ASCII Code Generator

Introduction

Parity bits are used during information transfer as a form of error checking. When old modems transmitted data over a network or communications device, the data was encoded into a series of so-called bits and zeros (modern modems send information in packets and check for errors in different ways, therefore a parity bit is unnecessary). Only two possible states, 0 (off) or 1 (on), can be described with a single bit. Thus, it takes several bits to describe a letter or number. A total of seven bits are used to describe the set of ASCII characters, including the letters and numbers which make up this document.

For example, the letter "A" has ASCII code 65 which is 1000001 in binary. Note that each binary digit corresponds to a power of two, and 65, which is two to sixth power plus two to zero power, has only one in its binary representation 's sixth and zeroth columns. The letter C has the binary sequence 1000011, having code value 67.

Since noise or electrical interference can sometimes interfere with data transmission, it's useful to have some way to easily detect information errors as they are exchanged. One way to do this is to add an 8th bit to transmitted characters, called a parity bit. The parity bit is selected in such a way that all properly transmitted characters would either have an even or uncommon number of ones. Thus, if a zero for letter A, which produces 01000001, were placed at the start of the code, one would be placed before the code for C, producing 11000011.

The prefixes shown here produce binary codes with even numbers (and zeros), and thus the added bit is called a parity bit even. It might also be possible to choose the leading ones and zeros to always produce odd numbers of ones and zeros and this scheme is called odd parity. E7, which uses one even parity bit and seven data bits, and N8, which implies no parity bit and the use of all eight bits for data transmission, are common settings in communications programmes.

Using the wrong parity setting will generate screens full of garbage characters in a communications programme. If this happens, check to see what parity settings expected of the computer you are connecting to. If this information is not available, try connecting to a different parity setting and see if it produces legible text.

Diagrams

UML Class Diagram

This project contains just one class which is the Main class which extends the Application class.

The members' fields of the class are instance for the nodes that will be placed on the screen. It defines 4 variables for TextArea and 2 RadioButtons.

Also, it has 6 methods. Two of them are crucial for the application runtime like start method and main method. The rest are helper methods to do the needed functionality of the project.

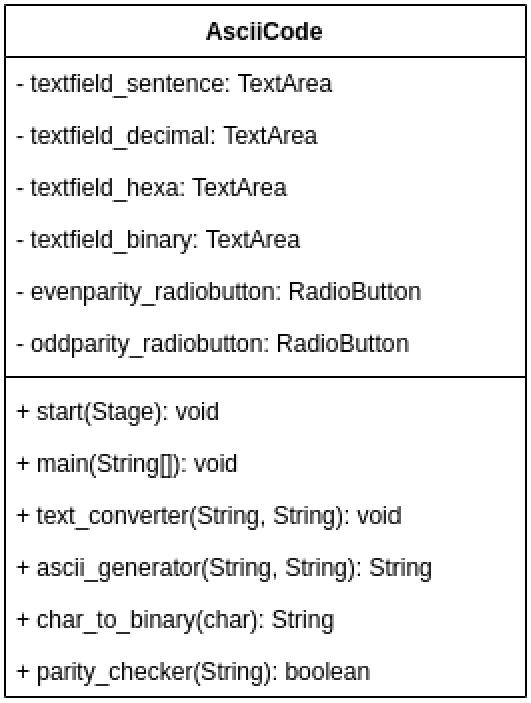

AsciiCode
- textfield_sentence: TextArea
- textfield_decimal: TextArea
- textfield_hexa: TextArea
- textfield_binary: TextArea
- evenparity_radiobutton: RadioButton
- oddparity_radiobutton: RadioButton
+ start(Stage): void
+ main(String[]): void
+ text_converter(String, String): void
+ ascii_generator(String, String): String
+ char_to_binary(char): String
+ parity_checker(String): boolean

Figure 1 Class Diagram

Flowchart

Figure 2 shows the flowchart of the program which explain the control flow and the algorithm for that. It starts by taking the input from the user, then, check the needed parity. If the needed parity is even, the text will be converted to even parity. Else, odd parity will be considered.

