CS234 Computer Science II
Lab 4
Total points: 100

Read the instructions carefully.

The objective of this assignment is to practice the use of methods to create modularity in your code. Therefore, in this lab you will create a **Java program** that presents a menu with submenus for different tasks.

```
/**
    * This method gets the user input for the string
    * and calls countA()
*/
public static void reg_option1()


/**
    * The method takes a String and returns the number of lower-case letters 'a'
    * in the string. Do not use the count() built-in method.
    * @param s, the string
    * @return the count
*/
public static int countA(String s)


/**
    * This method gets the input string, the target character
    * and the new character, then it calls replace() and
    * prints the new string
*/
public static void reg_option2()



/**
    * The method takes a string and two characters and returns a string in which
    * every occurrence of the first character is replaced by the second one.
    * Do not use the .replace() built-in method.
    * @param s, the string
    * @param a, the first character
    * @param b, the second character
    * @return a new string
*/
public static String replace(String s, char a, char b)



`/**
    * This method gets input for the String and the target character
    * Then, it calls the recursive method containsCharacter() and
    * it prints the result (true or false)
*/
public static void rec_option1()

/**
    * Recursive method
    * Returns True or False if the target character exists in the string
    * @param str (string)
    * @param target (char)
    * @return true or false (boolean)
*/
public static boolean containsCharacter(String str, char target)
```

Your program's **main()** method should have **just one line** to call **Menu().**

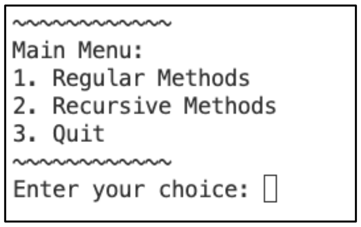
```
/**
    * Method to present the initial Menu with 3 options:
    * 1. Regular Methods, 2. Recursive Methods, 3. Quit
    * The menu uses a Switch statement.
    * For case 1, it calls the method MenuRegular()
    * For case 2, it calls the method MenuRecursive()
    * For case 3, it exits
    * (see the examples)
*/
public static void Menu()

/**
    * Method to present the options for the Regular Methods
    * 1. Count number of As, 2. Replace target character,
    * 3. Return Main
    * The method uses a switch statement to call the corresponding
    * method.
    * If the user selects option 3, the general Menu should be displayed
    * (see the examples)
*/
public static void MenuRegular()
```

```
/**
    * Method to present the options for the Recursive Methods
    * 1. Search Character, 2. Return Main
    * The method uses a switch statement to call the corresponding
    * method.
    * If the user selects option 2, the general Menu should be displayed
    * (see the examples)
    */
   public static void MenuRecursive()
```

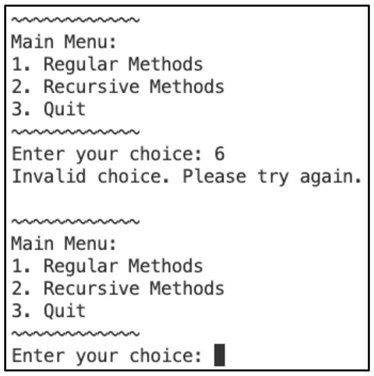
For this lab, you don’t need to worry about the most efficient data type. However, remember that in real world applications this is an important consideration.
No additional methods are allowed. You don’t need them.
Remember that for a recursive method, the method calls itself. It does not use a loop to repeat the tasks.

The main menu must present **three** options as shown in the following figure.



```
~~~~~~~~~~~~~~
Main Menu:
1. Regular Methods
2. Recursive Methods
3. Quit
~~~~~~~~~~~~~~
Enter your choice: ▯
```

The user can select any of those options. The program **must end only** when the user inputs the number 3 (i.e., “**3. Quit**”).
If the user selects an option not listed, then the program must let the user know that it was an **invalid option**, and the menu should be **displayed again.** For example:

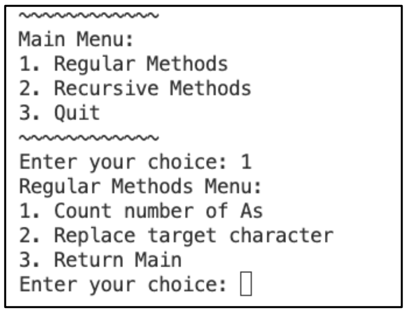


```
~~~~~~~~~~~~
Main Menu:
1. Regular Methods
2. Recursive Methods
3. Quit
~~~~~~~~~~~~
Enter your choice: 6
Invalid choice. Please try again.

