Chris Piech Section #4

CS 106A May 2, 2018

Solution to Section #4

Portions of this handout by Eric Roberts, Patrick Young, Jeremy Keeshin, and Julia Daniel

**Warmup:**

1. True
2. False; strings are immutable. So **charAt()** can’t be used to reassign characters in a string – it can only be used to retrieve the character in a string at a specific index.
3. False; this approach (building a new string out of substrings and then reassigning it to the old string’s variable) is *almost* correct, but there’s an off-by-one error in the bounds on the first substring, so it actually prints **“CS10B rocks my socks!”**.

1. Adding commas to numeric strings

|  |  |  |
| --- | --- | --- |
|  | private String addCommasToNumericString(String digits) {  String result = "";  int len = digits.length();  int nDigits = 0;  for (int i = len - 1; i >= 0; i--) {  result = digits.charAt(i) + result;  nDigits++;  if (((nDigits % 3) == 0) && (i > 0)) {  result = "," + result;  }  }  return result;  } |  |

2. Deleting characters from a string

|  |  |  |
| --- | --- | --- |
|  | private String removeAllOccurrences(String str, char ch) {  String result = "";  for (int i = 0; i < str.length(); i++) {  if (str.charAt(i) != ch) {  result += str.charAt(i);  }  }  return result;  } |  |

A slightly different approach that involves a while loop instead of a for loop:

|  |  |  |
| --- | --- | --- |
|  | private String removeAllOccurrences(String str, char ch) {  while (true) {  int pos = str.indexOf(ch);  if (pos >= 0) {  str = str.substring(0, pos) + str.substring(pos + 1);  } else {  break;  }  }  return str;  } |  |

**3. Separating Digits and Letters**

|  |  |  |
| --- | --- | --- |
|  | private String separateDigitsAndLetters(String str) {  String numbers = "";  String letters = "";  for(int i = 0; i < str.length(); i++) {  char ch = str.charAt(i);  if (Character.isLetter(ch)) {  letters += ch;  } else if (Character.isDigit(ch)) {  numbers += ch;  }  }  return numbers + letters;  } |  |

