```
/*************************************************************************************
 * This class implements a message that allows for setting a message and priority level,
 * and encryption of the message given a particular key
 *
 * This class gives the user multiple ways to input messages into the class and has some
 * extra functions to increase usability.
 *
 * It also contains some static functions that an outside class can call
 *
 *
 * CST 183 Programming Assignment 5
 * @author Michael Clinesmith
 *
 *************************************************************************************/

public class Message
{
    //  class fields
    private String message;
    private char priority;


    /**
     * Constructor with no arguments
     */
    public Message()
    {
        priority = 'R';
        message = " ";
    }

    /**
     * Constructor with message string given
     * @param msg   String containing the message
     */

    public Message(String msg)
    {
        priority = 'R';
        message = msg;
    }

    /**
     * Constructor with message string a priority code given
     * @param msg   String containing the message
     * @param code  char containing the priority code
     */

    public Message(String msg, char code)
    {
        code = Character.toUpperCase(code);         // converts code to upper case if not already

        priority = code;
        message = msg;
    }

    /**
     * Mutator method to set priority level of message
     * @param priority  char to set priority level
     */
```

30/30 points for Program 5

Great solution overall. Testing spot on. Fun to run.

CHECKED
- Object oriented design with back-end class separate from driver
- Class components including constructors, set/get methods, and encryption method
- Correctness of encryption algorithm
- General program structure and documentation

TESTING
- Test cases:

R,Delta College rocks
DELTA
==> ROUTINE
GIWMAFSWEEJICHCNW

Z,I need more coffee
JAVA
==> FLASH
RNZEMMJRNCJFOEZ

```
    public void setPriority(char priority)
    {
        this.priority = priority;
    }

    /**
     * Mutator method to set the message
     * @param message   String containing the message
     */
    public void setMessage(String message)
    {
        this.message = message;
    }

    /**
     * Accessor method to get priority level
     * @return  char containing priority level
     */
    public char getPriority()
    {
        return priority;
    }

    /**
     * Accessor method to get message
     * @return  String containing message
     */
    public String getMessage()
    {
        return message;
    }

    /**
     * Method to get the string representation of the priority level
     * @return  String listing priority level
     */
    public String getPriorityString()
    {
        String priorStr;

        switch (priority)
        {
            case 'Z':
                priorStr = "FLASH";
                break;
            case 'O':
                priorStr = "IMMEDIATE";
                break;
            case 'P':
                priorStr = "PRIORITY";
                break;
            case 'R':
                priorStr = "ROUTINE";
                break;
            default:
                priorStr = "INVALID CODE";
        }
        return priorStr;

    }
```