~~~~~~~~~~~~
Main Menu:
1. Regular Methods
2. Recursive Methods
3. Quit
~~~~~~~~~~~~
Enter your choice: █
```

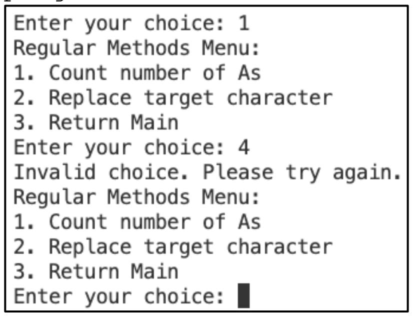
When the user selects a correct option then the menu should **call** the **corresponding method** for that option.
After getting the results for each option the menu should be displayed again.

Main Menu Option 1
This option shows the "Regular Methods Menu:



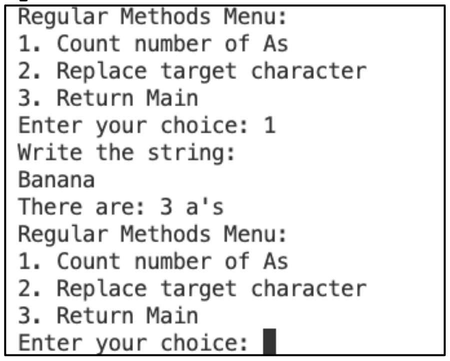
```
~~~~~~~~~~~~
Main Menu:
1. Regular Methods
2. Recursive Methods
3. Quit
~~~~~~~~~~~~
Enter your choice: 1
Regular Methods Menu:
1. Count number of As
2. Replace target character
3. Return Main
Enter your choice: ▯
```

The menu shows three options. If the user selects an incorrect option, the program should send a message and show the menu again

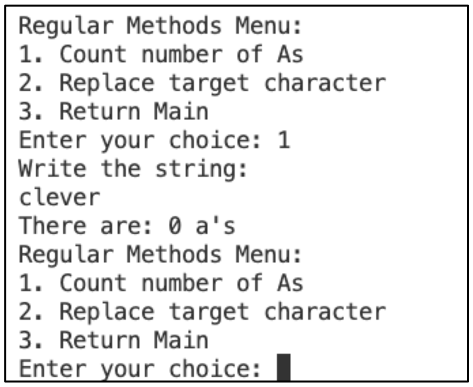


```
Enter your choice: 1
Regular Methods Menu:
1. Count number of As
2. Replace target character
3. Return Main
Enter your choice: 4
Invalid choice. Please try again.
Regular Methods Menu:
1. Count number of As
2. Replace target character
3. Return Main
Enter your choice: █
```

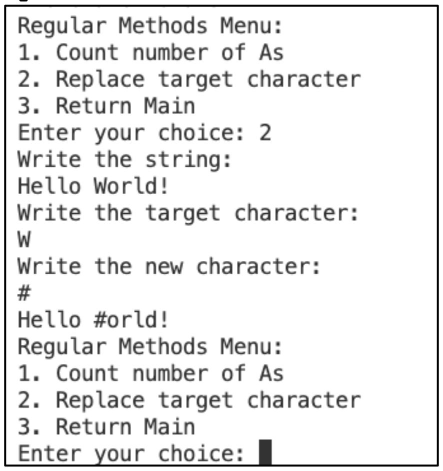
Option 1.