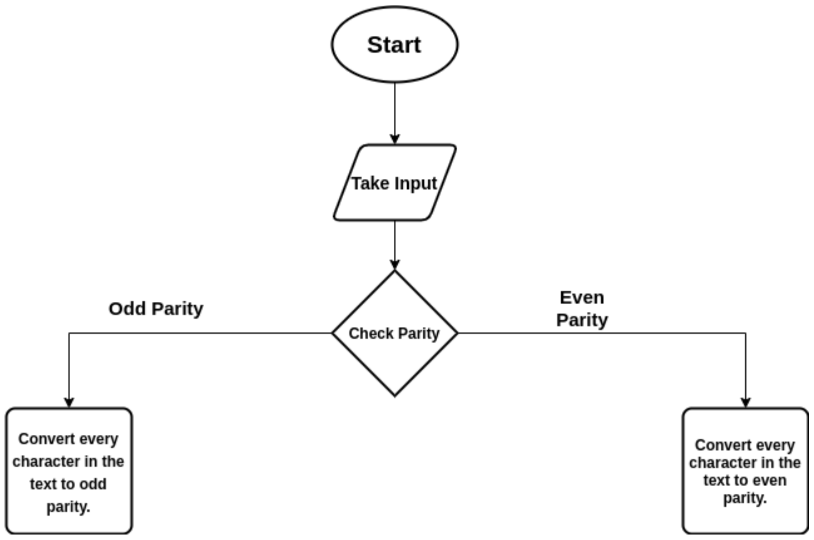

Start
Take Input
Check Parity
Odd Parity
Even Parity
Convert every character in the text to odd parity.
Convert every character in the text to even parity.

Figure 2 Flowchart Diagram

Used packages

Application class is the super class from which any JavaFx application should extend.

```
import javafx.application.Application;
```

This class defines offsets for 4 directions: up, down, left, and right of a rectangular area.

```
import javafx.geometry.Insets;
```

This class contains a set of values that describes the position of vertical and horizontal elements and alignment.

```
import javafx.geometry.Pos;
```

The Scene class acts like a container for the contents in a scene graph.

```
import javafx.scene.Scene;
```

This is the base class for all user interface controls like buttons and radio buttons.

```
import javafx.scene.control.*;
```

HBox lays out its children in a single horizontal row.

```
import javafx.scene.layout.HBox;
```

Base class for layout panes which need to expose the children list as public so that users of the subclass can freely add/remove children.

```java
import javafx.scene.layout.Pane;
```

The Color class is used to encapsulate colors in the default sRGB color space.

```java
import javafx.scene.paint.Color;
```

The Font class represents fonts, which are used to render text on screen.

```java
import javafx.scene.text.Font;
```

Specifies whether the font is italicized.

```java
import javafx.scene.text.FontPosture;
```

Specifies different font weights which can be used when searching for a font on the system.

```java
import javafx.scene.text.FontWeight;
```

The TextAlignment enum represents the horizontal text alignment.

```java
import javafx.scene.text.TextAlignment;
```

The JavaFX Stage class is the top level JavaFX container. The primary Stage is constructed by the platform.

```java
import javafx.stage.Stage;
```

Source Code and explanation

The AsciiCode class is the main class for the program. It extends Application class because it is needed to work as JavaFx program. This AsciiCode class defines 4 text areas that are responsible for taking the input from the user and display the desired results. The radio buttons are used to make the user selecting the needed conversion method. The start method is the method from which the JavaFx project starts its execution.

```java
public class AsciiCode extends Application {

    private TextArea textfield_sentence, textfield_decimal, textfield_hexa, textfield_binary;
    private RadioButton oddparity_radiobutton, evenparity_radiobutton;

    @Override
    public void start(Stage primaryStage) {
}
```

These lines are used for creating the main pane in the program and setting its different properties like preferred width and height.

```java
public class AsciiCode extends Application {

    private TextArea textfield_sentence, textfield_decimal, textfield_hexa, textfield_binary;
    private RadioButton oddparity_radiobutton, evenparity_radiobutton;

    @Override
    public void start(Stage primaryStage) {

        // Creating a Pane and setting properties to it related to the layout.
        Pane primary_pane = new Pane();
        primary_pane.setMinHeight(Double.NEGATIVE_INFINITY);
        primary_pane.setMinWidth(Double.NEGATIVE_INFINITY);
        primary_pane.setMaxHeight(Double.NEGATIVE_INFINITY);
        primary_pane.setMaxWidth(Double.NEGATIVE_INFINITY);
        primary_pane.setPrefHeight(400.0);
        primary_pane.setPrefWidth(600.0);
.....
```

