JAVA PROGRAMMING
(340)

REGIONAL – 2018

Production Portion:

Program 1: Communications _____ (345 points)

TOTAL POINTS _____ ***(345 points)***

Graders: Please double check and verify all scores and answer keys!

You will have ninety (90) minutes to complete your work.

Your name and/or school name should *not* appear on work you submit for grading.

1. Create a folder on the flash drive provided using your contestant number as the name of the folder.
2. Copy your entire solution/project into this folder.
3. Submit your entire solution/project so that the graders may open your project to review the source code.
4. Ensure that the files required to run your program are present and will execute on the flash drive provided.

   *Note that the flash drive letter may *not* be the same when the program is graded as it was when you created the program.

   * It is recommended that you use relative paths rather than absolute paths to ensure that the program will run regardless of the flash drive letter.

   The graders will *not* compile or alter your source code to correct for this.
   Submissions that do *not* contain source code will *not* be graded.

Assumptions to make when taking this assessment:

- The input file will contain only ASCII characters.
- A test input file will be available and will be named, "communications.txt".

Development Standards:

- Your Code must use a consistent variable naming convention.
- All subroutines, functions, and methods must be documented with comments explaining the purpose of the method, the input parameters (if any), and the output (if any).
- If you create a class, then you must use Javadoc comments.

Note to Graders: The contestants have been given a file named communications.txt. You must first delete that file from the contestant's flash drive and run the program for Test case #1. This should result in a message that the file was not found. Then replace the file on their flash drive with the grader's file with the same name and run Test case #2. You'll run their program with a grader's file named communication.txt which will contain different data than the student file. This will produce different results than the student's file. You must copy the grader's communication.txt file into the folder where their program runs. Listed below are the test cases and their results.

Test case #1 verifies that the program checks for the correct input file. The output wording does *not* need to be exact.

Text case #2 processes the correct data file name and produces output. The output wording *must* be exact.

Test Case #1

The input file was not found.

Test case #2

transmission 005 confirmed
BPAe>?/eovAr over

transmission 222 check total error
AAeovAr over

transmission 333 length error
>?/As>?/ePeovAr over

transmission 444 incomplete transmission
BPaovAr over

transmission 555 length error, check total error, incomplete transmission
chAck over

Grader's data

5
5 00421 005 fFe t over
222 00065 002 AA over
333 00550 001 test F over
444 00713 007 fFaover
555 03333 022 check

Solution and Project

The project is present on the flash drive	_____	10 points
The projects main class is named **Communications**	_____	10 points

Program Execution

The program runs from the USB flash drive	_____	15 points

If the program does *not* execute, then the remaining items in this section receive a score of zero.

The program runs and reads the input file	_____	20 points
The program displays an error message if the file cannot be found	_____	20 points
The program displays the 3-digit transmission number	_____	10 points
The program displays "confirmed" if transmission passes all checks	_____	20 points
The program displays "length error" correctly	_____	20 points
The program displays "check total error" correctly	_____	20 points
The program displays "incomplete transmission error" correctly	_____	20 points
The program displays the encoded original message adding " over" at the end	_____	30 points

Source Code Review

The source code is properly commented		
A comment containing the contestant number is present	_____	10 points
Methods and code sections are commented	_____	20 points
A method exists to perform the "length error" check	_____	20 points
A method exists to perform the "check total error" check	_____	20 points
A method exists to perform the "incomplete transmission" check	_____	20 points
A method exists to perform the encoding of the original message	_____	20 points
A method exists to perform the display of output	_____	20 points
Code uses try... catch for exception handling	_____	10 points
Code uses a consistent variable naming convention	_____	10 points

Total Points = 345

Suggested Solution

```
/**
 * BPA Java Programming Contest : Communications
 *
 * @author - contestant number goes here
 *
 * @version  January 2017
 *
 */

import java.io.FileNotFoundException;
import java.io.*;
import java.util.Scanner;

public class Communications
{	//declare variables
		static int messageNum, checkTotal, messageLength;
		static String message, originalMessage;

	public static void main(String[] args) throws FileNotFoundException
	{
		try		// try to open the input file named in the command line
		{
			Scanner sc = new Scanner(new File("communications.txt"));
	// get the number of Starfleet messages to verify
			int numMessages = sc.nextInt();
	// process all messages
			for(int i = 0; i<numMessages;i++)
			{
		// get the message number, check total and length
				messageNum = sc.nextInt();
				checkTotal = sc.nextInt();
				messageLength = sc.nextInt();
		// get the message
				originalMessage = sc.nextLine();
		// process and verify the message
				message = cleanMessage();
		// build and print output
				printConfirmation();
			}
		}catch(FileNotFoundException e) //if file not found display error message
			{
				System.out.println(" Input file not found");
			}

	}

	// This method cleans extra spaces off the ends and removes the " over"
	//and returns a cleaned messager for processing
```

```
	public static String cleanMessage()
	{
		//remove beginning and ending spaces in message

	originalMessage = originalMessage.trim();
		//check to see if message ends in " over". if so remove "over"
		message = "";
		if(originalMessage.substring(originalMessage.length() - 5).equals(" over"))
			message = originalMessage.substring(0,originalMessage.length() - 5);
		else
			message = originalMessage;

		return message;
	}

	// This method verifies that the check total matches the sum of the chars in the message
	//- use cleaned message
	public static boolean verifyCheckTotal()
	{

		int sum = 0;
		for(int i = 0;i < message.length(); i++)
		{
			sum += message.charAt(i);
		}

//		System.out.println( "check = "+ checkTotal + "   sum = " + sum);
		if(sum == checkTotal)
		{
			return true;
		}
		return false;
	}

	//This method verifies the length of the message matches length transmitted
	//- use cleaned message
	public static boolean verifyLength()  //return true if length received = input length
	{
		if(messageLength == message.length())
			return true;
		return false;
	}

	// This method encodes the original message
	public static String encodeMessage()
	{			//replace in proper sequence using Strings
		originalMessage = originalMessage.replace("f","B");
		originalMessage = originalMessage.replace("F","P");
		originalMessage = originalMessage.replace("e","A");
		originalMessage = originalMessage.replace(" ","e");
		originalMessage = originalMessage.replace("t",">?/");
```

```
                return originalMessage;
        }

        // This method prepares and formats the output
        public static void printConfirmation()
        {

        System.out.printf("transmission %03d ",messageNum);  //must print 3-digits
                String temp = " confirmed";    //default message
                if(!verifyLength())   //lengths NOT equal - use cleaned message
                {
                        temp = " length error";
                }
                if(!verifyCheckTotal())   //totals NOT equal - use cleaned message
                {
                        if(temp.equals(" confirmed"))
                                temp = " check total error";
                        else
                                temp = temp + ", check total error";  //adds comma if needed
                }
                // ORIGINAL message ends in " over" ?
                if(!originalMessage.substring(originalMessage.length() - 5).equals(" over"))
                {
                        if(temp.equals(" confirmed"))
                                temp = " incomplete transmission";
                        else
                                temp = temp + ", incomplete transmission";   //adds comma if needed
                }
                System.out.println(temp);    //adds confirmation/error message
                // send converted original message + over + blank line
                System.out.println(encodeMessage() + " over\n");   //end transmission with over
        }
}
```