Great job with the class - especially the javadoc documentation.

```java
/**
 * Static method to check if a key is all capital letters and at least 4 characters
 * @param key   String the key to check if a valid key
 * @return  boolean value, true if key it is a valid key, false if not
 */
public static boolean isValidKey(String key)
{
    boolean valid = true;

    if (key.length() < 4 )
    {
        valid = false;
    }

    for (int i = 0; i<key.length() && valid; i++)     // end for loop if invalid character found
    {
        // if character at position i is not a capital letter -> not valid
        if (!Character.isUpperCase(key.charAt(i)))
        {
            valid = false;
        }
    }
    return valid;
}

/**
 * Static method to return the key to partially decrypt a message
 * @param key   String that is a key to encrypt a message
 * @return  String that is a key to decrypt the message
 */
public static String antiKey(String key)
{
    String str = "";
    int code = 0;

    if(isValidKey(key))
    {
        StringBuilder antikey = new StringBuilder(key);
        for (int i=0; i<antikey.length(); i++)
        {
            code = 26 - letterToInt(antikey.charAt(i));      √                      // find value to add to 26
            code = code % 26;                                                       // wrap around if necessary for 'A'
            antikey.setCharAt(i, intToChar(code));                                      // set character
        }

        str = antikey.toString();
    }

    return str;
}

/**
 * Static method to check if string message is formatted properly with a
 * character code, a comma, then a message
 * so it can be converted into a Message class object
 * @param str   String that may be used to create a Message object
 * @return  boolean value, true if it is formatted properly, false if not
 */
public static boolean createMessageIsValid(String str)
```

```java
{
    boolean valid = true;
    char firstChar = 'R';
    int anyCommas = 0;

    if (str.length()<3)                  // if less than three characters, message not valid
    {
        valid = false;
    }
    else
    {
        firstChar = str.charAt(0);
                                         // first character must be a valid code
        if (firstChar != 'R' && firstChar != 'P' && firstChar != 'O' && firstChar != 'Z')
        {
            valid = false;
        }
        else if (str.charAt(1) != ',') // second character must be a comma
        {
            valid = false;
        }

    }
    return valid;

}

/**
 * Static method to create a Message object if formatted properly with a
 * character code, a comma, then a message
 * If not properly formatted, an empty default Message object is created
 *
 * @param str   String to create a Message object
 * @return  Message object storing a priority code and message
 */
public static Message createMessage(String str)
{
    Message msg = new Message();    // default object
    char prior;
    String strMsg;

    if (createMessageIsValid(str))  // if valid, set Message object
    {
        prior = str.charAt(0);
        strMsg = str.substring(2);
        msg.setPriority(prior);
        msg.setMessage(strMsg);
    }

    return msg;
}

/**
 * Method to encrypt a message if the given key is valid
 * @param key   String used to encrypt a message
 * @return      boolean value, true if message was encrypted, false if not
 */
public boolean encryptMessage(String key)
{
    boolean valid = true;
```

```
        if (isValidKey(key))
        {
            formatMessage();            // changes to uppercase and removes whitespace and special characters
            encrypt(key);               // encrypts message based on key
        }
        else
        {
            valid = false;              // do nothing if not valid key but return false
        }

        return valid;
    }

    /**
     * Method to check if the message has already been formatted to remove punctuation and white space
     * @return  boolean value, true if the message has had punctuation and white space removed, false if not
     */
    public boolean isFormatted()
    {
        boolean valid = true;

        for (int i=0; i<message.length() && valid; i++)     // end for loop if invalid character found
        {
            // if character at position i is not (a letter or digit) or is lower case -> not formatted
            if (!Character.isLetterOrDigit(message.charAt(i)) || Character.isLowerCase(message.charAt(i)))
            {
                valid = false;
            }
        }

        return false;
    }

    /**
     * Method to check if the priority code is valid
     * @return  boolean value, true if priority code is valid, false if not
     */
    public boolean isValidCode()
    {
        boolean valid = false;
        if (priority == 'R' || priority == 'P' || priority == 'O' || priority == 'Z')
        {
            valid = true;
        }

        return valid;
    }

    /**
     * Method to convert a message (including a priority code) to a string
     * @return  String including a priority string then the message
     */
    public String toString()
    {
        return getPriorityString() + "\n" + message;
    }

    /**
     * Private method to convert the value stored in message to uppercase without spaces or punctuation
     * This method uses the StringBuilder class to modify the string
```

```
 */
private void formatMessage()
{
    if (!isFormatted())
    {
        StringBuilder msg = new StringBuilder(message.toUpperCase());
        int i=0;

        while (i<msg.length())
        {
            // check char at position i, if not character or digit, delete it, otherwise increase i
            if (Character.isLetterOrDigit(msg.charAt(i)))
            {
                i++;
            }
            else
            {
                msg.deleteCharAt(i);
            }
        }

        message = msg.toString();           // set message to modified msg
    }

}

/**
 * Private method that encrypts the value stored in message based on the given key
 * This method uses the StringBuilder class to modify a string that is being encrypted
 * The key is modified to repeat so it has the same number of letters as the message
 * Then the letters are "added" to encrypt the message
 *
 * @param key String that is to be used to encrypt the message
 */
private void encrypt(String key)
{
    StringBuilder msg = new StringBuilder(message);
    StringBuilder keymsg = new StringBuilder("");
    int code=0;                                       // used to add letters

    // create keyword with length of message
    for (int i=0; i<msg.length()/key.length(); i++)    // copy msg.length()/key.length() copies of key
    {
        keymsg.append(key);
    }
    if (msg.length() % key.length() !=0)               // if remander, and enough characters from key to keymsg
    {
        keymsg.append(key.substring(0,msg.length()%key.length()));
    }

    // encrypt message, one character at a time
```

