

## **Assignment No. 2**

**Meet Patel**  
**( B00899516 )**

**CSCI 6704: Advance Topics in Networks**

**Dr. Srinu Sampalli**

**Dalhousie University**

**Fall 2022**

## Programming Exercise 1

Write a program that accepts a binary string as input and generates waveforms using a GUI corresponding to the following encodings of the input binary string:

- a) Unipolar
- b) NRZ
- c) Manchester

### Sample Run for Unipolar Encoding

Input: 00110100010

Output: Shown in Figure1

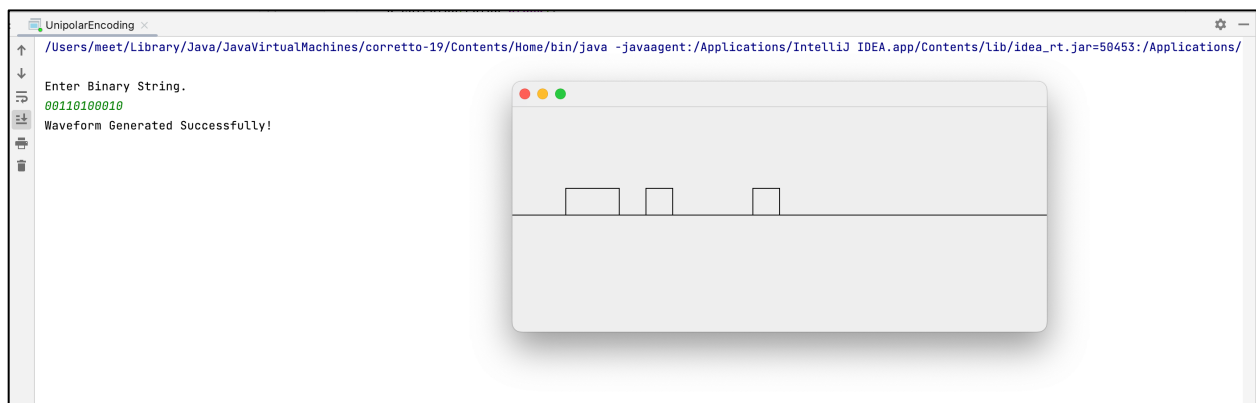


Figure 1: Sample Run for Unipolar Encodings

### Sample Run for Non-return-to-zero (NRZ) Encoding

Input: 00110100010

Output: Shown in Figure2

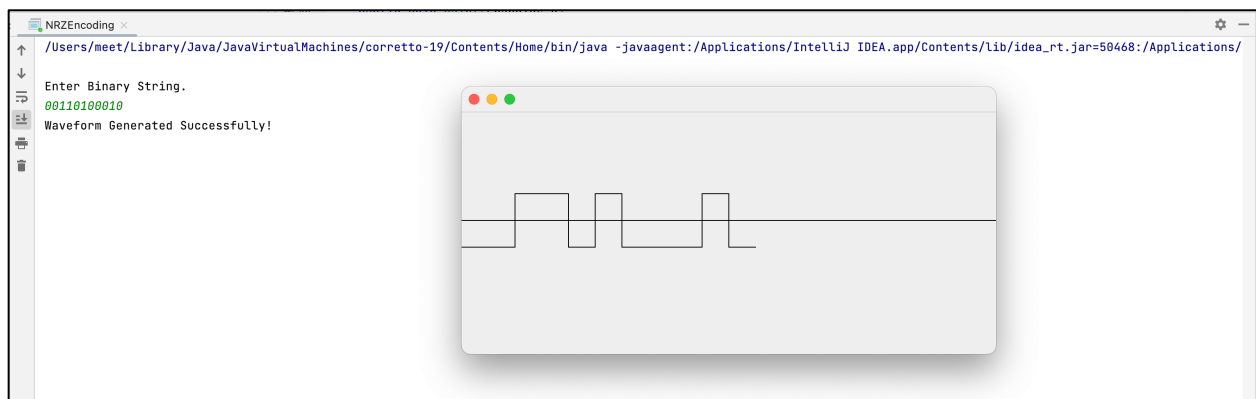


Figure 2: Sample Run for Non-return-to-zero (NRZ) Encoding

### Sample Run for Manchester Encoding

Input: 00110100010

Output: Shown in Figure3

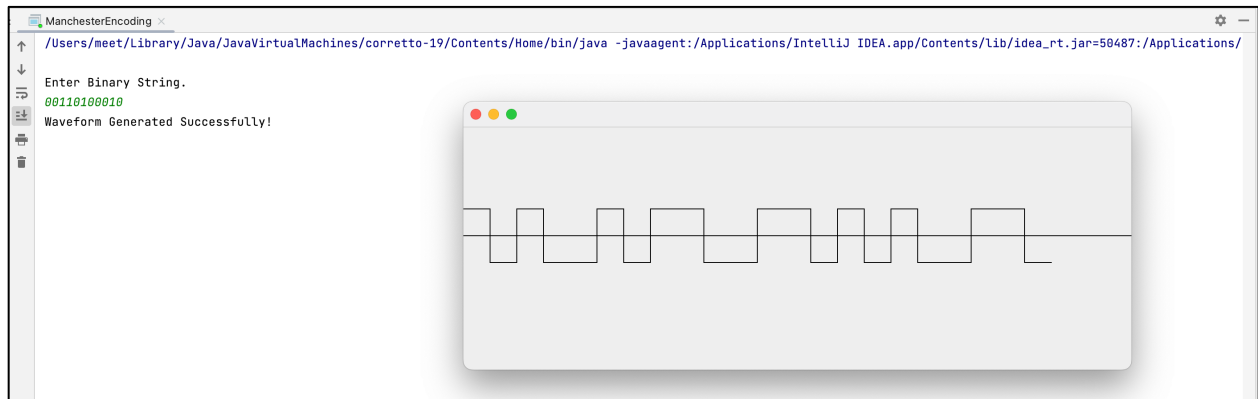


Figure 3: Sample Run for Manchester Encoding

### Source Code

The source code for **Waveform Generation Program** is saved in package "exercise\_1\_encodings".

It contains three Java files which are listed below:

1. **ManchesterEncoding** : Responsible for generating Manchester Waveform
2. **NRZEncoding** : Responsible for generating NRZ Waveform
3. **UnipolarEncoding** : Responsible for generating Unipolar Waveform

## Programming Exercise 2 <Bit Stuffing Program>

In this exercise, you will be writing a simple program that does the following:

- Read a String of hex digits
- Convert the String into a String of binary numbers
- Perform bit stuffing on the binary String
- Unstuff the bits from the binary String
- Produce the original hex String

### Sample Run 1

Enter hexadecimal string.

**ABEFFFF**

Input: ABEFFFF

Conversion to binary: 101010111110111111111111111111

