Google Applied CS with Android

Simple Anagrams Game

Anupam Das
Google Android with CS facilitator

What is Android ?

Android is a mobile operating system developed by Google.

Google

Creating an Android app for a simple anagram game.

What is anagram?

What is anagram?

According to Wikipedia,

An anagram is word or phrase formed by rearranging the letters of a different word or phrase, typically using all the original letters exactly once.

For example, the word "allergy" can be rearranged into "gallery", "largely", "regally".

Requirements

1. Laptop / PC with proper internet connectivity and power.
2. Log in to https://cswithandroid.withgoogle.com/
3. Follow my steps.

The mechanics of the game

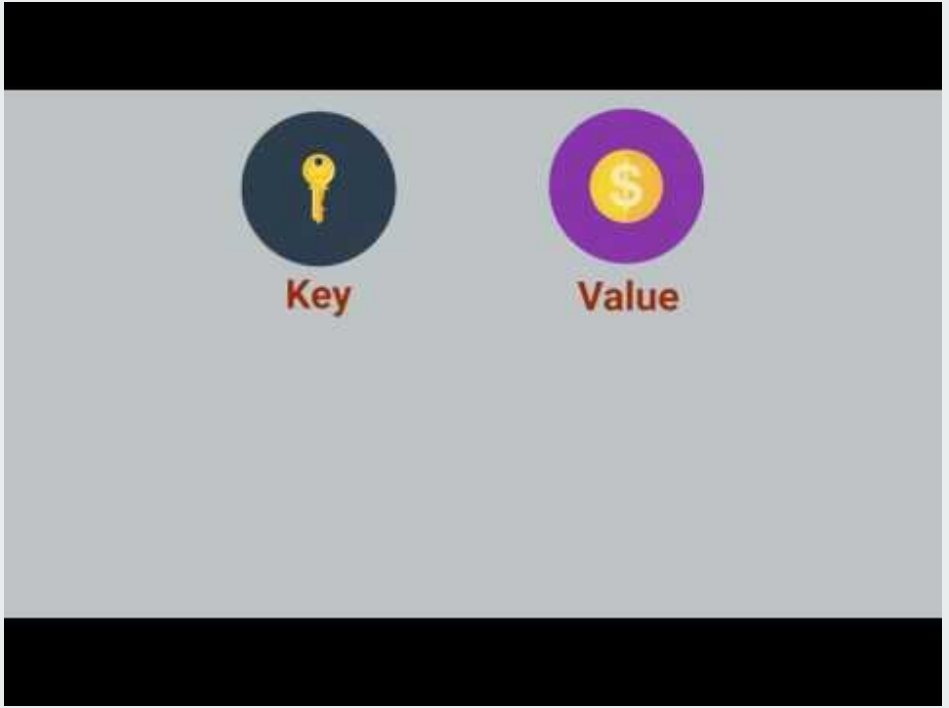
The mechanics of the game

- The game provides the user with a word from the dictionary.
- The user tries to create as many words as possible that contain all the letters of the given word plus one additional letter.

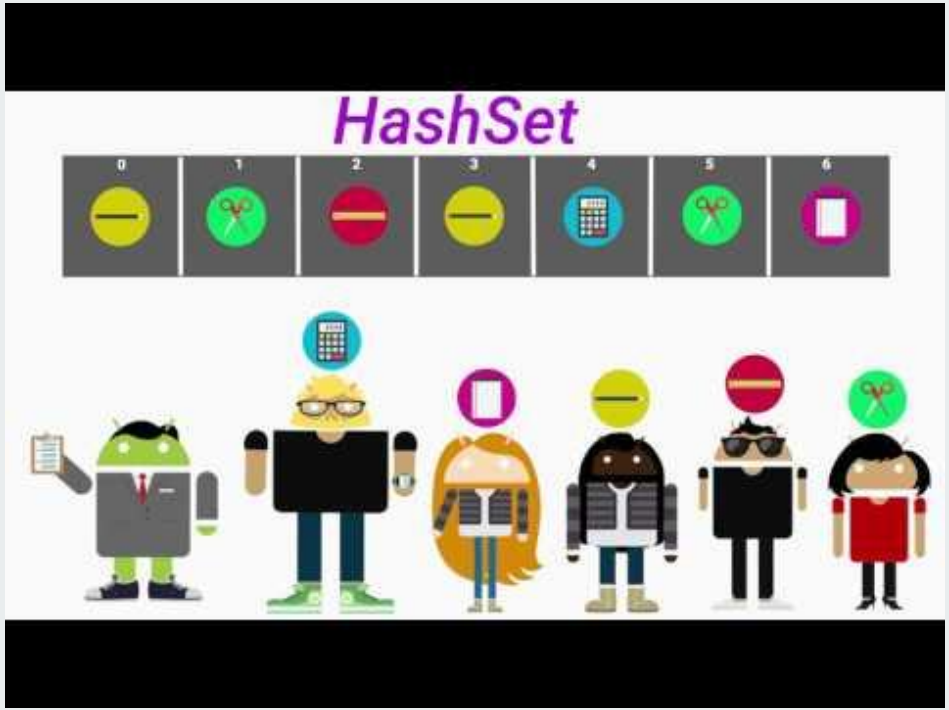
  Note that adding the extra letter at the beginning or the end without reordering the other letters is not valid. For example, if the game picks the word 'ore' as a starter, the user might guess 'rose' or 'zero' but not 'sore'.