Then setting the labels for the title of the program and customize it for different color and font. Also, setting the label for the text area which the user will enter his input.

```java
public class AsciiCode extends Application {

    private TextArea textfield_sentence, textfield_decimal, textfield_hexa, textfield_binary;
    private RadioButton oddparity_radiobutton, evenparity_radiobutton;

    @Override
    public void start(Stage primaryStage) {

        // Creating a Pane and setting properties to it related to the layout.
        Pane primary_pane = new Pane();
        primary_pane.setMinHeight(Double.NEGATIVE_INFINITY);
        primary_pane.setMinWidth(Double.NEGATIVE_INFINITY);
        primary_pane.setMaxHeight(Double.NEGATIVE_INFINITY);
        primary_pane.setMaxWidth(Double.NEGATIVE_INFINITY);
        primary_pane.setPrefHeight(400.0);
        primary_pane.setPrefWidth(600.0);

        // Creating title label
        Label label_title = new Label();
        label_title.setLayoutX(83.0);
        label_title.setLayoutY(14.0);
        label_title.setText("ASCII CODE GENERATOR PROGRAM");
        label_title.setTextFill(Color.valueOf("#0d086e"));
        label_title.setFont(Font.font("System", FontWeight.BOLD, FontPosture.ITALIC, 22.0));

        Label text_label = new Label();
        text_label.setLayoutX(14.0);
        text_label.setLayoutY(61.0);
        text_label.setText("Enter Text");
        text_label.setTextFill(Color.valueOf("#c69a09"));
        text_label.setFont(Font.font("System", FontWeight.BOLD,  13.0));

......
```