After bit stuffing: 101010111110011111101111101111101

After bit unstuffing: 101010111110111111111111111111

Output: ABEFFFF

The screenshot shows an IDE window titled "assignment\_2\_code - BitsStuffingMain.java". The code is as follows:

```

package exercise_2_bit_stuffing;

import java.util.Scanner;

public class BitsStuffingMain {
    public static void main(String[] args) {
        BitsConversion bitsConversion = new BitsConversion();
        BitsStuffingAndUnstuffing bitsStuffingAndUnstuffing = new BitsStuffingAndUnstuffing();

        Scanner scanner = new Scanner(System.in);
        System.out.println("\nEnter hexadecimal string.");
        String hexadecimalInput = scanner.nextLine();
        try {
            for (int i = 0; i < hexadecimalInput.toUpperCase().length(); ++i) {
                char currentHexadecimalInputChar = hexadecimalInput.toUpperCase().charAt(i);
                if ("0123456789ABCDEF".indexOf(currentHexadecimalInputChar) == -1) {
                    throw new Exception("Invalid hexadecimal string!");
                }
            }
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }
}

```

The Run window at the bottom shows the following output:

```

/Users/meet/Library/Java/JavaVirtualMachines/corretto-19/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA.app/Contents/lib/idea_rt.jar=50526:/Applications/
Enter hexadecimal string.
ABEFFFF
Input: ABEFFFF
Conversion to binary: 101010111110111111111111111111
After bit stuffing: 101010111110011111101111101111101
After bit unstuffing: 101010111110111111111111111111
Output: ABEFFFF

Process finished with exit code 0

```

Figure 4: Output of Bit Stuffing Program for input hexadecimal string ABEFFFF

## Sample Run 2

Enter hexadecimal string.

