CS 249: Assignment 03

Programming Assignments (95%)

GreetingCard.java

Create a java file with a public class GreetingCard. The purpose of this class is to generate a greeting card (each of which hold at most 5 lines of text). The card is filled in with a boundary character, and all text is centered. Any non-empty line of also has a space on either side. Cards have a maximum width of **50 characters**. It will have the following public methods (note that some of these are NOT directly used Hallmark):

- **public GreetingCard(String [] lines, char boundaryChar)**
  - Stores the lines and boundary character
  - WARNING: when storing the lines, remember to:
    - Reallocate the instance variable
    - Copy the individual array values
  - **Do NOT just do:** this.lines = lines
    - I would suggest calling setLines() here
- **public char getBoundaryChar()**
  - Returns the boundary character
- **public String getLines()**
  - Returns a SINGLE String that concatenates the lines, with a newline "\n" at the end of each line.
- **public void setBoundaryChar(char boundaryChar)**
  - Stores the boundary character
- **public void setLines(String [] lines)**
  - Stores the lines, BUT remember to:
    - Reallocate the instance variable
    - Copy the individual array values
  - **Do NOT just do:** this.lines = lines
- **public String generateBoundaryLine()**
  - Returns a String with 50 boundary characters followed by a newline
- **public String generateCenteredLine(String text)**
  - Returns a String with a CENTERED line of text flanked by a space on either side (if non-empty) and boundary characters, ending in a newline.
  - You can assume that the text will never be too large to fit (i.e., never over 46 characters).

    - To do this:
        - If text has length greater than zero: text = " " + text + " " (i.e., put a space on either side).
        - Start with an empty String (or you can use StringBuilder)
        - Compute how much total padding will be needed:
          50 – (length of text after appending spaces)
        - Get half of the total number of padding **using integer division**
        - Get the second half of padding by: (total padding) – (first half of padding)
        - Append the first half of padding (boundary character)
        - Append the text (with the spaces)
        - Append the second half of padding
        - Append a newline
        - Return the String
- **public String toString()**
    - This **returns a String** with a set of greeting cards.
    - **NOTE: This function does NOT print anything out! In other words, DON'T use System.out.println here!!!**
    - Each card will have the following dimensions:
        - 50 characters in width
        - 9 lines in height
    - Start with an empty String (or you can use String builder)
    - For every 5 lines:
        - If we already have text, append a newline (to put a separator between cards)
        - Append two boundary lines
        - Compute how many lines are left at this point
        - If the number of lines left are less than 5:
            - Line count will be however many lines are left
            - Extra line count will be 5 – (lines left)
        - Otherwise:
            - Line count will be 5
            - Extra line count will be zero
        - Append (line count) number of lines from your array of lines
        - Append (extra line count) number of boundary lines
        - Append two boundary lines regardless
    - Return the single String containing all of these greeting cards

Hallmark.java

The purpose of this program is to ask the user for information for their GreetingCard, and then print out the final set of cards. Create a class Hallmark, and add these methods (both are public and static):

- **public static GreetingCard generateCard(Scanner input)**
    - **WARNING: Scanner input has ALREADY been created! Do NOT create a new one here!**
    - Print **"Enter boundary character:"** using System.out.println().
    - Get the boundary character as the first character of the nextLine() from the Scanner object.
        - NOTE: You may assume the user will NOT enter an empty line.
    - Print **" Enter number of lines:"** using System.out.println().
    - Get the number of lines by:
        - Reading in the next String LINE using nextLine()
        - Using Integer.parseInt() to convert this line to an integer
    - Create a String array with the appropriate number of Strings (*allLines*).
    - Print **"Enter lines:"** using System.out.println().
    - In a loop, read in the correct number of lines from the user using nextLine() on the Scanner object and store each line in the String array *allLines*.
    - Create a new GreetingCard object using *allLines*, and boundary character.
    - Return the newly-created GreetingCard object.
- **public static void main(String [] args)**
    - Create a Scanner object to read from System.in.
    - Create a GreetingCard object using generateCard(), remembering to grab what the method returns and putting it into a variable *n*.
    - Print **"For any occasion:"** using System.out.println().
    - Print out the cards using: System.out.println(n)

Example Runs (user input highlighted in blue):

```
Enter boundary character:
@
Enter number of lines:
10
Enter lines:
A Poem

Written in memory
of dear Ragamuffin.

There once was a dog named Ragamuffin.
Whose days were spent seeking a MacGuffin.
He lost it a lot,
to further the plot,
And found that the item was good-for-nothing.
For any occasion:
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@ A Poem @@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@ Written in memory @@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@ of dear Ragamuffin. @@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@ There once was a dog named Ragamuffin. @@@@@
@@@ Whose days were spent seeking a MacGuffin. @@@
@@@@@@@@@@@@@@@ He lost it a lot, @@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@ to further the plot, @@@@@@@@@@@@@@
@ And found that the item was good-for-nothing. @@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
```

Testing Screenshot (5%)

Submit a screenshot showing the results of running the test program(s).

Grading

Your OVERALL assignment grade is weighted as follows:

- 5% - Testing results screenshot
- 95% - Programming assignments

For the **PROGRAMMING** portion of the assignment, in addition to the usual penalties:

Issue	***Penalty (in %)***
GreetingCard.java missing / not implemented	70
Hallmark.java missing / not implemented	30
GreetingCard.java not properly implemented	35
Hallmark.java not properly implemented	15