Analysis

Assignment 3 tasks us to design and develop a Java application that emulates a shopping cart that will take a file input by the user and do operations on the cart depending on what operations the user specified. The program should take the file from args, parse each line as an operation, perform the operation on the cart, printing feedback about each operation along the way, then lastly output a cart receipt/summary then terminate. If a user enters an invalid operation in the file, our program should output an error message, skip the invalid operation, and continue performing operations.

There are 5 different types of operations the user can perform: insert, search, update, delete, print. Insert adds an item to the shopping cart. Search counts the total quantity of items in the user’s cart with the specified name. Update changes the quantity value of the first instance of an item with the specified name. Delete removes all items with the specified name from the user’s cart, returning the total quantity deleted. Print outputs a cart receipt/summary to the standard output screen.

Each item in the cart can be of 3 types (clothing, groceries, or electronics) and each type has its own special features. Clothing has no premium shipping option. Groceries can be perishable or nonperishable; premium shipping is required if perishable. Also, sales tax isn’t applied to groceries. Electronics can be fragile or not, premium shipping is required when fragile, and the electronics must be shipped to a valid state. Certain states don’t apply sales tax on electronics: TX, NM, VA, AZ, and AK.

The design phase needs to include a system-level use case diagram, a UML model, ADT descriptions for every class, a functional block diagram showing calling relationships, and the algorithm for the main driver.

The code itself should also follow the coding style guidelines specified on Canvas, have understandable (and commented) logic, and be efficiently modifiable with encapsulated functionality.

Design

**Use Case Diagram:**



**Main Algorithm Pseudocode:**

Get file path from args.

Initialize a new Shopping Cart object

Read file one line at a time. For each line:

Is line empty or just whitespace?

If so:

Do nothing.

If not:

Initialize an Operation object from the line in the file.

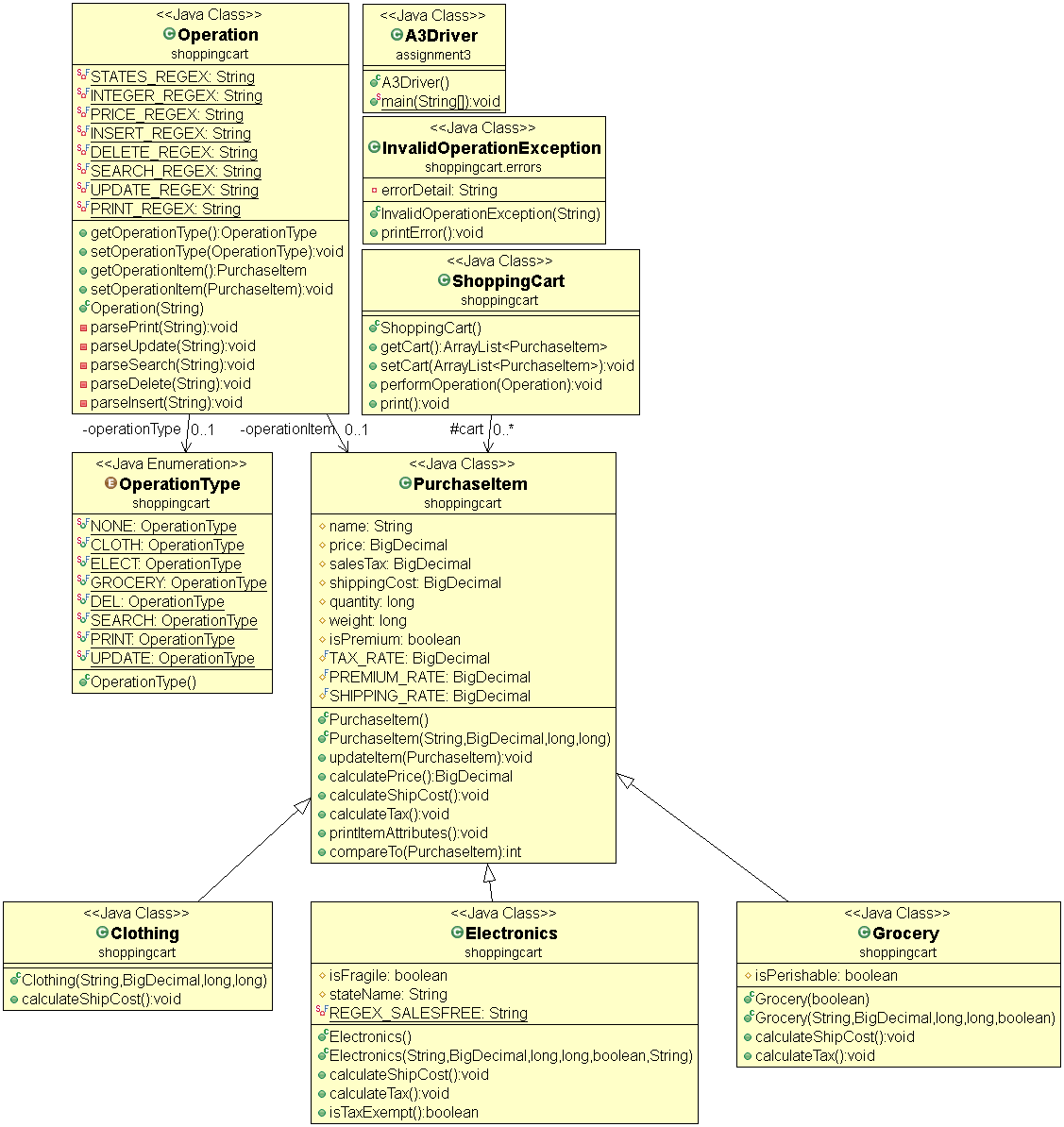
If the operation entered is invalid, display the error to the user.