**ABCDEF**

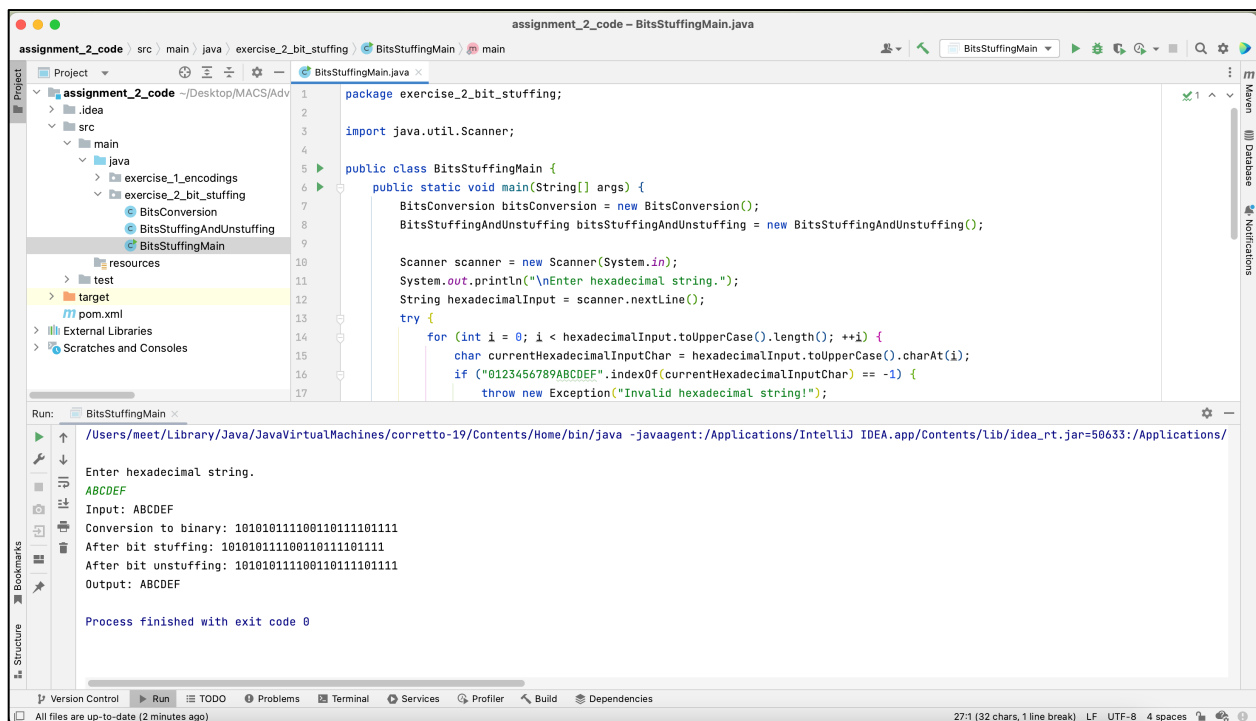
Input: ABCDEF

Conversion to binary: 101010111100110111101111

After bit stuffing: 101010111100110111101111

After bit unstuffing: 101010111100110111101111

Output: ABCDEF



```
package exercise_2_bit_stuffing;

import java.util.Scanner;

public class BitsStuffingMain {
    public static void main(String[] args) {
        BitsConversion bitsConversion = new BitsConversion();
        BitsStuffingAndUnstuffing bitsStuffingAndUnstuffing = new BitsStuffingAndUnstuffing();

        Scanner scanner = new Scanner(System.in);
        System.out.println("\nEnter hexadecimal string.");
        String hexadecimalInput = scanner.nextLine();
        try {
            for (int i = 0; i < hexadecimalInput.toUpperCase().length(); ++i) {
                char currentHexadecimalInputChar = hexadecimalInput.toUpperCase().charAt(i);
                if ("0123456789ABCDEF".indexOf(currentHexadecimalInputChar) == -1) {
                    throw new Exception("Invalid hexadecimal string!");
                }
            }
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }
}
```