- The user can give up and see the words that they did not guess.

HashMaps



Hashsets



Tour of the code

- AnagramsActivity

  Activity is a single, focused thing that the user can do.

- AnagramDictionary

  This class will store the valid words from the text file and handle selecting and checking words for the game.

Milestone 1

AnagramDictionary

First task will be to advance the implementation of the AnagramDictionary's ***constructor***. Each word that is read from the dictionary file should be stored in an ArrayList (called ***wordList***).

AnagramDictionary

```
private ArrayList<String> wordList = new ArrayList<String>();

public AnagramDictionary(Reader reader) throws IOException {
  BufferedReader in = new BufferedReader(reader);
  String line;
  while ((line = in.readLine()) != null) {
     String word = line.trim();
     wordList.add(word);
  }
}
```

AnagramDictionary

Create a helper function (call it ***sortLetters***) that takes a String and returns another String with the same letters in alphabetical order (e.g. "post" -> "opst").

Determining whether two strings are anagrams is then a simple matter of checking that they are

- the same length (for the sake of speed) and
- checking that the sorted versions of their letters are equal.

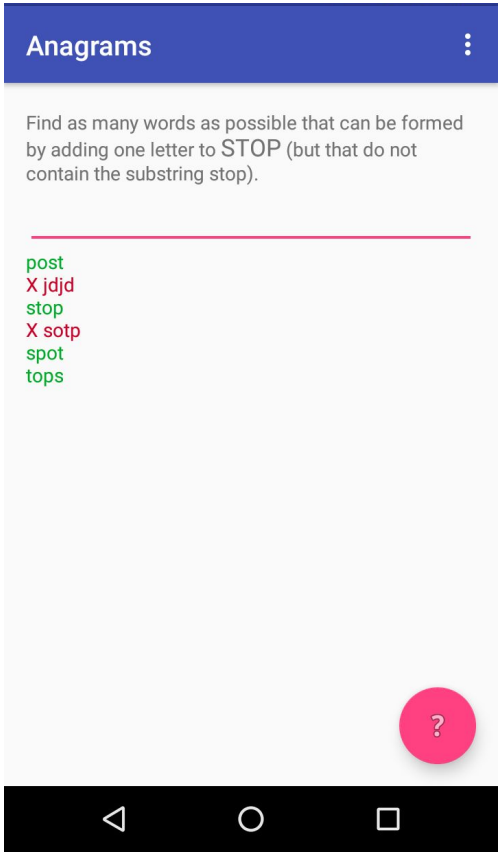
SortLetters

```
private static String sortLetters(String word) {
    char[] chars = word.toCharArray();
    Arrays.sort(chars);
    return new String(chars);
}
```

getAnagrams

```
public List<String> getAnagrams(String targetWord) {
   ArrayList<String> result = new ArrayList<String>();
   for (String word : wordList) {
      if (sortLetters(targetWord).equals(sortLetters(word))) {
         result.add(word);
      }
   }
   return result;
}
```

Now you have an working, simple anagram checker app.



We will create two new data structures (in addition to wordList):

- A HashSet (called ***wordSet***) that will allow us to rapidly (in O(1)) verify whether a word is valid.
- A HashMap (called ***lettersToWord***) that will allow us to group anagrams together.

  We will do this by using the sortLetters version of a string as the key and storing an ArrayList of the words that correspond to that key as our value.

