Analytics Code Sim: Coffee Tasting

Background

Game Closure recently opened a retail website to sell imported coffee. We've done quite well, but we can do better! We'd like to use our historic data to build a recommendation engine.

Our sample set includes N people and M types of coffee, and each person has rated approximately half of the possible coffee types. The file is tab delimited, and each line contains a user id, a descriptive name for the coffee, and the person's rating:

```
Input:
  person_id<tab>coffee_name<tab>rating

Example:
5	Organic Fair Trade AA Guatemalan	1
5	Black Satin Costa Rican	4
5	Organic Decaf Mandheling Bolivian	3
5	Organic Fair Trade Decaf Swiss Water Malian	1
6	Organic Fair Trade Black Satin Peruvian	1
6	Organic Fair Trade AA Guatemalan	1
6	Fair Trade Decaf Black Satin Panamanian	2
6	Organic Decaf Mandheling Bolivian	5
```

Part One: Write a coffee-name parser.

Your first task is to write a coffee-name parser. It should extract properties based on the coffee's descriptive name. We've identified five properties:

```
Decaf        boolean
Organic      boolean
Fair trade   boolean
Adjective    String
Country      String
```

For example, given "Organic Decaf Mandheling Bolivian", it should extract:

```
Decaf        True
Organic      True
Fair trade   False
Adjective    Mandheling
Country      Bolivia
```

To help get you started, we've provided a sample framework written in Java. Using this framework, you'll implement the parser as a static factory:

Example

```
// Constructor example:
Coffee c = Coffee.fromName("Fair Trade Decaf Black Satin Panamanian");
```

We've also provided a command-line tool **GCCoffee** to assist in testing. Please configure the tool such that you can parse coffee names from the command line; for example:

```
// Command line example:
$ java GCCoffee parse Fair Trade Decaf Black Satin Panamanian
Fair Trade Decaf Black Satin Panamanian
       Decaf        true
       Organic      false
       Fair trade   true
       Adjective    Black Satin
       Country      Panama
```

Lastly, be sure to convert the adjectival form of a country to the noun form. Specifically, you'd change *Panamanian* to *Panama*. Below is a table of countries that are used in the data set.

```
"Balinese", "Bali"
"Bolivian", "Bolivia"
"Brazilian", "Brazil"
"Costa Rican", "Costa Rican"
"Dominican", "Dominican Republic"
"Salvadorean", "El Salvador"
"Ethiopian", "Ethiopia"
"Guatemalan", "Guatemala"
"Indian", "India"
"Kenyan", "Kenya"
"Malian", "Mali"
"Mexican", "Mexico"
"Panamanian", "Panama"
"Peruvian", "Peru"
"Sumatran", "Sumatra"
```

Part Two: Summarize the data

Let's summarize the data set! Using the parser that you just wrote, please iterate through the input file and create a summary table. You should expose this through GCCoffee:

Example:

```
java GCCoffee summarize coffee_ratings.txt
Total people 140
Total coffee types  50
```

```
Decaf
       True   20
       False  30
Organic
       True   10
       False  40
Fair trade
       True   5
       False  45
Adjective
       Bright 15
       Supremo      35
Country
       India  10
       Peru   40
```

Part Three: Build a recommendation engine

Let's build the recommendation engine! Your job is to recommend three new coffee types to each person. The output should include the user's id, the coffee's name, and the predicted rating:

Example:

```
java GCCoffee recommend coffee_ratings.txt
954512 Decaf Black Satin Balinese 4
954512 Black Satin Balinese      4
954512 Organic Longberry Mexican 4
954513 Honey Burst Dominican     5
954513 Organic Decaf Cuzcachapa Balinese      4
954513 Organic Fair Trade AA Guatemalan 4
954514 Organic Fair Trade Black Satin Peruvian 4
954514 Organic Fair Trade Decaf Swiss Water Malian   4
954514 Fair Trade Decaf Black Satin Panamanian 3
```