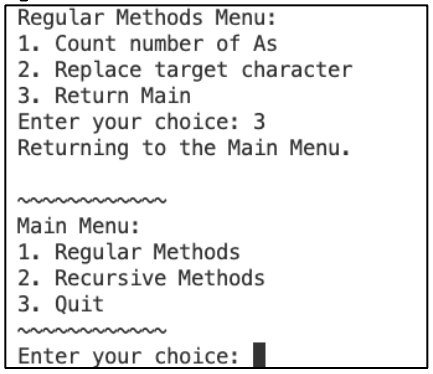
```
Regular Methods Menu:
1. Count number of As
2. Replace target character
3. Return Main
Enter your choice: 1
Write the string:
Banana
There are: 3 a's
Regular Methods Menu:
1. Count number of As
2. Replace target character
3. Return Main
Enter your choice: █
```


```
Regular Methods Menu:
1. Count number of As
2. Replace target character
3. Return Main
Enter your choice: 1
Write the string:
clever
There are: 0 a's
Regular Methods Menu:
1. Count number of As
2. Replace target character
3. Return Main
Enter your choice: 
```

Option 2


```
Regular Methods Menu:
1. Count number of As
2. Replace target character
3. Return Main
Enter your choice: 2
Write the string:
Hello World!
Write the target character:
W
Write the new character:
#
Hello #orld!
Regular Methods Menu:
1. Count number of As
2. Replace target character
3. Return Main
Enter your choice: 
```

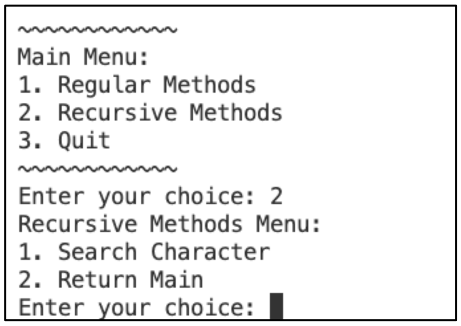
Option 3


```
Regular Methods Menu:
1. Count number of As
2. Replace target character
3. Return Main
Enter your choice: 3
Returning to the Main Menu.

~~~~~~~~~~~~
Main Menu:
1. Regular Methods
2. Recursive Methods
3. Quit
~~~~~~~~~~~~
Enter your choice: 
```

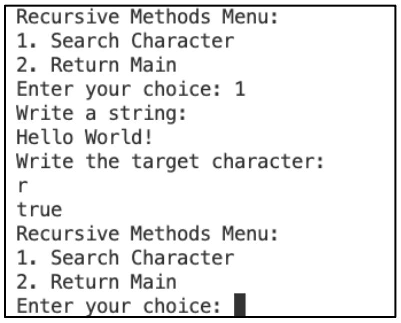
Main Menu Option 2

This option shows the “Recursive Methods Menu:

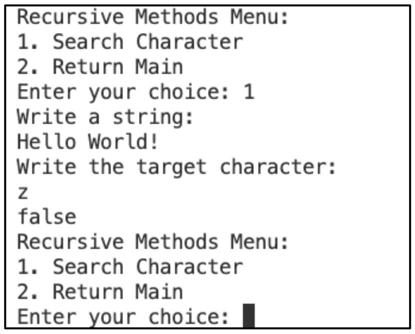


```
~~~~~~~~~~~~
Main Menu:
1. Regular Methods
2. Recursive Methods
3. Quit
~~~~~~~~~~~~
Enter your choice: 2
Recursive Methods Menu:
1. Search Character
2. Return Main
Enter your choice: █
```

Option 1

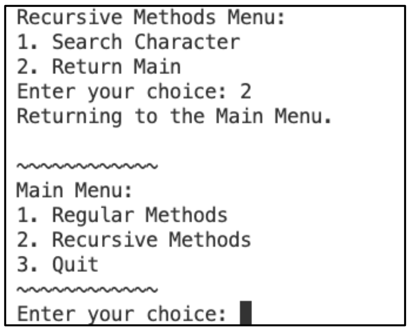


```
Recursive Methods Menu:
1. Search Character
2. Return Main
Enter your choice: 1
Write a string:
Hello World!
Write the target character:
r
true
Recursive Methods Menu:
1. Search Character
2. Return Main
Enter your choice: █
```



```
Recursive Methods Menu:
1. Search Character
2. Return Main
Enter your choice: 1
Write a string:
Hello World!
Write the target character:
z
false
Recursive Methods Menu:
1. Search Character
2. Return Main
Enter your choice: █
```

