Solution to Section #4

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

```java
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

```java
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:

```java
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';
}
```
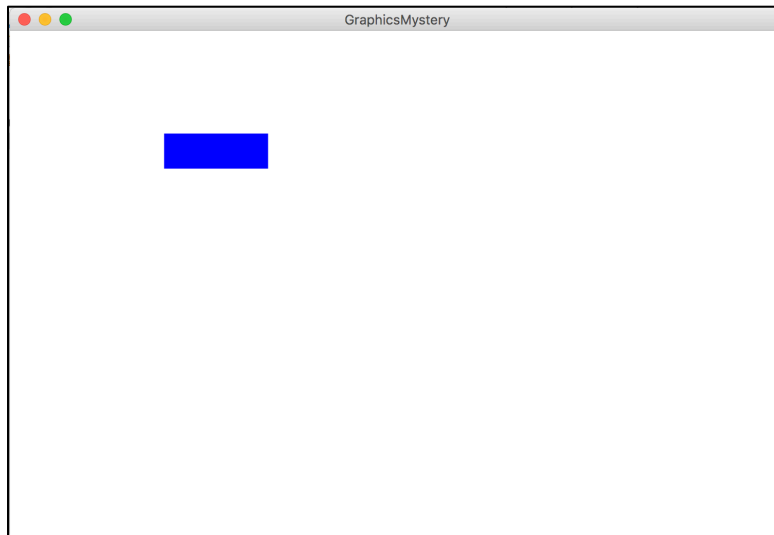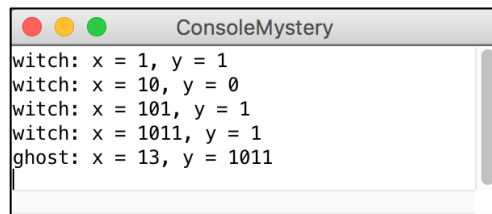
5. Tracing method execution - Graphics

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

7. Working with references

The error with this program stems from the fact that we initialize the `GOval` inside our `setUpOval` method rather than in the run method. In Java, when we pass objects as parameters, we're passing *copies of references to those objects*. As a result, when we initialize the `GOval` inside our `setUpOval` method, we're only modifying the oval reference in that method. Thus, when the method returns and we try to `add(o, 100, 100)` in our run method, `o` is still null and so we get a `NullPointerException`. We can fix this problem either by initializing the oval in the run method before passing it in or by returning the oval from `setUpOval` and storing that return value in `o`.

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.