The rest of lines are just drawing the rest of nodes on the screen and customizing the different attributes to change its color and font.

```java
public class AsciiCode extends Application {

    private TextArea textfield_sentence, textfield_decimal, textfield_hexa, textfield_binary;
    private RadioButton oddparity_radiobutton, evenparity_radiobutton;

    @Override
    public void start(Stage primaryStage) {

        // Creating a Pane and setting properties to it related to the layout.
        Pane primary_pane = new Pane();
        primary_pane.setMinHeight(Double.NEGATIVE_INFINITY);
        primary_pane.setMinWidth(Double.NEGATIVE_INFINITY);
        primary_pane.setMaxHeight(Double.NEGATIVE_INFINITY);
        primary_pane.setMaxWidth(Double.NEGATIVE_INFINITY);
        primary_pane.setPrefHeight(400.0);
        primary_pane.setPrefWidth(600.0);

        // Creating title label
        Label label_title = new Label();
        label_title.setLayoutX(83.0);
        label_title.setLayoutY(14.0);
        label_title.setText("ASCII CODE GENERATOR PROGRAM");
        label_title.setTextFill(Color.valueOf("#0d086e"));
        label_title.setFont(Font.font("System", FontWeight.BOLD, FontPosture.ITALIC, 22.0));

        Label text_label = new Label();
        text_label.setLayoutX(14.0);
        text_label.setLayoutY(61.0);
        text_label.setText("Enter Text");
        text_label.setTextFill(Color.valueOf("#c69a09"));
        text_label.setFont(Font.font("System", FontWeight.BOLD,  13.0));


        textfield_sentence = new TextArea();
        textfield_sentence.setLayoutX(95.0);
        textfield_sentence.setLayoutY(52.0);
        textfield_sentence.setPrefHeight(102.0);
        textfield_sentence.setPrefWidth(491.0);
        textfield_sentence.setWrapText(true);

        HBox hbox = new HBox();
        hbox.setAlignment(Pos.CENTER);
        hbox.setLayoutX(139.0);
        hbox.setLayoutY(171.0);
        hbox.setSpacing(15.0);
        hbox.setPadding(new Insets(10, 10, 10, 10));

        Button button_convert = new Button();
        button_convert.setAlignment(Pos.CENTER);
        button_convert.setMnemonicParsing(false);
        button_convert.setText("Convert");
        button_convert.setTextAlignment(TextAlignment.CENTER);
        button_convert.setTextFill(Color.valueOf("#150db2"));
        button_convert.setFont(Font.font("System", FontWeight.BOLD,  14.0));
        button_convert.setPadding(new Insets(10, 10, 10, 10));

        Button button_reset = new Button();
        button_reset.setAlignment(Pos.CENTER);
        button_reset.setMnemonicParsing(false);
        button_reset.setText("Reset");
        button_reset.setTextAlignment(TextAlignment.CENTER);
        button_reset.setTextFill(Color.valueOf("#ad3232"));
        button_reset.setFont(Font.font("System", FontWeight.BOLD,  14.0));
        button_reset.setPadding(new Insets(10, 10, 10, 10));

        evenparity_radiobutton = new RadioButton();
        oddparity_radiobutton = new RadioButton();
        evenparity_radiobutton.setMnemonicParsing(false);
        evenparity_radiobutton.setText("Even Parity");
        oddparity_radiobutton.setMnemonicParsing(false);
        oddparity_radiobutton.setText("Odd Parity");
        ToggleGroup toggleGroup = new ToggleGroup();
        oddparity_radiobutton.setToggleGroup(toggleGroup);
        evenparity_radiobutton.setToggleGroup(toggleGroup);
        hbox.getChildren().addAll(button_convert, button_reset, oddparity_radiobutton, evenparity_radiobutton);


        Label label_decimal = new Label();
        label_decimal.setLayoutX(17.0);
        label_decimal.setLayoutY(229.0);
        label_decimal.setText("Decimal Conversion");

        Label label_hexadecimal = new Label();
        label_hexadecimal.setLayoutX(378.0);
        label_hexadecimal.setLayoutY(229.0);
        label_hexadecimal.setText("Hexadecimal Conversion");

        textfield_decimal = new TextArea();
        textfield_decimal.setLayoutX(17.0);
        textfield_decimal.setLayoutY(245.0);
        textfield_decimal.setPrefHeight(147.0);
        textfield_decimal.setPrefWidth(208.0);
        textfield_decimal.setWrapText(true);
        textfield_decimal.setEditable(false);

        textfield_hexa = new TextArea();
        textfield_hexa.setLayoutX(378.0);
        textfield_hexa.setLayoutY(245.0);
        textfield_hexa.setPrefHeight(147.0);
        textfield_hexa.setPrefWidth(208.0);
        textfield_hexa.setWrapText(true);
        textfield_hexa.setEditable(false);

        textfield_binary = new TextArea();
        textfield_binary.setLayoutX(239.0);
        textfield_binary.setLayoutY(244.0);
        textfield_binary.setPrefHeight(147.0);
        textfield_binary.setPrefWidth(122.0);
        textfield_binary.setWrapText(true);
        textfield_binary.setEditable(false);

        Label label_binary = new Label();
        label_binary.setLayoutX(239.0);
        label_binary.setLayoutY(229.0);
        label_binary.setText("Binary");

        primary_pane.getChildren().addAll(label_binary, label_title, text_label, textfield_sentence, hbox, label_decimal, textfield_decimal, label_hexadecimal, textfield_hexa, textfield_binary);
       ........
```

These lines add the event listeners to the buttons. In reset button the text fields related to the main field, binary, decimal, and hexa decimal will be deleted.

In the convert button, it will check if the parity is selected correctly or not. If not, an alert will be shown to the user telling him to select proper parity. Else, the input of the user as well as the parity will be sent to the appropriate method to handle it.

```java
  button_reset.setOnAction(e -> {
        textfield_sentence.setText("");
        textfield_hexa.setText("");
        textfield_decimal.setText("");
        textfield_binary.setText("");
    });

    button_convert.setOnAction(e -> {
        String input = textfield_sentence.getText();
        if (!oddparity_radiobutton.isSelected() && !evenparity_radiobutton.isSelected
()){
            Alert alert = new Alert(Alert.AlertType.WARNING);
            alert.setTitle("Parity Selection Error");
            alert.setContentText("Please Select Parity");
            alert.showAndWait();
        } else {
            String userParity = "";
            if (evenparity_radiobutton.isSelected())
                userParity = "even";
            else if (oddparity_radiobutton.isSelected())
                userParity = "odd";

            text_converter(input, userParity);
        }


    });

    primaryStage.setTitle("ASCII Code Generator");
    primaryStage.setScene(new Scene(primary_pane));
    primaryStage.setResizable(false);
    primaryStage.show();
}
```

