CMSC B113 - Computer Science 1
Fall 2020
Homework Assignment #6

Overview

In the class this week, and in Lab#9 we explored safe passwords. For example, in 2019, Bryn Mawr College's safe password guidelines were as follows:

Your password must be at least 10 characters long and must contain at least one uppercase letter, lowercase letter, and number and/or symbol.

In Lab#9 (and in class) we wrote a password verifier that, given a possible password, would verify if the password meets the criteria specified above. While the above specification generally leads to secure passwords, it makes remembering them hard for many people (Did I have "Tweety2020!" or "Tweety 2020!"??).

Another scheme for creating passwords that guarantees safety and yet makes them easier to remember is to use ***passphrases***. A passphrase is made up of two or more common words: For example:

`Borrow, or rob?`
(which is a palindrome, by the way!).

or:

`Taco cat.`

To ensure password safety, the Electronic Frontier Foundation (eff.org) has created a curated list of over 7000 words that can be used for creating safe passphrases. Here are the first 5 lines from their list:

```
7776
11111 abacus
11112 abdomen
11113 abdominal
11114 abide
...
```

These words are provided in a file: **`eff_wordlist.txt`** (provided where you got this document). In the file there are 7776 such words (first line of file indicates this). Each word also has a 5-digit code next to it. To create a passphrase, first you decide how many words it is going to be. Let us say, we will create a two-word passphrase. Then, you will take five 6-sided dice and roll them. When you juxtapose all the five rolls together, you get a code. For example, if you rolled, 4, 6, 5, 1, 6 to get the code: 46516. Next, you look up the passphrase words list to locate the word with the code 46516. It happens to be the word **`rebel`**. That becomes your first word in the

passphrase. Repeat the process for the second word. Say you rolled, 6, 6, 6, 6, 1 (code: 66661) to get the word **zone**. Then your passphrase is **rebel zone**.

See, it is easily remembered. You should capitalize and add some punctuation as you fit. But how safe is this passphrase?

If you pick a passphrase of length 1 word out of the list, any hacker can crack your password by trying out all 7776 words. That would be very easy. For passphrases of two words, the hacker would need to try 7776*7776=60,466,176 different passphrases to crack your password. We may be getting somewhere! 60 million tries is a small number for any computer hacker to try, so you would typically need to have a passphrase that is 5 or more words long. For a five-word passphrase it would require 15,000,000,000,000,000,000 (or 15 quintillion) tries! We will be well protected.

In this exercise you will write a Java program that uses the list of words provided in the file **eff_wordlist.txt** to generate a passphrase of a given length. For example,

```
$ java PassPhrase 4 < eff_wordlist.txt
Your passphrase is: enviable swirl aqua onion
```

Here are some other example passphrases:

```
marathon strut angles spoils
oval strained vanity rug
stunning prize myth strut
refund trouble impromptu rebirth
```

If you can get a set of five dice (or just one will do, use it five times), try to generate some passphrases of your own by rolling the dice, creating a 5-digit code, and then looking up the word. One quick way to find the word in Linux is to use the command **grep**:

```
$ grep 66661 eff_wordlist.tx
66661 zone
$ grep 56622 eff_wordlist.txt
56622 stuck
```

So, the 2 word pass phrase is: **zone stuck**

Learning Outcomes

In completing this assignment, you will learn to:

- Implement static methods in Java
- Learn how to use the Java String API
- Pass arrays as arguments to methods
- Write methods that use arrays as return values

Submitting Your Solution

Due to the strike this is a self-study Assignment. No submission is required, and no grade will be awarded..