Assignment 1: Phone Number Encrypter

CSE 174 – Fall 2021

Assignment 1: 50 points – Due Sunday, Sep 12, 2021 by 11:59pm Ohio time

Outcome:

Students should demonstrate an ability to:

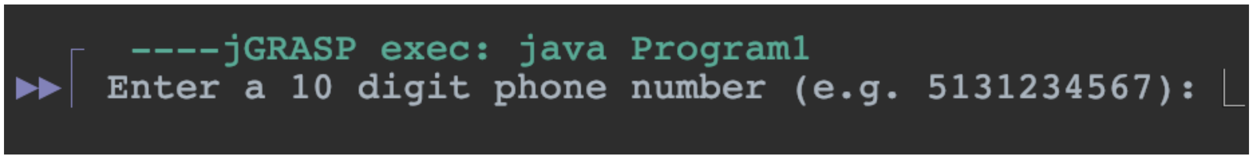
- Write a Java program that solves a problem involving mathematical operations.
- Write a Java program that displays information to the console according to a given set of formatting guidelines.
- Write Java source code that follows a given set of style guidelines.

Background:

A young man who started an internship in a small company asked his manager to give him a small task to do.
“We have lots of phone numbers that we want to share with other companies but we need to encrypt them first”, the manager immediately responded. “I want you to write a simple program that gets a phone number from the user and prints the encrypted version of the number”, he continued, “since this is your first assignment, I’ll send you instructions with details and you just follow them”.

Getting Started

1. Download Program1.java file.
2. Open the file in jGRASP.
3. Add your name and your section at the top of the code.
4. Do not change anything from line 4 through 20.
5. This code asks the user to enter a 10 digit phone number, and saves the given number inside the **PHONE_NUM** constant variable.
6. Compile and run your code, and you should see the following output:



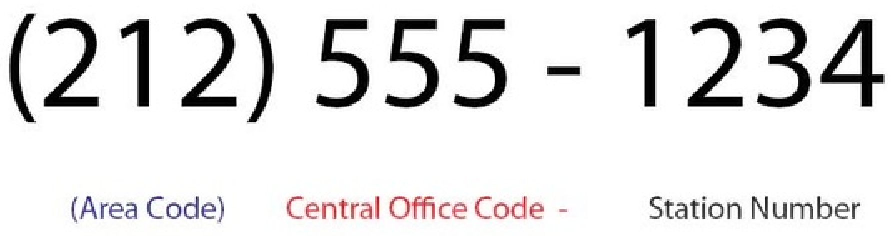
```
 ----jGRASP exec: java Program1
Enter a 10 digit phone number (e.g. 5131234567): 
```

7. You can assume the user always enters a correct 10 digit number.
8. Now, just read the following steps and implement a code for each step one by one.

Instructions Continued on Next Page!!!

Step 1: Separating Digits

Since the given number is a single 10 digit number, the first thing that needs to be done is to separate digits into smaller numbers as following:

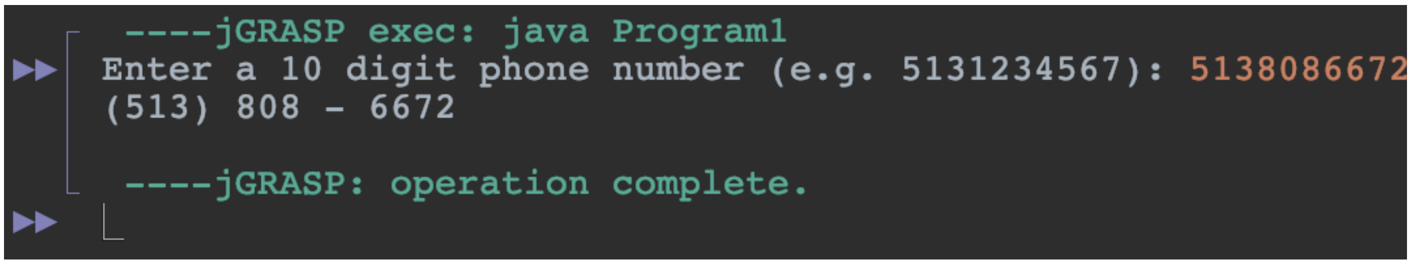


Divide the given number into 3 parts:

1. Area Code: the first 3 digits from the left
2. Central Office Code: the next 3 digits from the left
3. Station Number: the last 4 digits from the right

- Separate digits by only using math operators: / %
- For separating digits, **do not** use other classes such as: String, Integer, or Math class.
- Everytime you separate each part, save it inside an **int** variable. **Only and only int variables should be used for holding each part of the number.**
- Use meaningful variable names for holding different parts of the number.