This method just take the input from the user and the desired parity and apply it to every character in the input. It converts the results to the different numbering systems, then, manipulate the text area to the specific output.

```java
/*
    Apply the conversion between different systems and display it on the screen.
 */
private void text_converter(String input, String userParity) {
    StringBuilder sbHexa = new StringBuilder();
    StringBuilder sbDecimal = new StringBuilder();
    StringBuilder sbBinary = new StringBuilder();
    for (int i = 0 ; i < input.length() ; i++){
        String binaryChar = char_to_binary(input.charAt(i));
        String newAscii = ascii_generator(binaryChar,userParity);
        sbHexa.append("0X").append(Integer.toString(Integer.parseInt(newAscii, 2), 16)).append(" ");
        sbDecimal.append(Integer.toString(Integer.parseInt(newAscii, 2), 10)).append(" ");
        sbBinary.append(newAscii).append(" ");
    }
    String s = sbHexa.toString();
    String decimal = sbDecimal.toString();
    textfield_hexa.setText(s);
    textfield_decimal.setText(decimal);
    textfield_binary.setText(sbBinary.toString());
}
```

This method generates a new ASCII code represents in binary based on the passed parity to it. It checks the parity of the main string. If the passed parity is opposite to the main parity then it is converted by adding the character 1 in front of the binary string. Else, there is no need to convert anything just return it as it.

```java
// generate the new ascii binary string from the given binary representation.
public static String ascii_generator(String binary, String parity){
    String newString = null;
    // checks the parity of the binary string.
    boolean parityCheck = parity_checker(binary);
    /*
        if it is even parity it checks the request of the user
        if the user needs odd parity it converts it to odd
        else it keeps it even.
        the same logic in the odd parity
     */
    if (parity.equals("even")){
        if (parityCheck){
            newString = binary;
        } else {
            newString = '1' + binary;
        }

    } else if (parity.equals("odd")){

        if (!parityCheck){
            newString = binary;
        } else {
            newString = '1' + binary;
        }

    }

    return newString;
}
```

This method just returns a string of binary numbers which represents the character which has been passed as a parameter in the method. It uses the built-in method in the Integer class which can do the desired job correctly and efficiently.

```java
// Convert character to binary result is string.
public static String char_to_binary(char ch){
    return Integer.toBinaryString((int) ch);
}
```

This method takes a string of binary numbers as a parameter. Its role is to check the parity of this parameter by counting the number of ones in the string. If the number of one is even then the parity is even and vice versa.

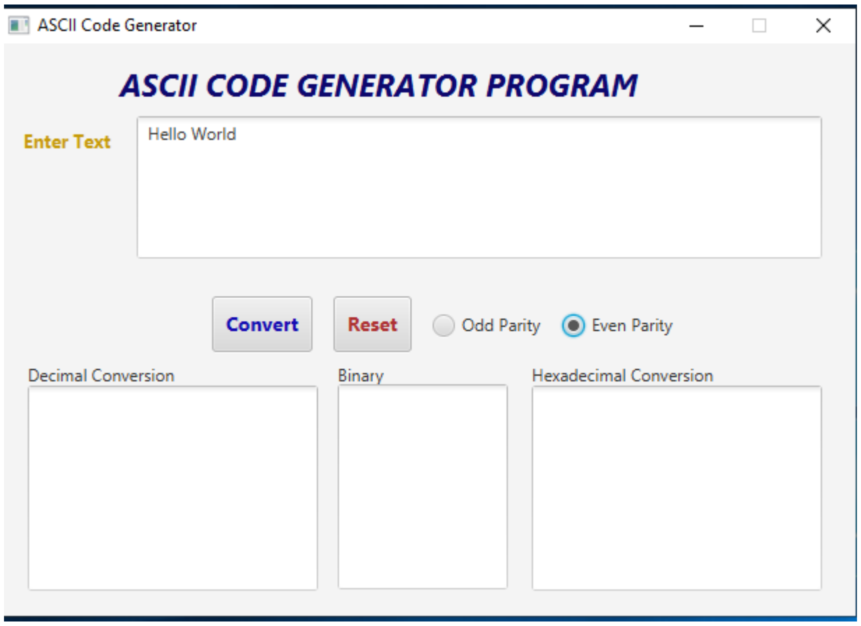
```java
// check the parity of the binary string ( counting number of ones ).
// true even parity
// false odd parity
public static boolean parity_checker(String binary){
    int parity = 0 ;
    for (int i = 0 ; i< binary.length();i++){
        if (binary.charAt(i) == '1')
            parity++;
    }
    return (parity % 2 == 0);
}
```

