COMP 2613 Assignment 1

The assignment will build on the labs and reinforce the concepts and features of the java framework we’ve learned in weeks one through five. You’ll be building the core of a motorcycle dealership (BC Motorcycle Centre – BCMC). As we haven’t covered Graphical User Interfaces, BCMC will be a commandline-based program, which means we’ll be focusing on good design and java programming techniques.

At the core of this application are your labs; the assignment is a consolidation and extension of the requirements for the labs.

## Requirements

The design of BCMC must follow good object-oriented principles and practices.

Your code must compile and run. Compile-time warnings are considered errors and must be eliminated from your code by using appropriate annotation tags.

The main class must be named Bcmc, and the jar file containing your runnable code must be named Bcmc.jar.

All activity must be logged to a text file named bcmc.log. Typical logged activities would be program startup and shutdown, and program flow. These activities will be logged as INFO messages. Any exception would be logged as ERROR messages.

Exceptions must be handled such that no stack traces are displayed in the console, but as mentioned above, a message will be logged explaining the cause of the error. Stack traces will be logged to the log file so that programmers maintaining your application can fix the errors by looking at the log information.

Instead of the data being passed as commandline arguments, it will be read from a data file in plaintext format. For the assignment, there will be three data files:

1. customers.dat
2. motorcycles.dat (in for service, not for sale)
3. inventory.dat

The format of the data in these files are:

1. ID|FIRST\_NAME|LAST\_NAME|STREET|CITY|POSTAL\_CODE|PHONE|EMAIL| JOIN\_DATE
2. ID|MAKE|MODEL|YEAR|SERIAL\_NUMBER|MILEAGE
3. MOTORCYCLE\_ID|DESCRIPTION|PART#|PRICE|QUANTITY

Note: the first line of the file describes the contents and is to be ignored when the file is read.

For each of these data files, you will need to have classes to represent the data:

1. Customer
2. Motorcycle
3. Inventory

The relationships between these classes are as follows:

* A customer can have zero or more motorcycles
* A motorcycle can have zero or more parts
* Parts belong to a particular motorcycle make and model

The individual datasets will be stored in appropriate collections – hint – think about how the data is stored and used, one of the collections is more appropriate than others.

If the application is run with the ‘**service’** argument, then it will generate a basic **service** **report** similar to:

First Name Last Name Make Model Year Mileage

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

Jeanette Price Triumph Tiger 2012 22,683

Conrad Washington Yamaha Bolt 2015 751

Laurie Nash Honda CB1100 2015 1,106

Fred Fish BMW R75/5 1972 63,000

Conrad Washington Harley Davidson Sportster 2002 36,000

Fred Fish Suzuki V-Strom 1000 2007 31,680

Laurie Nash Ducati Monster 696 2009 32,000

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

If the ‘**inventory’** argument is used then, a report similar to the following is displayed (note this sample is truncated):

Make+Model Description Part# Price Quantity

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

Triumph+Tiger ASA-BOLT 7129907050 2.25 19

Yamaha+Bolt ASA-BOLT 7129907050 2.25 802

Honda+CB1100 BALANCE ROD WITH ROCKER 1277702426 666.03 0

BMW+R75/5 BALL PIN 7147202359 3.49 20

Triumph+Tiger BEARING SHELL 1277690501 40.95 18

Harley Davidson+Sportster BELT PULLEY 7727706466 512.92 1

Honda+CB1100 BELT WHEEL, REAR 7727678299 569.65 2

Suzuki+V-Strom BELTGUARD BOTTOM INT 6628526813 41.52 6

Yamaha+Bolt BELTGUARD, BOTTOM EXT 6628526814 25.64 59

...

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

And if ‘**customers’** is used then the customer data is printed out similar to the above reports.

BCMC will accept commandline options (full or short ) that will have the following effect (**each option is always preceded by a dash** ‘-‘):

|  |  |  |
| --- | --- | --- |
| Full | Short | Result |
| service | s | Print the service report |
| inventory | i | Print the inventory report |
| customers | c | Print the customer report |
| total | t | Print the inventory report adding a Value column and calculated value for each part and the total value of the inventory is added to the end of the report. |
| by\_description | D | Sorts the inventory report by part description name ascending. This is ignored if ‘inventory’ isn’t also specified |
| by\_count | C | Sorts the inventory report by part count ascending.  This is ignored if ‘inventory’ isn’t also specified |
| by\_join\_date | J | Sorts the customer report by join date.  This is ignored if ‘customers’ isn’t also specified |
| make=<make> | m | Filters the service or inventory report by make ascending |
| desc | d | Any sorted value is sorted in a descending order |

The following tests must pass (note multiple arguments can be specified in ANY order):

1. If no commandline parameters are specified, all three reports are printed.
2. If **service** (-s) is specified, the service report is printed.
3. If **inventory** (-i) is specified, the inventory report is printed.
4. If **customers** (-c) is specified, the customers report is printed.
5. If **service** **inventory** **customers** (-s –i –c) are specified, all three reports are printed.
6. If **customers** **by\_join\_date** (-c –J) are specified, the customers report is printed and the customers sorted by join date.
7. If **customers** **by\_join\_date desc** (-c –J –d) are specified, the customers report is printed and the customers sorted by join date in descending order.
8. If **inventory by\_description** (-i –D) are specified, the inventory report is printed and the parts sorted by description.
9. If **inventory by\_description desc** (-i -D –d) are specified, the inventory report is printed and the parts sorted by description in descending order.
10. If **inventory by\_count** (-i –C) are specified, the inventory report is printed and the parts sorted by quantity.
11. If **inventory by\_count total** (-i -C –t) are specified, the inventory report is printed and the parts sorted by quantity; a column is added to the end of the row specifying the inventory value (total) of the part (quantity x price). The total value of all of the displayed items is printed at the end of the report.
12. If **inventory desc by\_count total** (-i -d -C –t) are specified, the inventory report is printed and the parts sorted by quantity in descending order; the total is also printed at the end of the row. The total value of all of the displayed items is printed at the end of the report.
13. If **inventory make=Honda desc by\_count total** (-i -m=Honda -C –t) are specified, the report is similar to #12, but ONLY the parts containing the specified make are printed. Ex:

Inventory Report

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

Make+Model Description Part# Price Quantity Value

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

Honda+CB1100 FLAT NUT 6637659955 2.45 583 1,428.35

Honda+CB1100 HEX NUT 7119904920 2.37 465 1,102.05

Honda+CB1100 ISA SCREW 7129904261 3.44 323 1,111.12

Honda+CB1100 RETAINER SPRING 6717660163 4.08 138 563.04

Honda+CB1100 BULB 3217169203 5.15 114 587.10

Honda+CB1100 STOPPER 6712330765 8.03 84 674.52

Honda+CB1100 WASHER 7119907030 6.95 71 493.45

Honda+CB1100 SLEEVE 32727673990 9.05 53 479.65

Honda+CB1100 COUNTER NUT 32727678101 7.15 15 107.25

Honda+CB1100 VOLTAGE REGULATOR 1312346550 188.76 8 1,510.08

Honda+CB1100 BELT WHEEL, REAR 7727678299 569.65 2 1,139.30

Honda+CB1100 NIPPLE 1447687378 37.75 2 75.50

Honda+CB1100 FOOTREST, RIGHT 6717700914 80.31 1 80.31

Honda+CB1100 BALANCE ROD WITH ROCKER 1277702426 666.03 0 0.00

Value of current inventory: $9,351.72

## Submission Checklist

I have:

☐ Named my main class Bcmc (if something else -1)

☐ Created and submitted a runnable Jar file named Bcmc.jar in the out subfolder (0 otherwise)

☐ Met all the functional requirements (see ‘Grading’ below)

☐ Followed the java coding guidelines (up to -3 for violations)

☐ Used a file template to add my name & student number to all source files (-2 if not used)

☐ Used packages; the root package is my student number, (-3 if not followed)  
ex. package a00123456.…;

☐ Used proper object-oriented design (up to -3 for poor design)

☐ Included all source code & required resources (0 if source code is not submitted)

☐ Tested my solution on the commandline from the out folder (at least -1, but more may be deducted)

☐ Zipped up all my files into a single file named <your student number>.zip,   
ex. A00123456\_assignment1.zip

☐ Submitted my lab before the due date & time (0 otherwise)

## Grading

The assignment will be marked out of 10.

Except for the function requirements, for each item not followed in the checklist, you’ll lose 1 mark.

No customer report -2

No service report -2

No inventory report -5

No/incorrect sorting -1

No descending sort -1

No/incorrect total -1

No filtering -2