Print your results on the display to make sure you did this step correctly. In your print statement, separate different parts by using (), space, and - as follows, just to make your output more appealing:



```
 ----jGRASP exec: java Program1
Enter a 10 digit phone number (e.g. 5131234567): 5138086672
(513) 808 - 6672

 ----jGRASP: operation complete.
```

Steps Continued on Next Page!!!

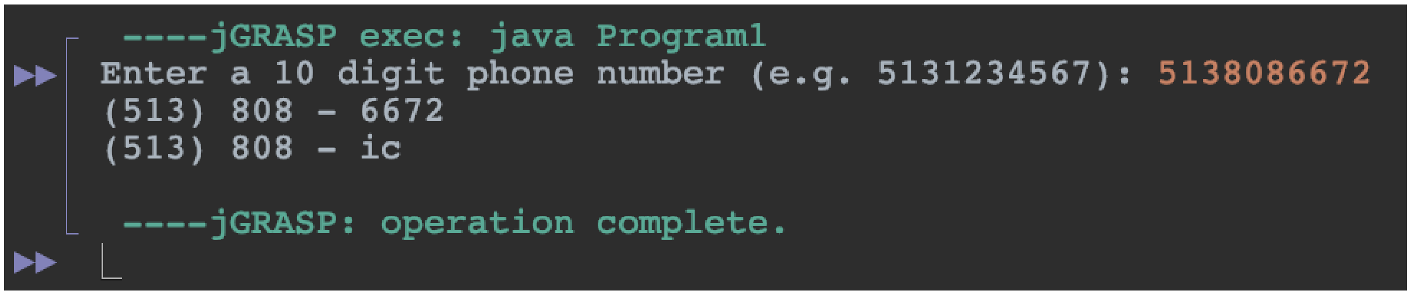
Step 2: Encrypting Station Number

In this step you are to convert the last 4 digits of the phone number into 2 characters as follows:

1. Divide the number into two 2 digit numbers and save each one inside a variable e.g. 6672 should be separated into two numbers 66 and 72 and saved into 2 different variables.
2. Add 33 to each number and save the result back into the variable.
3. Convert each number into a **char** type value and save the result inside a **char variable**.

- Separate digits by only using math operators: / %
- For converting a number into a char type value, **do not** use other classes such as: String, Integer, or Character class. You can easily do it by explicitly casting each number to a char type.
- The result must be saved inside two **char type variables**.
- Reverse the order of the characters when you are printing them as shown below.

Print the result to make sure your code is generating the correct characters. You output should look like the following:



```
 ----jGRASP exec: java Program1
Enter a 10 digit phone number (e.g. 5131234567): 5138086672
(513) 808 - 6672
(513) 808 - ic

 ----jGRASP: operation complete.
```

Steps Continued on Next Page!!!

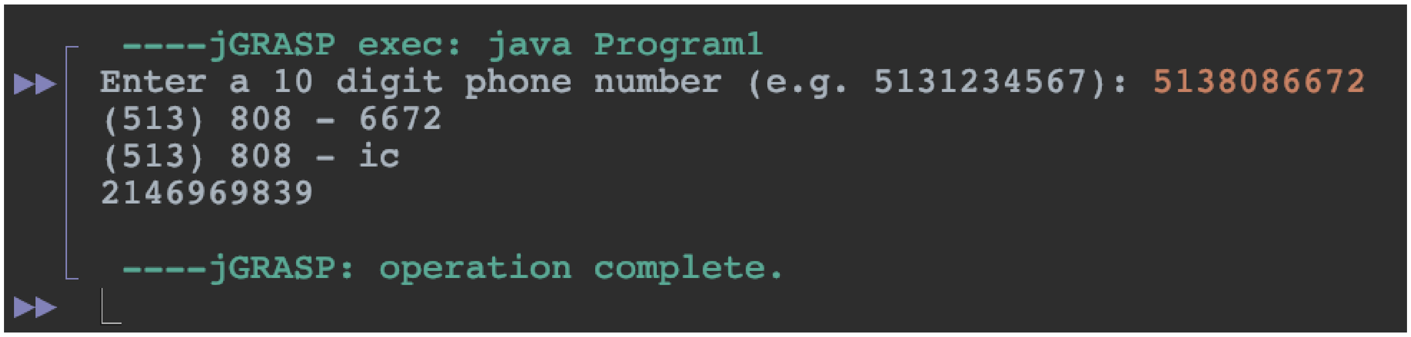
Step 3: Encrypting the Rest of the Digits

At this point you are left with 6 digits of the phone number which you saved inside 2 different variables in step1.

