CSCI 136 Programming Exam #2
Fundamentals of Computer Science II
Spring 2015

This part of the exam is like a mini-programming assignment. You will create a program, compile it, and debug it as necessary. This part of the exam is open book and open web. You may use code from the course web site or from your past assignments. When you are done, submit all your Java source files to the Moodle exam #2 dropbox. Please ***double check you have submitted all the required files***.

You will have 100 minutes. No communication with any non-staff members is allowed. This includes all forms of real-world and electronic communication.

Grading. Your program will be graded on correctness and to a lesser degree on clarity (including comments) and efficiency. You will lose a substantial number of points if your code does not compile or crashes on typical inputs.

"Never, never, never give up."
-Winston Churchill

Overview. You are building a program to search a database of famous quotations. Each quote has some text, an author, a category, and zero or more user ratings on a 1 to 5 star scale. You will be building a class `Quote` to represent an individual quotation, a class `Quotes` to hold a collection of quotations, and a program `SearchQuotes` that can be used to perform searches. To get started, download the file: http://katie.mtech.edu/classes/csci136/quotes.zip

Part 1: Quote. The class `Quote` represents a single quotation. A `Quote` knows things like the text of the quote, the author of the quote, and the category of the quote. A `Quote` also keeps track of the different ratings give to the quote by zero or more users. A `Quote` can do things like create itself based on a semicolon-delimited line of text, return the quote's text/author/category, add a new user rating, calculate the average of all user ratings, return a string representation of itself, and determine if a any of the quote's fields (text, author, or category) contains a given bit of text. Here is the API:

```
class Quote
---------------------------------------------------------
            Quote(String line)
      String getText()
      String getAuthor()
      String getCategory()
        void addRating(int rating)
      double getAverageRating()
      String toString()
     boolean contains(String searchText)
```

A `Quote` is created from a single line of text which has four columns separated by semicolons: the text of the quote, the author, the category, and a list of user ratings (if any). Here are a few examples:

```
Love is a better teacher than duty.;Albert Einstein;teacher;1,5,5,4
The future will be better tomorrow.;Dan Quayle;future;
Be kind whenever possible. It is always possible.;Dalai Lama;motivational;5
```

Hint: the `String` method `split()` can be used to separate text based on some delimiter. Note that if there are no ratings, `split()` will only return 3 columns since there is nothing after the final semicolon. If a quote has ratings, the numeric ratings will be provided as a comma-delimited list.

User ratings of a quote are always an integer value of 1, 2, 3, 4, or 5. If a quote has no ratings, its average rating is considered to be 0.0. Attempting to add an invalid rating throws a `RuntimeException`.

The `contains()` method does a case insensitive search for the given text. A quote matches if the search text appears anywhere in the text, author, or category of a quote. So for example the search text "the" would match the text of the quote "Their name is forgotten" since "the" appears somewhere as part of the word "Their". *Hint:* you may want to use the `String` methods `contains()` and `toLowerCase()`.

Further details about each method is provided in the comments in the stub code. We have provided a test `main()` method. Here is the output of our solution:

```
% java Quote
Love is a better teacher than duty. -Albert Einstein (category: teacher, rating: 3.8)
getText             = 'Love is a better teacher than duty.'
getAuthor           = 'Albert Einstein'
```

```
getCategory         = 'teacher'
getAverageRating    = 3.75
after addRating(1)  = 3.20
after addRating(6)  = 3.20
after addRating(-6) = 3.20
contains("the")     = false
contains("tHE")     = false
contains("dan")     = false
contains("tiv")     = false

The future will be better tomorrow. -Dan Quayle (category: future, rating: 0.0)
getText             = 'The future will be better tomorrow.'
getAuthor           = 'Dan Quayle'
getCategory         = 'future'
getAverageRating    = 0.00
after addRating(1)  = 1.00
after addRating(6)  = 1.00
after addRating(-6) = 1.00
contains("the")     = true
contains("tHE")     = true
contains("dan")     = true
contains("tiv")     = false

Be kind whenever possible. It is always possible. -Dalai Lama (category: motivational,
rating: 5.0)
getText             = 'Be kind whenever possible. It is always possible.'
getAuthor           = 'Dalai Lama'
getCategory         = 'motivational'
getAverageRating    = 5.00
after addRating(1)  = 3.00
after addRating(6)  = 3.00
after addRating(-6) = 3.00
contains("the")     = false
contains("tHE")     = false
contains("dan")     = false
contains("tiv")     = true
```

