Requirements

1. The application should no longer handle Television, HDMI Cable, and Laptop Computers.
2. Each product will have a product identifier associated to it that will be a unique string for that product. The identifier is used to look up the item.
   1. Product identifiers will be alphanumeric values.
   2. All whitespace must be removed from the product id before storing it in the catalog.
3. The application must read the product catalog from the input file and store all products in a single ArrayList (leveraging inheritance).
   1. A Product class will be required.
   2. The CellPhone class must inherit from the Product class.
   3. The Headphones class must inherit from the Product class.
   4. Invalid products should be ignored and not added to the product catalog.
4. Methods required for ProductCatalog:
   1. Add a product
   2. Find a product by id
5. New class: LineItem
   1. A line item is a single line for a transaction.
   2. LineItem is used to track the selling of products within a RetailTransaction. It should have the following instance fields:
      1. The actual product sold (the instance of the Product sold)
      2. Quantity of the Product sold
      3. The number of warranties sold for the Product sold
   3. If the same product has multiple lines in the input file, a new LineItem should NOT be created. The existing quantity and warranty count should just be incremented.
6. Retail transactions must track instances of products purchased using a single ArrayList of LineItems.
7. Transaction type must be an enum instead of a String.
8. Methods required for RetailTransaction:
   1. Scan an item (this adds the item information to the RetailTransaction list of LineItems)
   2. Calculate total should now determine the transaction total by iterating the line items with the same calculations (including warranty calculations). It should also determine the tax and add it to the total.
   3. Number of products sold should iterate the line items and determine total quantity
   4. Generate a receipt
      1. A receipt will be a text file output with the name format of receipt\_<transactionId>\_<registerId>.txt. An example name would be receipt\_123456\_100.txt.
      2. Receipt format is listed in Appendix C.
      3. Receipt text must be built using a StringBuilder before being output to file.

*NOTE: If you don’t use an ArrayList where stated as required, you can use other data structure collection types that haven’t been covered in class yet. If you do this, make sure that the implementations can satisfy the functionality described by the requirements. Choosing a different data structure MUST provide an improvement to the design.*

# Appendix A: Updated UML Design

The UML below describes the general guidance for classes in your program. They contain the methods and relationships that should be followed. If you choose somewhat different names for methods or fields, that is okay, but the data structures themselves should match the format (unless you decide to improve on it).

## ProductCatalog & Product Classes:



## RetailTransaction and Related Data Structures:



# Appendix B: Input File Format

The input file will have be formatted with the following sections:

1. Products in the Product Catalog
2. Transaction 1
3. Transaction 2  
   …
4. Transaction n

*NOTE: Changes for this assignment are highlighted in yellow.*

## Product Catalog Section

The product catalog will start on the first line of the file. The products will be listed on their own line of the file, in no particular order. The attributes for each type of product are separated by commas.

### Cell Phone Product Line Format

cell phone,[product id],[price],[name],[warranty eligible], [brand],[carrier]

### Headphones Product Line Format

headphones,[product id],[price],[name],[warranty eligible],[brand],[Bluetooth capable]

## Retail Transaction Sections

Each retail transaction will start with the transaction detail line and end with the text “End Transaction”. The items sold in the transaction will appear between the start and end transaction lines.

### Retail Transaction Header

Transaction,[transaction id],[register id],[transaction date/time],[sale/return]

### Item(s) Sold Line

[product id],[quantity],[number of warranties]

## Example Input File (Product Catalog + 2 sales + 1 return)

cell phone,cp-111111,199.99,iPhone 4S,true,Apple,Verizon

headphones,h-111111,249.99,Bose QC25,true,Bose,false

headphones,h-111112 ,299.99,Bose QC35,true,Bose,true

cell phone,111111,99.99,Basic Android Phone,false,Google,Verizon

cell phone, 111112,99.99,Basic Android Phone,false,Google,AT&T

headphones,a-111111,29.99,Apple Buds,false,Apple,false

headphones,a-111111b,39.99,Apple Wireless Buds,false,Apple,true

headphones,h-11111,29.95,Skull Candy Buds,false,Skull Candy,false

Transaction,123456,100,2017-01-01 08:01:59,sale

cp-111111,1,1

a-111111b,1,0

a-111111,1,0

h-111112,1,1

End Transaction

Transaction,2468,15,2017-01-01 08:01:59,return

a-111111b,1,0

End Transaction

Transaction,2468,100,2017-01-03 17:11:11,sale

111112,2,1

111112,1,1

End Transaction

# Appendix C: Output Files

The sections below show examples of output files.

## Example Summary Output of Input File (transaction\_summary.txt)

Transaction 123456: 4 items totaling $664.91 on register 100

Transaction 2468: 1 item totaling ($42.89) on register 15

Transaction 2468: 3 items totaling $343.17 on register 100

Total Sales: $965.19

## Example Receipts Output

For the example in 5.3, there will be three receipts generated.

The format of a receipt is shown below.

* Text between square brackets [] indicates a value that needs to be inserted.
* 🡪 indicates a tab.
* All other text is the literal text to be inserted.

### Receipt File Format

**File Name for Each Receipt:**

receipt\_[transaction id]\_[register id].txt

**File Contents for Each Receipt:**

[transaction type] Transaction: [transaction id]

Register: [register id]

Transaction Date: [transaction date]

----------------------------------------

[product1 name]

🡪quantity: [quantity of product1 sold]

🡪price: [product1 price]

🡪warranties: [number of warranties for product1]

🡪subtotal: [cost for product1, including warranties, no tax]

...

[productN name]

🡪quantity: [quantity of productN sold]

🡪price: [productN price]

🡪warranties: [number of warranties for productN]

🡪subtotal: [cost for productN, including warranties, no tax]

=======================================

Subtotal: [sum of subtotals of all items]

Tax: [tax amount for subtotal]

TOTAL: [total transaction cost]

### receipt\_123456\_100.txt

SALE Transaction: 123456

Register: 100

Transaction Date: 2017-01-01 08:01:59

----------------------------------------

iPhone 4S

quantity: 1

price: $199.99

warranties: 1

subtotal: $219.99

Apple Wireless Buds

quantity: 1

price: $39.99

warranties: 0

subtotal: $39.99

Apple Buds

quantity: 1

price: $29.99

warranties: 0

subtotal: $29.99

Bose QC35

quantity: 1

price: $299.99

warranties: 1

subtotal: $329.99

=======================================

Subtotal: $619.96

Tax: $44.95

TOTAL: $664.91

### receipt\_2468\_15.txt

RETURN Transaction: 2468

Register: 15

Transaction Date: 2017-01-01 08:01:59

----------------------------------------

Apple Wireless Buds

quantity: 1

price: $39.99

warranties: 0

subtotal: $39.99

=======================================

Subtotal: ($39.99)

Tax: ($2.90)

TOTAL: ($42.89)

### receipt\_2468\_100.txt

SALE Transaction: 2468

Register: 100

Transaction Date: 2017-01-03 17:11:11

----------------------------------------

Basic Android Phone

quantity: 3

price: $99.99

warranties: 2

subtotal: $319.97

=======================================

Subtotal: $319.97

Tax: $23.20

TOTAL: $343.17