# COSC 1436: Programming Fundamentals I

# (Assignment 3)

* **DUE Time and Date**: **Midnight, Friday, 10/26/2018**
* **What to submit:** **LastNameFirstNameTranslator.java**

**LastNameFirstNameESPGamejava**

**LastNameFirstNameLetsPlay.java**

**Note 1:** Unless otherwise mentioned, you are asked to upload ONLY your java source files through blackboard. Email submission is not accepted, because of confusion in grading.

**Note 2:** If your programs contain any syntactical errors, no points will be given. Thus, please make sure your programs are properly compiled with computers at the CS labs, not only in your laptop or desktop environments.

**Note 3:** No late submission will be accepted, thus keep the deadline.

**Note 4:** Grading will be divided into two categories, formatting and logic, where formatting compromise 25% of your total grade. Formatting will be based on the following rules.

**Rule 1:** Naming is an important issue in Java. Not only you should you define meaningful variable names, but should also give appropriate names for the physical java file, which must be the same as the public class name that you edit.

Unless otherwise mentioned, you will follow **the industry standard for Java naming convention**:

(1) Java Classes start in uppercase and each individual word in the class name is capitalized;

(2) All Java methods and variables start in lowercase and each individual word in the method and variable is capitalized;

(3) Each final variable (known as a constant) should be written in all uppercase.

**Rule 2:** There should be a space around all operators (e.g., 3 + 5, not 3+5). In addition, spacing with regards to parentheses should be consistent.

**Rule 3:** In addition to the Java naming conventions, you are asked to add your name in front of each class name like **LastNameFirstNameClassName.java.**

For instance, if your name is “John Doe” and the class name is “RightTriangle”, then the class name in your source code should be “DoeJohnRightTriangle” and your corresponding physical file name should be “DoeJohnRightTriangle.java”.

**Rule 4:** Everything nested inside of an open brace should be indented with regular-sized spaces (say, 4 or 8 spaces). The open brace for functions and classes should (1) come at the end of the line and be preceded by a space like

**public class DoeJohnRightTriangle {**

**public static void main( String args[] ) {**

**}**

**}**

or (2)start with the new line as shown below:

**public class DoeJohnRightTriangle**

**{**

**public static void main( String args[] )**

**{**

**}**

**}**

**Rule 5:** Always type block Javadoc comments to include title of the project, program’s purpose, your name, the date, and the version number as in the lectures or in the labs. For example,

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**@Title: LastNameFirstNameClassName**

**@Purpose: To verify the edit, compile, execute function in Textpad**

**@Author: (your last & first name)**

**@Date: (today’s date)**

**@Version: 1.0**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

## Question 1 (50 points): Translator

Write a program that

1. reads the user input that is an alphabetic phone number
2. translates each of the alphabetic phone numbers to the equivalent numeric phone number, and
3. displays the translated numeric numbers

For example, when the program reads 555-GET-FOOD, it should print 555-438-3663

Note: A template, as a .java file, is provided for you to get a better understanding of the code’s layout.

On a standard telephone, the alphabetic letters are mapped to numbers in the following fashion;

A, B and C = 2

D, E, and F = 3

G, H, and I = 4

J, K, and L = 5

M, N, and O = 6

P, Q, R, and S = 7

T, U, and V = 8

W, X, Y, and Z = 9

You need to break your code into the following methods:

* **getInput**: It asks the user to enter an alphabetic phone number and reads the input as a String from the keyboard. It calls **isValid** method for validity check of the phone number. If **isValid** method returns true, then it calls **getPhoneNumber** method for translation. Otherwise, it prompts the user continuously until a valid input is entered.
* **isValid**: It returns true if the alphabetical phone number:

1. includes only letters, numbers or dashes, and
2. its length is between 10 and 14 (with country code).

Otherwise, it returns false.

* **getPhoneNumber**: It accepts one argument that is the alphabetical phone number passed from **getInput** mehod, translates it to the equivalent numeric phone number, and writes the result on the screen.