Runtime Results

Figure 3 shows the initial start of the program. We have a text area so the user can enter the text needed to be converted. The 2 buttons are used to trigger the conversion by clicking on “Convert” and “Reset” deletes texts in all of text areas.

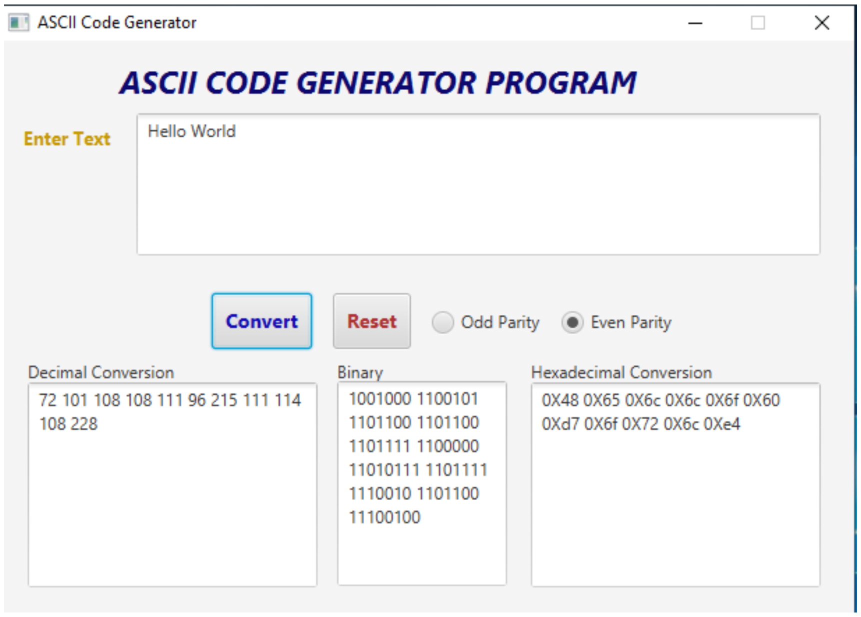
Radio buttons are used to make the user choose the parity, which is odd or even.

Figure 4 shows that the used has typed “Hello World” in the text area specified for that, then he needs to convert this text to ASCII code based on even parity. The program will convert the text into 3 forms: Decimal form, Binary, and Hexadecimal. Each of them will appear in its own text field.



```
Hello World
```

Figure 3 Start of the program

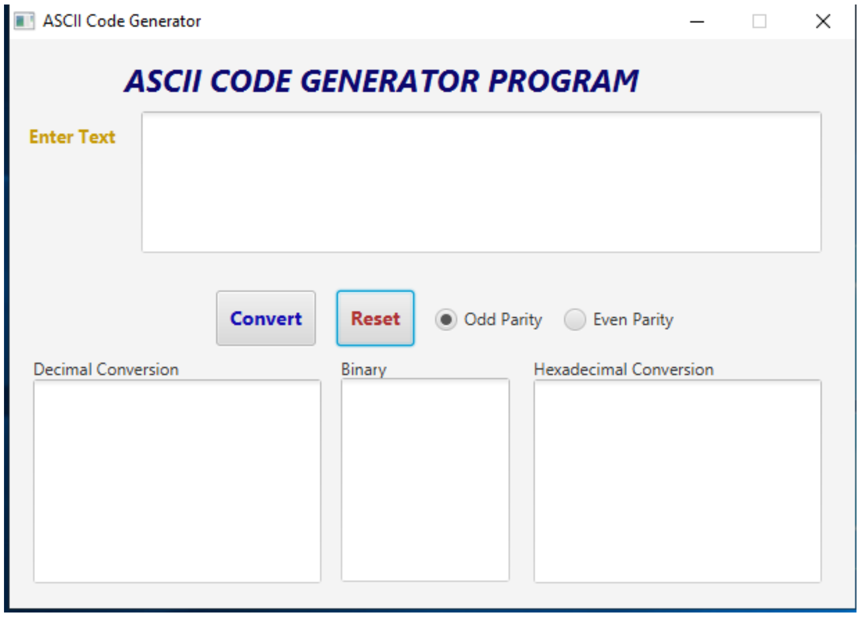


```
Hello World

Decimal Conversion
72 101 108 108 111 96 215 111 114
108 228

Binary
1001000 1100101
1101100 1101100
1101111 1100000
11010111 1101111
1110010 1101100
11100100

Hexadecimal Conversion
0X48 0X65 0X6c 0X6c 0X6f 0X60
0Xd7 0X6f 0X72 0X6c 0Xe4
```

