Project One Specification

Chad A. Baxter

September 11, 2017

0.1 Introduction

The project is designed as a database managment system for a simple store/shop warehouse. The software must be able to parse and edit values provided to it in a *C*omma *S*eparated *V*alue format. A self-intuitive command line interface must be provided.

0.2 Classes

0.2.1 Descriptions

Main The entry point of the program. Initializes important program-wide objects and gives control of standard input to *Interpreter*.

Interpreter Parses and handles user input. Insures user input values are valid and passes them to *Manipulator*.

Manipulator Takes data from *Interpreter* and runs the specified functions provided to it by all other helper classes.

PartList Handles all functions required to manipulate an *ArrayList* of *Part*s.

Part Stores all information in regards to a *Part*. Gets stored in *PartList*.

Importer Handles all data import including cli entries and file import parsing.

DatabaseHandler Handles all persistence operations.

0.2.2 Fields

An important feature of this design is that the only fields that need to exist are fields for *PartList* and *Part*. This reduces overhead and allows most other classes to remain static.

PartList	**Part**
list: *ArrayList¡Part¿*	partName: *String*
	partNumber: *Long*
	listPrice: *Double*
	salePrice: *Double*
	onSale: *Boolean*
	quantity: *Integer*

0.2.3 Methods

The list of methods contains names and parameters for each method as well as return types. These are subject to change as is the nature of programming larger projects.
*Key: * = all field names from a class descriptor.*

Main: **main**:*String[]* → *void*

Interpreter: **call**:*PartList* → *PartList*

Manipulator: **prompt**:*String* → *PartList*; **display**:*PartList* → *PartList*;
enter:*PartList* → *PartList*; **read**:*PartList* → *PartList*;
save:*PartList* → *PartList*; **read**:*PartList* → *PartList*;
save:*PartList* → *PartList*; **sell**:*PartList* → *PartList*;
sortByName:*PartList* → *PartList*; **sortByNumber**:*PartList* → *PartList*

PartList: **getList**:*void* → *ArrayList¡Part¿*

Part: **get***:*void* → **; **set***:*void* → **

Importer: **importLine**:*PartList, String* → *PartList*;
importFile:*PartList, String* → *PartList*

DatabaseHandler: **save**:*PartList* → *void*; **load**:*void* → *PartList*