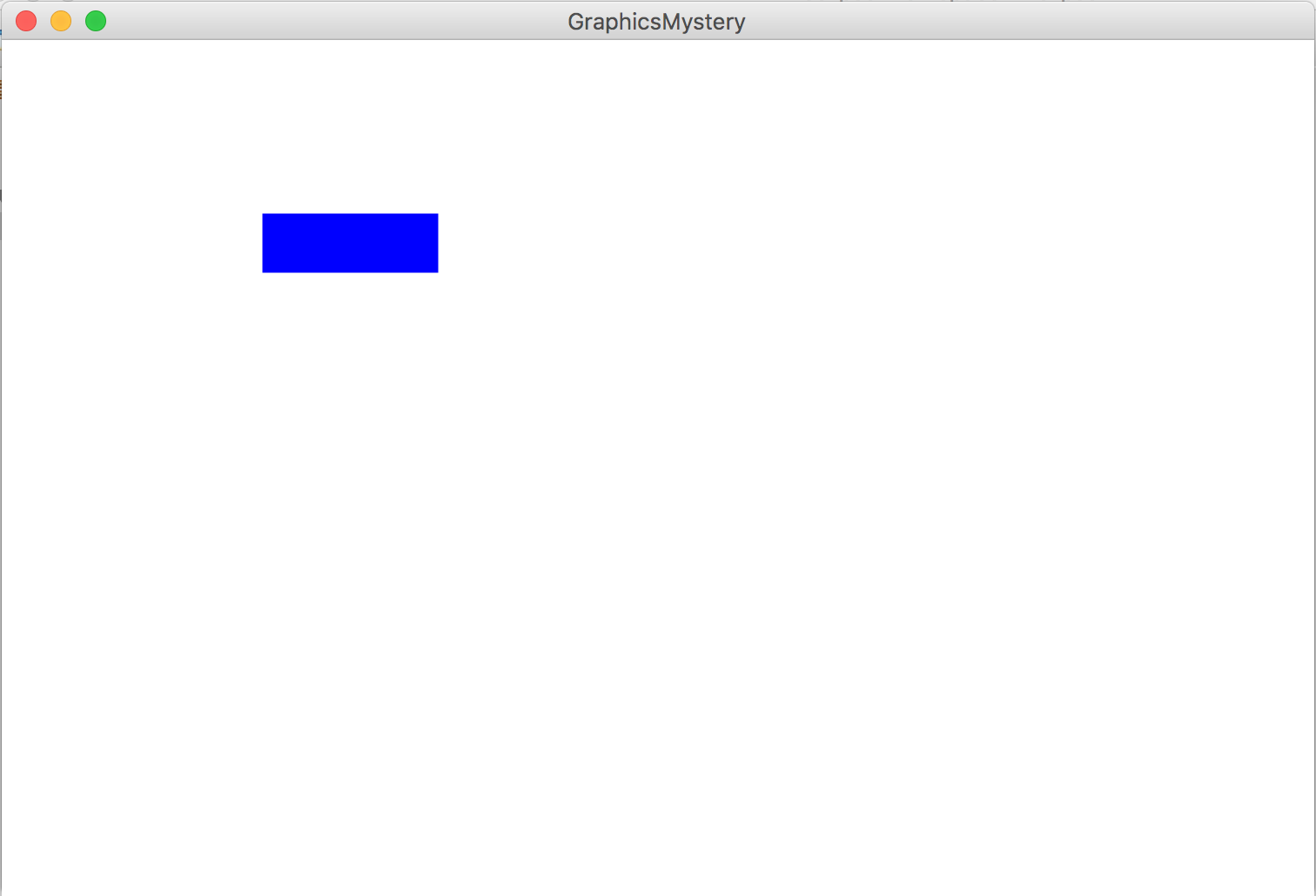
4. Pig Latin

|  |  |  |
| --- | --- | --- |
|  | private String pigLatin(String word) {  if (word.length() == 0) {  return "";  }    // Words starting with vowels  if (isVowel(word.charAt(0))) {  return word + "yay";  }    // Words starting with consonants  int firstVowelIndex = 1;  for (int i = 1; i < word.length(); i++) {  if (!isVowel(word.charAt(i))) {  firstVowelIndex++;  } else {  break;  }  }    return word.substring(firstVowelIndex) +  word.substring(0, firstVowelIndex) + "ay";  }  /\* This is a helper method that returns true if ch is a vowel,  \* and false otherwise.  \*/  private boolean isVowel(char ch) {  return ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o'  || ch == 'u';  } |  |

A slightly different approach that involves a while loop instead of a for loop:

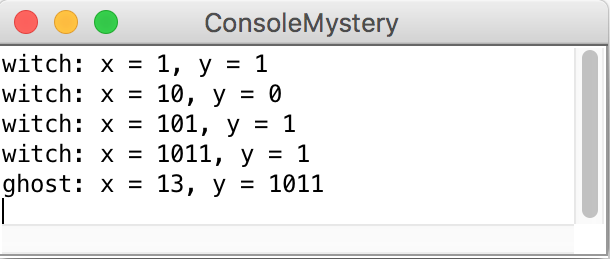
|  |  |  |
| --- | --- | --- |
|  | private String pigLatin(String word) {  if (word.length() == 0) {  return "";  }    // Words starting with vowels  if (isVowel(word.charAt(0))) {  return word + "yay";  }    /\* Word starting with consonants:  \* increment firstVowelIndex while we have not gotten  \* to the end of the string, and have not seen a vowel.  \*/  int firstVowelIndex = 1;  while (firstVowelIndex < word.length() &&  !isVowel(word.charAt(firstVowelIndex))) {  firstVowelIndex++;  }    return word.substring(firstVowelIndex) +  word.substring(0, firstVowelIndex) + "ay";  }  /\* This is a helper method that returns true if ch is a vowel,  \* and false otherwise.  \*/  private boolean isVowel(char ch) {  return ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o'  || ch == 'u';  } |  |

5. Tracing method execution - Graphics



There is one GRect filled blue, with x and y coordinates of (150, 100), a width of 100 and a height of 33.33333… .

6. Tracing method execution - Console



**Style Focus for Section 4**

**Common Programming Idioms:** A programming *idiom* is a commonly used expression or pattern, like using ++ to increment a variable, or the loop-and-a-half. In this section we went over a common pattern of iterating through a string and building up a new result string. It is good to familiarize yourself with common programming idioms because you will see them appear in others’ code, and it will make your own code better.