Good, concise solution for encryption nicely described.

```
    for (int i=0; i<msg.length(); i++)
    {
        if (Character.isLetter(msg.charAt(i)) )         // only encrypt if character is a capital letter
        {
            code = letterToInt(msg.charAt(i)) + letterToInt(keymsg.charAt(i));  // add two letters
            code = code % 26;                                                    // wrap around if necessary
            msg.setCharAt(i, intToChar(code));                                   // set character
        }
        }

        message = msg.toString();                                              // store encrypted message

    }

    /**
     * Private static method that changes a capital letter to a number (A-0, B-1, ..., Z-25)
     * @param chr   char a character
     * @return      int a number representing the character
     */
    private static int letterToInt(char chr)
    {
        return (int) chr - (int) 'A';
    }

    /**
     * Private static method that changes a number to a capital letter (0-A, 1-B, ..., 25-Z)
     * @param num   int a number
     * @return      char a capital letter representing the number
     */
    private static char intToChar(int num)
    {
        String str = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
        return str.charAt(num);
    }
}
```

```
import javax.swing.JOptionPane;

/*************************************************************************************************
 *  This class demonstrates the functionality of the Message class
 *  by requesting the user to enter a message to encode
 *
 *  It does allow the user to enter digits in a message, but does not encode them.
 *  Some dialogue boxes received seperate functions to shorten the main function code
 *
 *  CST 183 Programming Assignment 5
 *  @author Michael Clinesmith
 *************************************************************************************************/

public class MessageTest
{

    public static void main(String[] args)
    {
        Message messageObject;
        String inputString, outputString, keyWord, messageText;
        boolean isValid, anotherEncryption = true;
        char code;

        printOpeningMessage();              // displays opening message

        while (anotherEncryption)           // loops while user wants to do another encryption
        {

            printFormatMessage();           // displays formatting message
            inputString = getMessage();     // gets message from user

            isValid = Message.createMessageIsValid(inputString);    // checks if message is valid

            while (!isValid)                                        // repeat if problem with input
            {
                outputString = "There was a problem with your message, please try again.";
                JOptionPane.showMessageDialog(null, outputString);

                printFormatMessage();
                inputString = getMessage();

                isValid = Message.createMessageIsValid(inputString);
            }

            messageObject = Message.createMessage(inputString);

            keyWord = getKeyword();                                 // gets keyword from user

            isValid = Message.isValidKey(keyWord);                  // checks if keyword is valid
            while (!isValid)                                        // repeat if problem with input
            {
                outputString = "There was a problem with your keyword, please try again.";
                JOptionPane.showMessageDialog(null, outputString);

                keyWord = getKeyword();
                isValid = Message.isValidKey(keyWord);
            }

            messageText = messageObject.getMessage();
        messageObject.encryptMessage(keyWord);

        printEncryptionMessage(messageText, keyWord, messageObject);    // displays information regarding the message

        anotherEncryption = askIfAnotherEncryption();           // asks user if another encryption wanted
    }

    printClosingMessage();                                      // displays ending message

}

/**
 *  This method displays an opening message for the user regarding the program
 */
public static void printOpeningMessage()
{
    String outputString;

    outputString = "Welcome to the Message Encryption Program!\n\n" +
            "This program allows you to enter a priority code and a message,\n" +
            "then will encrypt it based on the key you provide.\n\n" +
            "Program designed by Michael Clinesmith";
    JOptionPane.showMessageDialog(null, outputString);
}
```

Thorough work on user interface.

```
/**
 * This method displays the codes and format the message needs to be in
 */
public static void printFormatMessage()
{
    String outputString;

    outputString = "You will enter a code then a message in the form of code,message\n" +
            "The possible message codes are:\n" +
            "Z - FLASH\n" +
            "O - IMMEDIATE\n" +
            "P - PRIORITY\n" +
            "R - ROUTINE\n\n" +
            "One example is given below:\n\n" +
            "P,Delta College is closed.";

    JOptionPane.showMessageDialog(null, outputString);
}

/**
 * This method requests the user for a message to encode
 * @return  String representing the message (including the priority code)
 */
public static String getMessage()
{
    String outputString, inputString;

    outputString = "Please enter your code, a comma, then your message:\n" +
            "Codes: Z, O, P or R";

    inputString = JOptionPane.showInputDialog(outputString);

    if (inputString==null)              // catch if user cancelled dialogue box
    {
        inputString = "";
        }

        return inputString;
    }

    /**
     * This method requests the user for a keyword to encode a message
     * @return  String representing the keyword
     */
    public static String getKeyword()
    {
        String outputString, inputString;

        outputString = "Please enter a keyword consisting of all capital letters, at least four letters in length:";

        inputString = JOptionPane.showInputDialog(outputString);

        if (inputString==null)              // catch if user cancelled dialogue box
        {
            inputString = "";
        }

        return inputString;
    }

    /**
     * This method displays information regarding the message that was encrypted
     * @param messageText   String for the initial message
     * @param keyWord       String the keyword used to encode the message
     * @param messageObject Message the object containing the encrypted message
     */
    public static void printEncryptionMessage(String messageText, String keyWord, Message messageObject)
    {
        String outputString;

        outputString = "Original message\n" +
                        messageText +
                        "\n-------------------------------------------------------\n" +
                        "Priority code\n" +
                        messageObject.getPriority() +
                        "\n-------------------------------------------------------\n" +
                        "Keyword\n" +
                        keyWord +
                        "\n-------------------------------------------------------\n" +
                        "Encrypted message\n" +
                        messageObject.toString();

        JOptionPane.showMessageDialog(null, outputString);
    }

    /**
     * This method requests if the user wants to do another encryption
     * @return  boolean value, true if the user entered a message beginning with 'Y' or 'y', false otherwise
     */
    public static boolean askIfAnotherEncryption()
    {
        String outputString, inputString;
        boolean isYes = false;

        outputString = "Do you want to do another encryption? Y/N";
        inputString = JOptionPane.showInputDialog(outputString);

        if (inputString != null && inputString.length()>0)          // catch if user cancelled or did not submit anything
        {
            if(inputString.charAt(0) == 'y' || inputString.charAt(0) == 'Y')
            {
                isYes = true;
            }
        }
        return isYes;
    }

    /**
     * This method displays an ending message
     */
    public static void printClosingMessage()
    {
        String outputString;
        outputString ="Thank you for using the Message Encryption Program!\n";
        JOptionPane.showMessageDialog(null, outputString);
    }
}
```