Figure 4 Even parity conversion

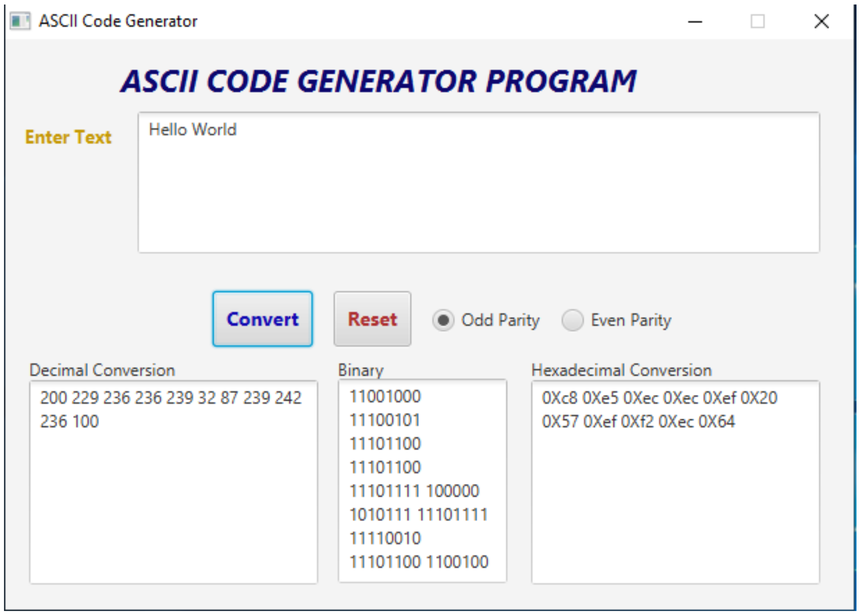
Figure 5 shows that the user needs to convert the same text to ASCII code but this time he will convert it using odd parity.

Figure 6 shows that the used has clicked on the reset button so all text fields will be all deleted.



Odd Parity [✓] Even Parity

Figure 6 Reset Button



Enter Text: Hello World

Odd Parity [✓] Even Parity

Decimal Conversion:

```
200 229 236 236 239 32 87 239 242
236 100
```

Binary:

```
11001000
11100101
11101100
11101100
11101111 100000
1010111 11101111
11110010
11101100 1100100
```

Hexadecimal Conversion:

```
0Xc8 0Xe5 0Xec 0Xec 0Xef 0X20
0X57 0Xef 0Xf2 0Xec 0X64
```

Figure 5 Odd parity conversion

Recommendations and Conclusion

Parity bits are important in many fields of communication and the efficiency of detecting and converting different inputs are highly used in error detection. Our program has converted the user input which is a text to different numbering systems based on the user selection of parity bit.

An enhanced version is to convert the binary code or another form to the text itself and check the correctness of the text.

References

1- Liang, Y. Daniel. Introduction to Java Programming. Pearson, 2018.

2- silversilver 4, et al. “ASCII with Odd Parity.” Stack Overflow, 1 June 1963, stackoverflow.com/questions/19342931/ascii-with-odd-parity.

3- “Indiana University.” What Is a Parity Bit?, kb.iu.edu/d/afdh.