```
/* SELF ASSESSMENT

 1. Did I use easy-to-understand meaningful variable names formatted properly (in lowerCamelCase)?

        Mark out of 5:5
        Comment:  I used easy to understand variable names.

 2. Did I indent the code appropriately?

        Mark out of 5: 5
        Comment: Yes, the code is indented well.

 3. Did I write the createCipher function correctly (parameters, return type and function body) and invoke it correctly?

       Mark out of 20: 20
        Comment:  I created the create cipher program propertly and it functions propertly creating a random cipher every time.

 4. Did I write the encrypt function correctly (parameters, return type and function body) and invoke it correctly?

       Mark out of 20: 20
        Comment: The encrypt function uses the correct parameters, has the correct return type as well as a clean and efficent function body
All is invoked to a good level.

 5. Did I write the decrypt function correctly (parameters, return type and function body) and invoke it correctly?

       Mark out of 20:   20
        Comment:The decrypt is similar to the encrypt but has diffrences as it has diffrent needs. The function works fine and decrypts the
String perfectly.

 6. Did I write the main function body correctly (repeatedly obtaining a string and encrypting it and then decrypting the encrypted version)?

       Mark out of 25   25
        Comment:The main body is written correctly and repeats infinetly. It encrypts and decrypts fine and has no problems when running.

 7. How well did I complete this self-assessment?

        Mark out of 5: 5
        Comment: I completed this self assesment to a high level and fullfiled all the tasks.

 Total Mark out of 100 (Add all the previous marks):

*/
import java.util.Scanner;
import java.util.Random;

public class Cipher {

        public static int[] initialise ( int[] charAlreadyDecrypted)
        {
                for ( int count = 0; count < charAlreadyDecrypted.length; count++)
                {
                        charAlreadyDecrypted[count] = -1;
                }
                return charAlreadyDecrypted;
        }

        public static char[] createCipher ()
        {
                char[] cipher = new char[27];
                Random generator = new Random();
                int spaceValueLocation = generator.nextInt(25);
                cipher[spaceValueLocation] = (char)32;
                boolean finished = false;
                boolean charNotAlreadyUsed = true;
                char newChar = 0;

                for ( int currentLetter = 0; currentLetter < 27 ; currentLetter++ )
                {
                        if ( cipher[currentLetter] == 32 )
                        {
                                finished = true;
                        }
                        while ( !finished )
                        {
                                newChar = (char)(97 + generator.nextInt(26));
                                for ( int counter = 0; counter<= currentLetter; counter++)
                                {
                                        if ( newChar == cipher[counter])
                                        {
                                                charNotAlreadyUsed = false;
                                                counter = currentLetter;
                                        }
                                        else if ( counter == currentLetter && charNotAlreadyUsed)
                                        {
                                                cipher[counter] = newChar;
                                                finished = true;
```

```
                                }
                        }
                        charNotAlreadyUsed = true;
                }
                finished = false;
        }
        return cipher;


}

public static String encrypt ( char[] cipher, char[] characterArray)
{
        char[] encryptedText = new char[characterArray.length];

        for ( int count = 0 ; count < 27 ; count++ )
        {
                for ( int counter = 0 ; counter < characterArray.length ; counter++ )
                {
                        if (characterArray[counter] == 32)
                        {
                                encryptedText[counter] = cipher[26];
                        }
                        if ( characterArray[counter] == (char)(count + 97 ))
                        {
                                if ( cipher[count] == 123)
                                {
                                        encryptedText[counter] = 32;
                                }
                                else
                                {
                                        encryptedText[counter] = cipher[count];
                                }
                        }
                }
        }
        String encryptedTextAsString = new String( encryptedText );
        return encryptedTextAsString;
}

public static String decrypt ( char[] cipher, char[] decryptedTextAsArray)
{
        char[] decryptedText = new char[decryptedTextAsArray.length];

        for ( int count = 0 ; count < 27 ; count++ )
        {
                for ( int counter = 0 ; counter < decryptedTextAsArray.length ; counter++ )
                {
                        if ( decryptedTextAsArray[counter] == cipher[count])
                        {
                                if ( count == 26)
                                {
                                        decryptedText[counter] = 32;
                                }
                                else
                                {
                                        decryptedText[counter] = (char)(count + 97 );
                                }
                        }
                }
        }
        String decryptedTextAsString = new String( decryptedText );
        return decryptedTextAsString;
}

 public static void main(String[] args) {

        char[] cipher = createCipher ();
        Scanner input = new Scanner ( System.in );
        boolean finished = false;

        while ( !finished)
        {
                System.out.println("Please enter your text.");
                String myString = input.nextLine();
                myString = myString.toLowerCase();
                char[] characterArray = myString.toCharArray();
                String encryptedText = encrypt ( cipher, characterArray);
                System.out.println("Encrypted text:  "+encryptedText);
                 char[] decryptedTextAsArray = encryptedText.toCharArray();
                String decryptedText = decrypt ( cipher, decryptedTextAsArray);
                System.out.println("Decrypted text:  "+decryptedText);

        }
        input.close();

}
```

```
}
```