Section Handout 7

Based on handouts by Eric Roberts and Mehran Sahami

Problem One: Word Walks

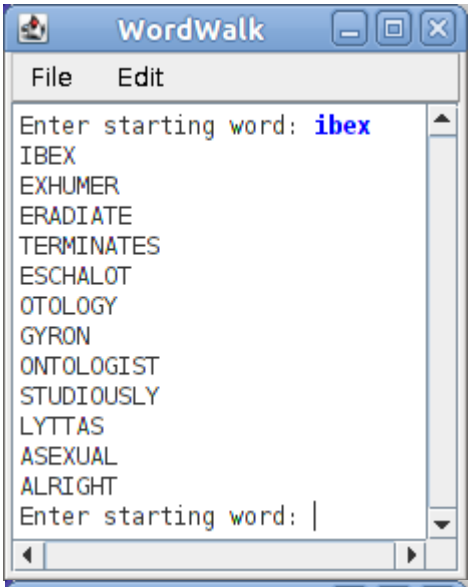
A **word walk** is a series of words where the last two letters of each word are also the first two letters of the next word. For example:

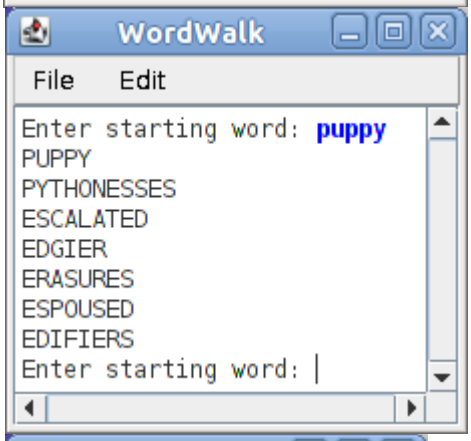
```
CODE
  DESIRE
      REWRITE
          TEMPERATE
                TEATIME
                    MEMENTO
                        TORRENT
```

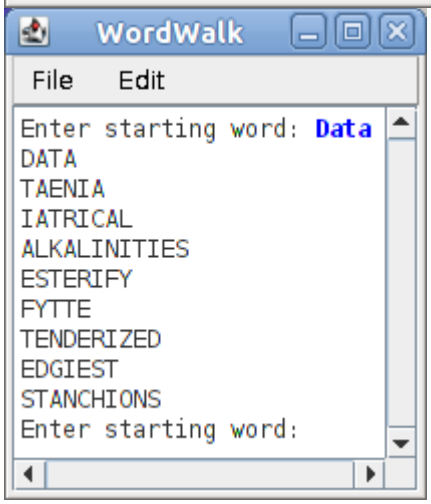
This word walk ends at "TORRENT" because there are no English words that start with "NT."

Your task is to write a program that generates random word walks based on some starting word. Your program should read in the dictionary file **`dictionary.txt`** for its list of words, then prompt the user for a starting word. Given that starting word, you should generate a random word walk starting from that word by repeatedly picking a random word whose first two letters are the same as the last two letters of the current word. Once you arrive at a word whose last two letters aren't the start of any other words, you should stop and prompt the user to enter another starting word.

Some sample runs of the program are shown to the right.





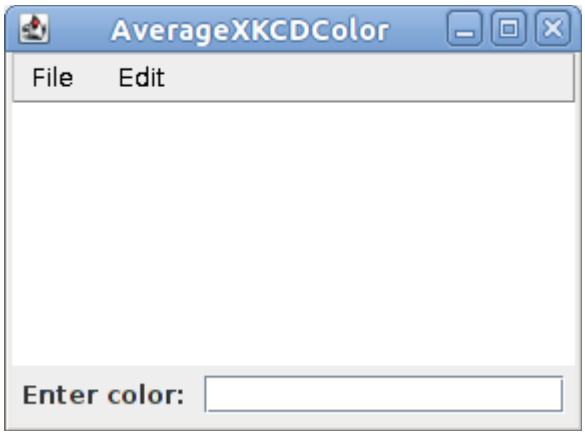


Problem Two: The Average Color

Last Friday, we worked through an example using the xkcd color survey data set, which consisted of a list of RGB triplets along with the name of the color with that triplet. Our program plotted the colors on a color wheel so that we could see the boundaries between different colors.

A different question we might ask is, given the name of a color, what that color actually looks like. The xkcd color set has a huge number of colors associated with each name, so we can't just pick one of them and say that it is *the* definitive color. Instead, we could consider averaging together all of the colors labeled with a given name to get one single color that represents (most likely) what that color actually looks like. Since we have the red, green, and blue components of all of the colors, we can compute the average color by finding the average red, green, and blue components of the data points for that color.

Your task is to write a program that, when started up, looks like this:



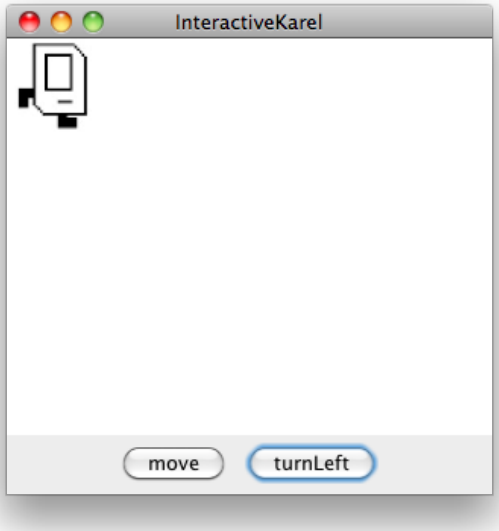
Whenever the user types the name of a color into the text area at the bottom of the screen, your program should find all colors that have that name, then average them together to get the average color. You should then set the background color for the program to be that color, which will fill the window with the given color. You can set the background color by calling **`setBackground`** with the **`Color`** of your choice. To create a **`Color`** out of its red, green, and blue components, you can use the **`Color`** constructor by writing **`new Color(r, g, b)`**.

In lecture, we wrote a method **`loadColorsFile`** that loads the xkcd color data from the appropriate data file. You can assume that this method exists and is written for you. Its signature is

```
private HashMap<String, ArrayList<int[]>> readXKCDColors()
```

This **`HashMap`** maps from the name of a color (in lower-case) to an **`ArrayList`** of the RGB triplets, each of which is represented by an **`int[]`** with three entries holding the red, green, and blue components, respectively.

Problem Three: Karel the Robot Returns!



Write a Graphics Program that lets you move Karel around by clicking the "move" button and the "turnLeft" button. You can assume you are given four images for each of the directions that Karel can face called **`KarelEast.jpg`**, **`KarelWest.jpg`**, **`KarelSouth.jpg`**, and **`KarelNorth.jpg`**. If Karel would move off the screen, instead just keep Karel where he is.