Chapter 6 Additional Practice

Loops and Strings are pretty important! Here are some more practice questions to sharpen your skills. Upload each class to your existing Ch6 GitHub repository; file names are given for each question.

Useful String methods:

str.toUpperCase() returns a new String that is the same as str, but all letters are capitalized.

str.toLowerCase() returns a new String that is the same as str, but all letters are uncapitalized.

str.contains(String A) returns true or false depending on if String A is contained within str.

str.charAt() returns the character at a given index as a char.

str.indexOf(String A) returns the index of the first occurrence of String A, or -1 if there is no such occurrence. There is also **str.lastIndexOf(String A)** if you want to search from the end instead.

str.indexOf(String A, int start) returns the index of the first occurrence of String A starting at index "start", or -1 if there is no such occurrence. There is also **str.lastIndexOf(String A, int start)** if you want to search backward from "start" instead.

str.indexOf(String A) Like indexOf above, but assumes you are searching from the start of the string. There is also **str.lastIndexOf(String A)** if you want to search from the end of the String instead.

str.substring(int start, int end) returns the substring from index start up to (but not including) index end.

str.substring(int start) also works, and returns the substring in str starting from start and going to the end of str.

Question 1: PPAP.java - The first four lines in the lyrics of the 2016 hit song "PPAP" by artist Kosaka Daimaou are as follows:

I have a pen, I have an apple.
Uh! Apple pen.

I have a pen, I have pineapple.
Uh! Pineapple pen.

Write a method that reads a String from the terminal that follows the pattern of the first and third lines of this epic song:

"I have a <object 1>, I have a <object 2>".

The method should return a String of the pattern

"Uh! <Object 2> <object 1>."

You can assume a comma always follows directly after the first object word, the object words are always preceded by the text "I have a", and that there are no spaces or commas in either object word (Note: we are not worrying about a / an. Assume there will be no an). In your return String, include a space between <object 2> and <object1> and capitalize the first letter of object 2. (3 marks)

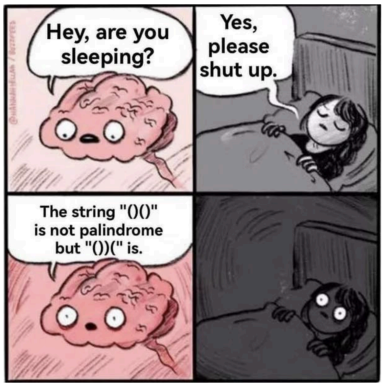
For example:

"I have a pointer, I have a null" will produce "Uh! Null pointer"

"I have a scholar, I have a AP" will produce "Uh! AP scholar"

Method Signature: `public static String ppap (String str)`

Question 2 - Palindrome.java: A palindrome is a word that can be read the same forwards and backwards, such as racecar or aibohphobia (the fear of palindromes). Write a method that reads a String from the terminal and prints to the terminal whether or not that String is a palindrome. Ignore capitalization and non-letter characters, so “A man, a plan, a canal: Panama.” Is a palindrome, as is “Amy, must I jujitsu my ma?”



How do you ignore non-letter characters? You could make a new String variable that contains them:

```
String nonLetters = ",.:;()[]{}!@#$%^&* ";
```

When you want to check if a substring is a nonLetter character, you can use something like:




```
if (nonLetters.contains(str)) { … }
```

Or, it might be more useful to see if a particular substring is a letter:

```
if (!nonLetters.contains(str)) { … }
```

Hint: Get your method working for just a single word such as tacocat. Then, improve your algorithm to deal with capital letters and nonletter characters (such as spaces and punctuation).

Method Signature: `public static boolean isPalindrome(String str)`

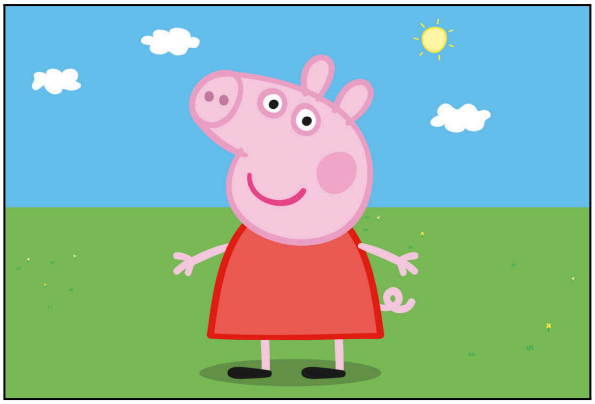


Question 3 - Binary.java: To convert from a base-10 integer to its base-2 (binary) equivalent, the number is divided by two, and the remainder is the right-most binary digit. The integer result is again divided by two, its remainder is the next digit. This process repeats until the quotient becomes zero Your textbook discusses this algorithm in section 8.6 (but don't worry about doing it recursively). And, here is a video that illustrates the algorithm.

Write a method that reads a positive integer from the terminal, and prints out the binary equivalent using the above algorithm. You can use String and the concatenation operator to store the remainders as they are produced in the correct order.

Method Signature: `public static String convertToBinary (int num)`

Question 4 - PigLatin.java: Write a method that reads a sentence String from the terminal and translates it into Pig Latin. In Pig Latin, any word longer than 2 letters is encoded by putting the first letter at the back and adding "ay".



For example:

car ➔ arcay

starcraft ➔ tarcraftsay

I love starcraft ➔ I ovelay tarcraftsay

You can assume the sentence String will only contain letters and spaces.

Hint: Start by thinking about how to translate a single word into pig latin, and create a method to do that. Then, make that a separate method that takes a String as a parameter and loops through the message String, converting one word at a time using the previous method you built. As it loops through the message String, it creates a new String with the translated words and returns it at the end of the method.

Method Signature: `public static String pigLatin (String msg)`