  For example, we may have an entry of the form: key: "opst" value: ["post", "spot", "pots", "tops", ...].

wordSet and lettersToWord

```
private HashSet<String> wordSet = new HashSet<>();
private HashMap<String, ArrayList<String>> letterToWords = new HashMap<String, ArrayList<String>>();
```

wordSet and lettersToWord

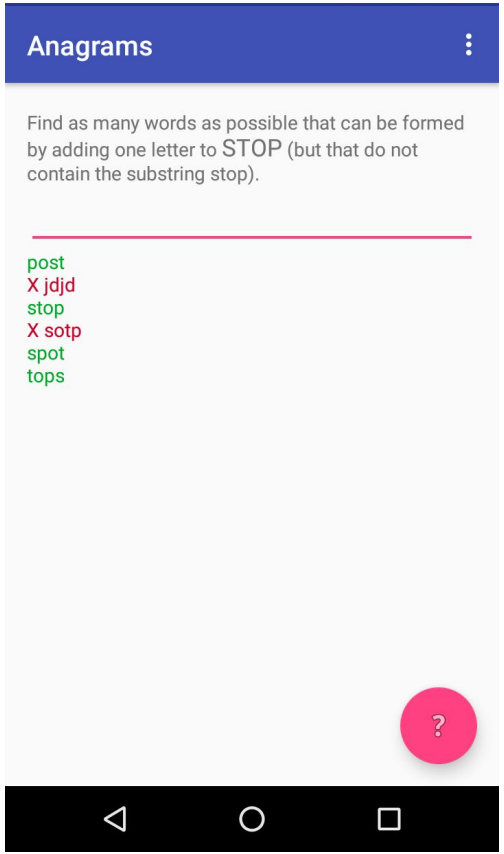
```
public AnagramDictionary(Reader reader) throws IOException {
    BufferedReader in = new BufferedReader(reader);
    String line;
    while ((line = in.readLine()) != null) {
        String word = line.trim();
        wordSet.add(sortLetters(word));
        if (!letterToWords.containsKey(sortLetters(word))) {
            letterToWords.put(sortLetters(word), new ArrayList<String>());
        }
        letterToWords.get(sortLetters(word)).add(word);
    }
}
```

wordSet and lettersToWord

```
public List<String> getAnagrams(String targetWord) {
    return letterToWords.get(sortLetters(targetWord));
}
```

Hurrah! Milestone 1 completed

Now you have an working, faster and simple anagram checker app.



Milestone 2

isGoodWord

Your next task is to implement isGoodWord which checks:

the provided word is a valid dictionary word (i.e., in wordSet), and
the word does not contain the base word as a substring.

Example

With the base word 'post':

Input	Output
isGoodWord("nonstop")	true
isGoodWord("poster")	false
isGoodWord("lamp post")	false
isGoodWord("spots")	true
isGoodWord("apostrophe")	false

isGoodWord

```
public boolean isGoodWord(String word, String base) {
    if (word.contains(base))
        return false;
    if (wordSet.contains(sortLetters(word))) {
        if (letterToWords.get(sortLetters(word)).contains(word))
            return true;
    }
    return false;
}
```

getAnagramsWithOneMoreLetter

Finds all anagrams that can be formed by adding one letter to that word.

getAnagramsWithOneMoreLetter

```
public List<String> getAnagramsWithOneMoreLetter(String word) {
    ArrayList<String> result = new ArrayList<String>();
    for (int i = 0; i < 26; i++) {
        String newWord = word + (char) ('a' + i);
        String sortedNewWord = sortLetters(newWord);
        if (wordSet.contains(sortedNewWord)) {
            for (int j = 0; j < letterToWords.get(sortedNewWord).size(); j++)

                if (!letterToWords.get(sortedNewWord).get(j).contains(word)) {
                    result.add(letterToWords.get(sortedNewWord).get(j));
                }
        }
    }
    return result;
}
```

pickGoodStarterWord

Pick a random starting point in the wordList array and check each word in the array until you find one that has at least MIN_NUM_ANAGRAMS anagrams.

pickGoodStarterWord

```
private HashSet<String> wordSetWithMin = new HashSet<String>();

public AnagramDictionary(Reader reader) throws IOException {
    BufferedReader in = new BufferedReader(reader);
    String line;
    while ((line = in.readLine()) != null) {
        String word = line.trim();
        wordSet.add(sortLetters(word));
        if (!letterToWords.containsKey(sortLetters(word))) {
            letterToWords.put(sortLetters(word), new ArrayList<String>());
        }
        letterToWords.get(sortLetters(word)).add(word);
    }

    for (String str : wordSet) {
        if (letterToWords.get(str).size() >= MIN_NUM_ANAGRAMS) {
            wordSetWithMin.add(str);
        }
    }
}
```