1. Combine those 2 numbers and create a new **int** number.
2. Save the result inside a new **int variable**.
3. Now, subtract the result from the maximum int value in java. Use **Integer.MAX_VALUE** to get the maximum possible int value in java.
4. Save the result of subtraction back inside the variable.

- Combine 2 numbers and create a new number only by using math operators: + *
- For combining two numbers, **do not** use other classes such as: Integer, Math, etc.
- The result must be saved inside an **int variable**.

Print the result to make sure your code is generating the correct numbers. Your output should look like the following:



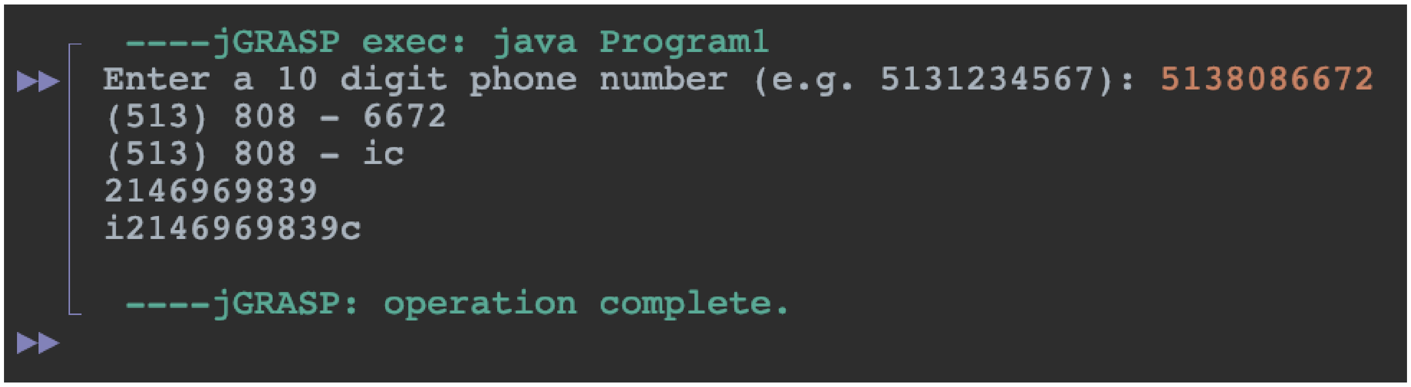
```
 ----jGRASP exec: java Program1
Enter a 10 digit phone number (e.g. 5131234567): 5138086672
(513) 808 - 6672
(513) 808 - ic
2146969839

 ----jGRASP: operation complete.
```

Steps Continued on Next Page!!!

Step 4: Rearranging the results

Combine the two characters from step2 with the generated number from the previous step in the following format: **first character + number from step 3 + second character (characters are reversed, remember!)**