Part 2: Quotes. The class Quotes represents a collection of quotes. It keeps track of a list of quotations. It can do things like load in quotations from a text file, return the number of loaded quotations, and return a string representation of the highest rated quote that matches zero or more search terms. Here is the API:

```
class Quotes
---------------------------------------------------------------------
       Quotes(String filename) throws FileNotFoundException
   int getSize()
String getHighestRated(String [] terms)
```

A `Quotes` collection is created by loading quotes from a text file. Each line in this text file is a single quote and appears in the form parsed by the constructor of the `Quote` class.

The collection can be searched by providing an array of zero or more bits of search text. A quote must contain each of the bits of search text in order to be eligible for return by `getHighestRated()`. If multiple quotes match a given set of search terms, the term with the highest average user rating is returned. In the event of a tie, return the quote that appeared first in the text file. If no search terms are provided (an

empty array), all quotes are considered to match and the highest rated quote occurring first in the file is returned.

Further details about each method is provided in the comments in the stub code. We have provided a test `main()` method. Here is the output of our solution:

```
% java Quotes
getSize() = 75966
terms0 -> Alas, after a certain age every man is responsible for his face. -Albert
Camus (category: age, rating: 5.0)
terms1 -> If an Internet company steals content, they shut it down. And let me tell
you, Apple France, Yahoo France or Google France, none of them have gone out of
business. -Harvey Weinstein (category: business, rating: 5.0)
terms2 -> It was the experience of mystery - even if mixed with fear - that engendered
religion. -Albert Einstein (category: experience, rating: 5.0)
terms3 ->
terms4 -> People never cease to amaze me. -Tina Yothers (category: amazing, rating:
2.3)
Exception when opening bogus file
```

Part 3: SearchQuotes. This program makes use of the `Quotes` class to allow a single search to be executed against a list of quotes. If run with no command-line arguments, it prints out a helpful error message and exits:

```
% java SearchQuotes
SearchQuotes <filename> [term1] [term2] ...
```

If the specified filename is not found, it prints out a different helpful error message and exits:

```
% java SearchQuotes bogus.txt
Failed to load file: bogus.txt
```

If the specified filename was successfully loaded, the program prints out the number of quotes loaded and the filename that they were loaded from. If no search terms are provided, the program returns the highest rated quote occurring earliest in the file:

```
% java SearchQuotes quotes.txt
Loaded 75966 quotes from quotes.txt
Alas, after a certain age every man is responsible for his face. -Albert Camus
(category: age, rating: 5.0)
```

If additional command-line arguments are provided, they are used as search terms that must match in the returned highest rated quote. Here are some further example runs:

```
% java SearchQuotes quotes.txt wine
Loaded 75966 quotes from quotes.txt
Beauty is worse than wine, it intoxicates both the holder and beholder. -Aldous Huxley
(category: beauty, rating: 5.0)

% java SearchQuotes quotes.txt wine white
Loaded 75966 quotes from quotes.txt
```

```
I'm only drinking white wine because I'm on a diet and I don't eat. -Oliver Reed
(category: diet, rating: 0.0)

% java SearchQuotes quotes.txt Winston Churchill
Loaded 75966 quotes from quotes.txt
No part of the education of a politician is more indispensable than the fighting of
elections. -Winston Churchill (category: education, rating: 5.0)

% java SearchQuotes quotes.txt education politician indispensable
Loaded 75966 quotes from quotes.txt
No part of the education of a politician is more indispensable than the fighting of
elections. -Winston Churchill (category: education, rating: 5.0)

% java SearchQuotes quotes.txt education politician indispensable fighting elections
Loaded 75966 quotes from quotes.txt
No part of the education of a politician is more indispensable than the fighting of
elections. -Winston Churchill (category: education, rating: 5.0)

% java SearchQuotes quotes.txt education yucky
Loaded 75966 quotes from quotes.txt

% java SearchQuotes quotes.txt THE QUICK
Loaded 75966 quotes from quotes.txt
Intelligence is quickness in seeing things as they are. -George Santayana (category:
intelligence, rating: 5.0)
```