Assignment by Imtiaz Ahmed

Business Requirement:

You have been hired as a consultant to complete a 1-month project. Developers in our company need a handy java API they can use to interact with a searchengine. You'll need to use the builder pattern to create the below JSON structure. This json structure is actually a query that can be submitted to a searchengine called elasticsearch.

Elasticsearch is an opensource tool one can download and use but that's not important. All your work will be in Java. Developers in our company using your builder API should be able to create json requests like this.

```
{
   "query": {
      "bool": {
         "must": [
            {
               "match": {
                  "item": "Milk"
               }
            },
            {
               "match": {
                  "item_type": "Dairy"
               }
            }
         ],
         "should": [
            {
               "match": {
                  "product_location": "New Mexico"
               }
            },
            {
               "match": {
                  "warehouse_number": 37
               }
            }
         ]
      }
   }
}
```

Here are the rules for the JSON structure.

You can have a single must or should section inside a bool section as shown. But keep in mind that inside of each one of these musts or shoulds, you can have nested bool sections.

The match section is simple, you can have any attribute name and it's value.
For example, the above json query is filtering for ONLY those items that are "Milk". And the "item_type" attribute has the value "Dairy". The product_location should be "New Mexico" with warehouse_number: 37.

You'll need to create a few java classes that will represent this JSON structure when they are converted to a JSON format.

Create the classes called Query, Bool, Must, Match and Test. You may also need a class called QueryBuilder, or whatever you want to name it. The Test class will contain the main method in which you will invoke the builder methods to create the instances of the Musts, Shoulds etc. and print out the composed objects json format to prove that the API works as expected.

Here is an example of how a developer expects to use the API:

```
QueryBuilder builder = new QueryBuilder();
builder.bool().mustMatch("item", "Milk").mustMatch("item_type", "Dairy");
builder.bool().shouldMatch("product_location","NewMexico")
.shouldMatch("warehouse_number", 37);
```

Remember the developers need to be able to nest bools inside of musts or shoulds if needed. So here's the case for a nested bool containing a must inside of an existing should section. The developers expect to use the api like this:

```
builder.bool().shouldMatch("lot_number", 307).bool().mustMatch("expiry_date", "Jan 2020");
```