Otherwise, Perform the operation on the shopping cart using performOperation(op)

Display the results of the operation to the user.

After reading all lines from the file, print a cart receipt/summary.

**UML Diagram:**



ADT level description:

The problem is to develop a program that parses an input file containing operations (insert, delete, search, update, and print) and process each operation accordingly (i.e. insert an item in the cart, delete all instances of an item, update quantity of item, search the cart for an item, or print a summary receipt).

* 1. PurchaseItem class
     1. Member Data:
* Name: String – Name of the item.
* Price: BigDecimal – Total price of specified quantity of an item.
* SalesTax: BigDecimal – Total tax of specified quantity of an item.
* ShippingCost: BigDecimal – Total shipping cost of specified quantity of an item.
* Quantity: long – quantity of an item.
* Weight: long – weight of an item.
  + 1. Member Methods:
* *UpdateItem():* update the quantity of an item.
* *BigDecimal calculatePrice():* calculate and return the total charges after tax of an item.
* *calculateShippingCost():* calculates total shipping cost.
* *CalculateTax():* calculates total taxes of an item.
  + 1. Electronics Class

(Inherits all member data from PurchaseItem class with some extra data).

* + - 1. Member Data:
* isFragile: Boolean – specifies fragility of item.
* stateName: String – specifies the state name from where the item is being purchased from
  + - 1. Member Methods:
* *Boolean isTaxExempt():* returns true if the item is being purchased from a tax exempt state.
  + 1. Gerocery Class

(Inherits all member data from PurchaseItem class with some extra data).

* + - 1. Member Data:
* isPerishable: Boolean – specifies perishability of item.
  + 1. Clothing Class

(Inherits all member data from PurchaseItem class).

* + 1. ShoppingCart Class
       1. Member Data:
* Cart: Arraylist<PurchaseItem> – Contains all items in the cart (ASCII alphabetized).
  + - 1. Member Methods:
* *performOperation():* processes each operation according to the operation type.
* *Print():* prints a summary of all the items in the carts with a total charge after all taxes and shipping.
  + 1. OperationType Enum

An enumerated type containing all the possible operation types (i.e. ELEC, CLOTH, DEL, SEARCH, PRINT, UPDATE.)

* + 1. Operation Class
       1. Member Data:
* operationType: OperationType – specifies the type of operation.
* operationItem: PurchaseItem – New item object used for processing the operation.
  + - 1. Member Methods:
* *parseSearch():* searches the item name in the cart and returns the total quantity of it.
* *parseUpdate():* updates the quantity of the first result in the shopping with the specified name.
* *parseDelete():* deletes every item with the specified name and returns the total quantity deleted.
* *parsePrint():* Calls shopping cart print to print a summary receipt of all the items in the cart.
* *parseInsert():* inserts an item into the cart.
  + 1. A3Driver (driver) Class
       1. Member Methods:
* *Main():* Opens, and reads the file each line at a time and processes each line input.

Functional Block Diagram:

