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 four 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.

If service and inventory and customers are specified,, or no commandline parameters are specified, then all are printed.

BCMC will accept commandline arguments that will have the following effect:

|  |  |
| --- | --- |
| Argument | Result |
| service | Print the service report |
| inventory | Print the inventory report |
| customers | Print the customer report |
| total | 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 | Sorts the inventory report by part description name ascending. This is ignored if ‘inventory’ isn’t also specified |
| by\_count | Sorts the inventory report by part count ascending |
| by\_join\_date | Sorts the customer report by join date |
| make=<make> | Filters the service or inventory report by make ascending |
| desc | Any sorted value is sorted in a descending order |

These requirements are subject to change.

## Submission Checklist

I have:

☐ Named my main class BCMC

☐ Met all the functional requirements

☐ Followed the java coding guidelines

☐ Run “Source > Format” on my project

☐ Used a file template to add my name & student number to all source files

☐ Used packages; the root package is my student number,   
ex. package a00123456.…;

☐ Used great object-oriented design

☐ Created a runnable Jar file

☐ Included all source code & required resources

☐ 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

For EACH requirement not followed in the checklist, you’ll lose 1 mark. OUCH!

## Grading

The assignment will be marked out of 10