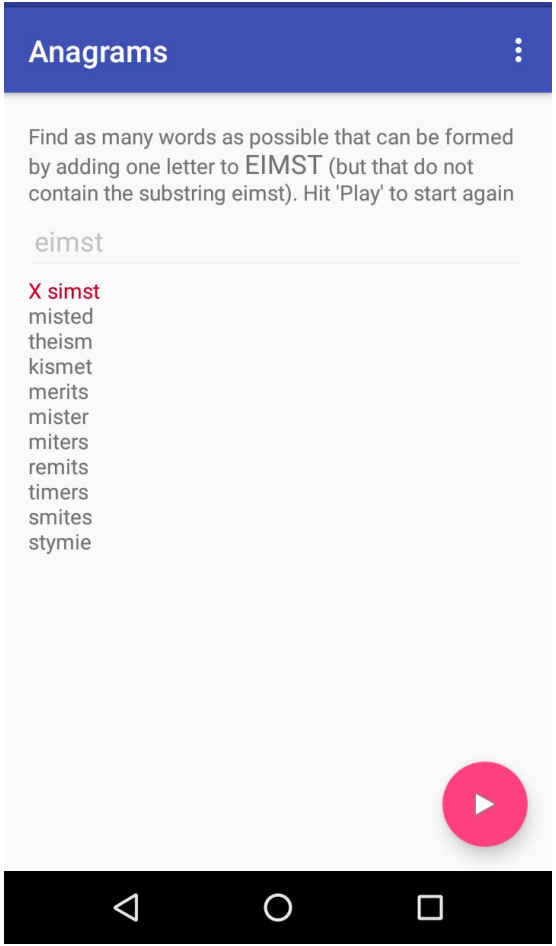
pickGoodStarterWord

```
public String pickGoodStarterWord() {
    Random random = new Random();
    int randomPos = random.nextInt(wordSetWithMin.size());

    int count = 0;
    for (String str : wordSetWithMin) {
        if (count == randomPos) {
            return str;
        }
        count++;
    }
    return "stop";
}
```

Hurrah! Milestone 2 completed

Now you have an working, faster and simple anagram checker app.



Milestone 3

Refactoring

Get all 4 letter words as starter word.

Refactoring

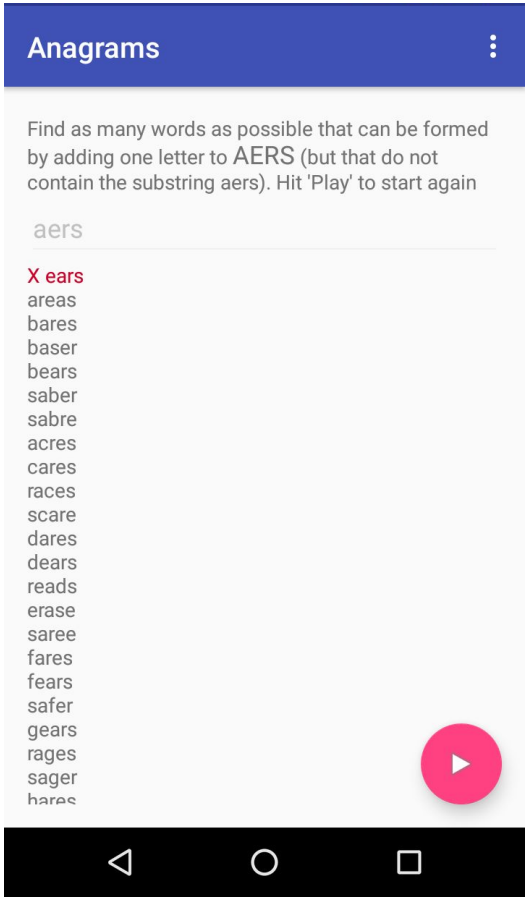
```
private static final int DEFAULT_WORD_LENGTH = 4;

public AnagramDictionary(Reader reader) throws IOException {
    BufferedReader in = new BufferedReader(reader);
    String line;
    while ((line = in.readLine()) != null) {
        String word = line.trim();
        wordSet.add(sortLetters(word));
        if (!letterToWords.containsKey(sortLetters(word))) {
            letterToWords.put(sortLetters(word), new ArrayList<String>());
        }
        letterToWords.get(sortLetters(word)).add(word);
    }

    for (String str : wordSet) {
        if (letterToWords.get(str).size() >= MIN_NUM_ANAGRAMS && str.length() == DEFAULT_WORD_LENGTH) {
            wordSetWithMin.add(str);
        }
    }
}
```

Hurrah! Milestone 3 completed

Now you have an working, faster and simple anagram checker app with starting word of 4 letters.



References

- Google Applied CS Skills Course Materials *https://appliedcsskills.withgoogle.com/unit?unit=1&lesson=2*
- Wikipedia *https://en.wikipedia.org/wiki/Anagram*
- Google Students Video Links:
  - *https://youtu.be/eMymKAFYaCs*
  - *https://youtu.be/O-zTuD8JRbE*

Questions?

Thanks!

Feel free to share what you have learned in social media (Facebook, Google +, Twitter.) and with your friends.

Anupam Das
opticod

Google Android with CS Facilitator

http://opticod.com/