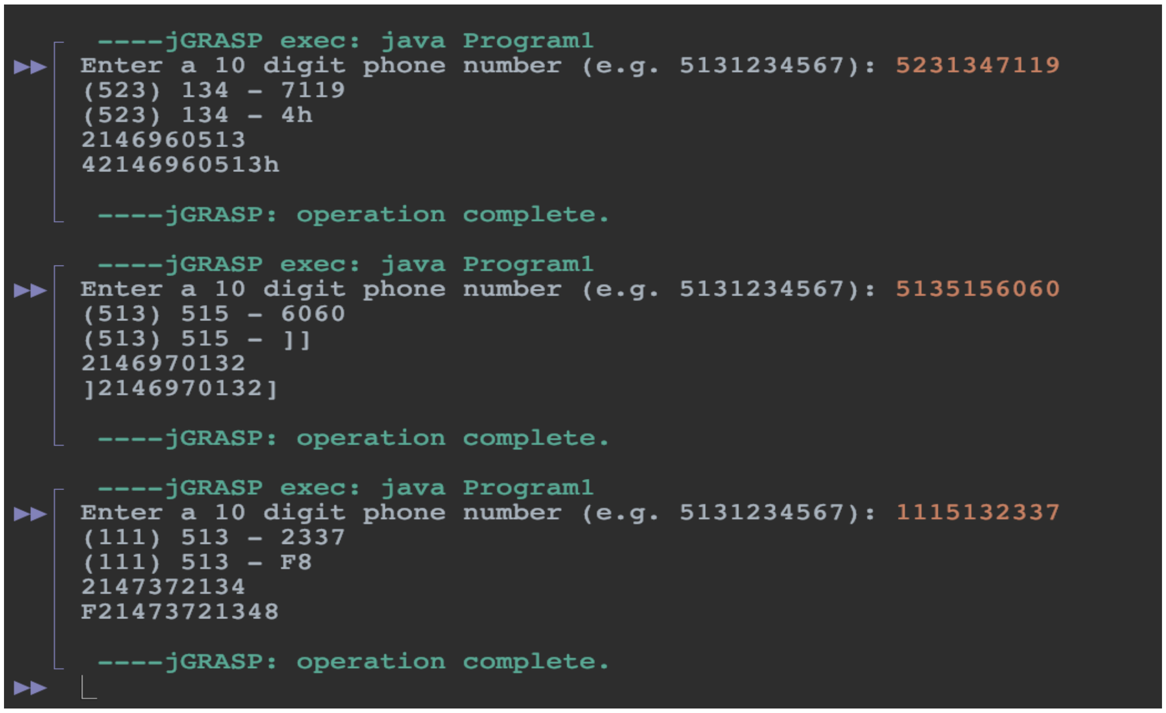
So your final result should look like the following:



```
 ----jGRASP exec: java Program1
Enter a 10 digit phone number (e.g. 5131234567): 5138086672
(513) 808 - 6672
(513) 808 - ic
2146969839
i2146969839c

 ----jGRASP: operation complete.
```

Sample Runs:

Test your code by entering the same inputs as the following sample runs. Your code should generate the same outputs.



```
 ----jGRASP exec: java Program1
Enter a 10 digit phone number (e.g. 5131234567): 5231347119
(523) 134 - 7119
(523) 134 - 4h
2146960513
42146960513h

 ----jGRASP: operation complete.

 ----jGRASP exec: java Program1
Enter a 10 digit phone number (e.g. 5131234567): 5135156060
(513) 515 - 6060
(513) 515 - ]]
2146970132
]2146970132]

 ----jGRASP: operation complete.

 ----jGRASP exec: java Program1
Enter a 10 digit phone number (e.g. 5131234567): 1115132337
(111) 513 - 2337
(111) 513 - F8
2147372134
F21473721348

 ----jGRASP: operation complete.
```

Tips on Next Page! You really want to read those at least!

Tips:

START EARLY!!! DO NOT WAIT UNTIL THE LAST MINUTE!!!! WE MADE THIS TIP AS ANNOYING AS POSSIBLE TO LOOK AT FOR A REASON!!!

- Always use **meaningful** variable **names**. You should avoid using names like ‘a’, ‘aa’, ‘ab’.
- Always start your code with **comments** at the top with at least your **name, your section and some description of the code**. Also add **comments inside** your **code** as well. The comments inside your code should explain what is happening on specific lines to help the reader understand your code.
- Finish your program step by step. Run the result of each step as shown to make sure you are on the right track and **getting the right output**. **Do not wait until the last step to test your code!**
- You can not use other classes for this assignment. Only simple math operators such as: + * / % are allowed to be used.

Scoring:

	Full Credit	No Credit or Partial Credit
Code runs as expected (25 points)	Your code runs with no errors and passes all the tests on Code plugins.	You will get 5 points for passing each test.
Proper use of variables (10 points)	You defined an adequate number of variables using meaningful variable names.	You did not use proper variable names, or you used too many variables.
Using only mathematical operators to solve the problem (10 points)	Your code successfully solves the problem by using only math operators.	You use other classes to solve the problem.
Format and comment source code (5 points)	You followed all of the given formatting requirements (indentation, comments at the top and in your code, etc).	You did not follow some or all of the formatting requirements as specified in the requirement.

Steps Continued on Next Page!!!

Your submission will be checked for potential academic dishonesty violation.

Submit your code on Canvas:

If your code is generating the same results as the sample runs, now you are ready to submit your code on canvas. If there are any style errors, fix them and resubmit your code.