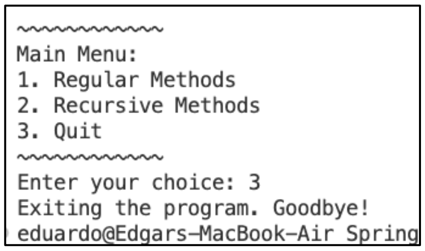
Option 2



```
Recursive Methods Menu:
1. Search Character
2. Return Main
Enter your choice: 2
Returning to the Main Menu.

~~~~~~~~~~~~
Main Menu:
1. Regular Methods
2. Recursive Methods
3. Quit
~~~~~~~~~~~~
Enter your choice: █
```

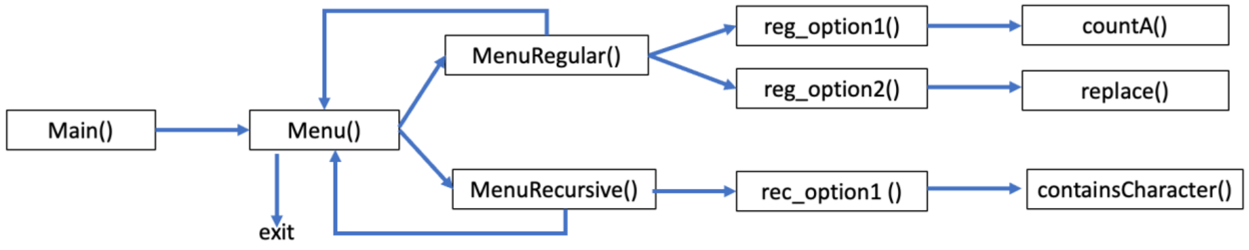
Main Menu Option 3



```
~~~~~~~~~~~~~~
Main Menu:
1. Regular Methods
2. Recursive Methods
3. Quit
~~~~~~~~~~~~~~
Enter your choice: 3
Exiting the program. Goodbye!
eduardo@Edgars-MacBook-Air Spring
```

If your program terminates after each option, then it is not correct!

The following diagram shows the relationships between the methods.



This modularity can help you to create more complex menus. Each method just solve a given task. You will need something similar for your project!

Submission details:

Upload a **single ZIP** file.
Name your file as follows: **Lab4_Lastname_Firstname.zip**

Your **.zip** file must contain the following:

1. Your **.java** source file (no .class).
2. A .txt file (readme.txt) with simple instruction on how to **compile** and **execute** your programs (I reviewed in class how to compile your .java files)
3. A **SINGLE PDF** with **screenshots** from your program running. Show the use of the different menu options. Do not send .jpg files.

In each .java file, write as a *multiline* comment at the beginning of the file the following: Your name

The **zip** file must be uploaded to Canvas. I do not accept answers via email or as comments.
I do not accept image files; it must be a PDF file with the screenshots.

The **zip** file must be **uploaded to Canvas**. I **do not accept answers via email.**
I do not accept answers in the comments section.
I do not accept image files; it must be a PDF file.

Make sure to check the **due date** for this activity on Canvas. Try to submit it before the due date so you can have time to check for improvements.
Make sure you are **submitting the correct files**. I will grade the files uploaded to Canvas.

Use the ***javac* and *java*** commands **before** submitting your solution to test if they work outside any IDE.

Make sure to review the **grading rubric.**

Read all the instructions carefully.

Late submissions are not allowed.

If you have questions, contact me before making assumptions about what you need to do for solving this Lab.

Do not assume requirements, contact me if you have questions!

Ensure that your code is original and developed by you.