**Here are the necessary steps:**

1. **getInput** method needs to
2. read the user’s input
3. call **isValid** method to check if the alphabetical phone number is valid or not. If yes, call **getPhoneNumber** method by passing the phone number, otherwise prompt the the user again.
4. **getPhoneNumber** method needs to
5. accept a String data (an alphabetical phone number) as its argument
6. have a loop to access each character of the String and do the followings until we reach the end of the String:
7. access each character in that String by specifying index number -> charAt(i)
8. check whether the character is only a digit, letter or a special character.  
   \* Hint: import java.lang.Character =>

In order to test if a character is a letter or digit, you should use isLetterOrDigit() method that is provided in Character wrapper class, also Character wrapper class has other methods such as isLetter, isDigit, etc. or remember Unicode that provides a unique number for every character.

1. If it is a letter, convert it into the corresponding digit (see the map for it above).
2. **isValid** method needs to
3. check each alphabetical phone number for validity, and if it is valid, return true, otherwise return false.

* the alphabetical phone number is valid if it includes:

1. only letters, numbers or dashes, and
2. its length (the number of characters in the phone number read) is between 10 or 14 (with country code)

**Hint:** Your program should display the result as shown in below

1-800-FLOWERS = 1-800-3569377

1-800-GOT-JUNK = 1-800-468-5865

555-GET-FOOD = 555-438-3663

1-800-PET-MEDS = 1-800-738-6337

1-800-LAWYERS = 1-800-5299377

1-800-GO-FONTS = 1-800-46-36687

713-333-MOVE = 713-333-6683

Grading criteria include documentation, descriptive variable names, and adherence to the coding convention noted on pages 1 & 2.

Your file will have the following documentation header:

Your file will have the header and the class definition as follows:

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**@Title: LastNamePhoneNumberTranslator**

**@Purpose: To get familiar with input/output classes, String object, and various operations in Java**

**@Author: (your last first name)**

**@Date: (today’s date)**

**@Version: 1.0**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

## Question 2 (50 points): ESP Game

1. Write a program that should have the following name fields and methods to test your ESP (extrasensory perception)

|  |
| --- |
| **ESPGame** |
| **-computer : String**  **-user : String** |
| **+runComputer() : void**  **+ setUserChoice():void**  **+isValid():boolean**  **+whoWins() : String** |

The program should randomly select the name of a color from the following list of words:

**Red, Green, Blue, Orange, Yellow**

**runComputer() method ->** This method generates a random number. For example, if the number is 0, the selected word is Red; if the number is 1, the selected word is Green; and so forth. It assigns the selected word to the field ‘computer’.

**setUserChoice() method ->** This method asks the user to enter a color that the computer has selected and assigns the user choice to the field ‘user’. If the color is not the one that is used for ESP game, the user needs to be prompted continuously until the valid choice is entered.

**isValid() method ->** It accepts the user choice as its parameter and returns true if the choice is ‘red’, ‘green’, ‘blue’ , ‘orange’ or ‘yellow’. Otherwise returns false

**whoWins() method ->** It accepts the user choice and the computer selection as its parameters and returns the winner ( ‘user’ or ‘computer’). Note: Ignore case sensitivity.

Write an application program named ‘LetsPlay’. The program should run the game 10 times and then display the grand winner of the game. **Be sure to modularize the program into methods that perform each major task.**

Grading criteria include documentation, descriptive variable names, and adherence to the coding convention noted on pages 1 & 2.

Your file will have the following documentation header:

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**@Title: LastNameFirstNameESP**

**@Purpose: To get familiar with methods , Random class in Java**

**@Author: (your last first name)**

**@Date: (today’s date)**

**@Version: 1.0**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**@Title: LastNameFirstLetsPlay**

**@Purpose: To get familiar with methods , Random class and objects in Java @Author: (your last first name)**

**@Date: (today’s date)**

**@Version: 1.0**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**