COS125 - Precept 9 (IO)

1 Counting Matches

Download `precept9.zip` from the precepts webpage, unzip and open the project folder. The folder contains two files with personal and healthcare information on (fictitious) residents of Massachussets, `healthcare_data.txt` and `personal_data.txt`. They are formatted similarly to a CSV file, with three modifications for your convenience:

- whitespace characters inside entries were removed;
- empty fields are identified by the character `*`; and
- fields are separated by a space (instead of a comma).

Write a program `CountMatches.java` that reads a string from the command line and counts the number of entries in the database that are exactly equal to it. (Recall that equality between strings `s` and `t` is checked with `s.equals(t)`, not the `==` operator.)

Then, count the number of matches for the strings below in `healthcare_data.txt`:

2	3	27	29	33	40	50	70	72	89	100
01060	01105	01604	01835	01841	01940	02138	02148	02149	02641	02726

Test your code by comparing the results with those you obtained in the previous precept!

2 Database Search

Let's now semi-automate the manual de-anonymization strategy of the previous precept: copy `CountMatches.java` into a file named `DatabaseSearch.java` and adapt it so it reads two strings, `gender` and `zip`, then prints every entry in `healthcare_data.txt` that match *both* the gender and zip code values.

Note that in order to print an entire entry, you need to use a `StdIn` method that reads all tokens until the next line (`StdIn.readString()` only reads until the next whitesapce). You may also find the `String` function `split()` useful.

3 Music with MIDI

Create a program `PlayWithMidi.java` that receives two command-line arguments – a `String` instrument name and an `int` tempo – sets the instrument and tempo accordinagly and plays the melody in `StdIn` where each note is defined by an `int` MIDI note and a `double` duration.

Your code should interpret the string `piano` as the instrument that `StdMidi` specifies by the constant `ACOUSTIC_GRAND_PIANO`; and `violin` as that corresponding to `VIOLIN`.

You can test your code with the following input, which represents "Twinkle Twinkle Little Star" starting at middle C:

```
> java-introcs PlayWithMidi piano 100
60 1.0 60 1.0 67 1.0 67 1.0 69 1.0 69 1.0 67 2.0 65 1.0 65 1.0 64 1.0 64 1.0
62 1.0 62 1.0 60 2.0
```

4 Database Search Deluxe (Bonus)

Copy `DatabaseSearch.java` into `DatabaseSearchDeluxe.java`, and adapt it so it reads an arbitrary number of pairs of strings, which specify feature names and values. Then it should read `healthcare_data.txt` from `StdInt` and prints the entries whose corresponding features match all such values.

Note that the outputs of the two commands below should be exactly the same for any strings in place of `[gender]` and `[zip]`. (And you can use this to test your new code!)

```
> java-introcs DatabaseSearch [gender] [zip] < healthcare_data.txt
> java-introcs DatabaseSearch GENDER [gender] ZIP [zip] < healthcare_data.txt
```