Run: BitsStuffingMain

```
/Users/meet/Library/Java/JavaVirtualMachines/corretto-19/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA.app/Contents/lib/idea_rt.jar=58633:/Applications/
Enter hexadecimal string.
ABCDEF
Input: ABCDEF
Conversion to binary: 101010111100110111101111
After bit stuffing: 101010111100110111101111
After bit unstuffing: 101010111100110111101111
Output: ABCDEF

Process finished with exit code 0
```

Figure 5: Output of Bit Stuffing Program for input hexadecimal string ABCDEF

### Sample Run 3

Enter hexadecimal string.

**AFFFFFFF**

Input: AFFFFFFF

Conversion to binary: 101011111111111110111011101111111

After bit stuffing: 1010111110111110111110011101110111110111

After bit unstuffing: 101011111111111110111011101111111

Output: AFFFFFFF

```

package exercise_2_bit_stuffing;

import java.util.Scanner;

public class BitsStuffingMain {
    public static void main(String[] args) {
        BitsConversion bitsConversion = new BitsConversion();
        BitsStuffingAndUnstuffing bitsStuffingAndUnstuffing = new BitsStuffingAndUnstuffing();

        Scanner scanner = new Scanner(System.in);
        System.out.println("\nEnter hexadecimal string.");
        String hexadecimalInput = scanner.nextLine();
        try {
            for (int i = 0; i < hexadecimalInput.toUpperCase().length(); ++i) {
                char currentHexadecimalInputChar = hexadecimalInput.toUpperCase().charAt(i);
                if ("0123456789ABCDEF".indexOf(currentHexadecimalInputChar) == -1) {
                    throw new Exception("Invalid hexadecimal string!");
                }
            }
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }

        String binary = bitsConversion.convertHexadecimalToBinary(hexadecimalInput);
        String stuffedBinary = bitsStuffingAndUnstuffing.performBitStuffing(binary);
        String unstuffedBinary = bitsStuffingAndUnstuffing.performBitUnstuffing(stuffedBinary);
        String outputHexadecimal = bitsConversion.convertBinaryToHexadecimal(unstuffedBinary);

        System.out.println("Input: " + hexadecimalInput);
        System.out.println("Conversion to binary: " + binary);
        System.out.println("After bit stuffing: " + stuffedBinary);
        System.out.println("After bit unstuffing: " + unstuffedBinary);
        System.out.println("Output: " + outputHexadecimal);
    }
}

```

Run: BitsStuffingMain

```

/Users/meet/Library/Java/JavaVirtualMachines/corretto-19/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA.app/Contents/lib/idea_rt.jar=50644:/Applications/
Enter hexadecimal string.
AFFFFFFF
Input: AFFFFFFF
Conversion to binary: 101011111111111110111011101111111
After bit stuffing: 1010111110111110111110011101110111110111
After bit unstuffing: 101011111111111110111011101111111
Output: AFFFFFFF

Process finished with exit code 0

```

Figure 6: Output of Bit Stuffing Program for input hexadecimal string AFFFFFFF

## Source Code

The source code for **Bit Stuffing Program** is saved in package “**exercise\_2\_bit\_stuffing**”.

It contains three JAVA files which are listed below:

1. **BitsConversion** – This java file is responsible for the conversion of Hexadecimal Input to the Binary Bits and then conversion of Binary Bits to Hexadecimal Output.
2. **BitsStuffingAndUnstuffing** – This Java file is responsible for stuffing and unstuffing 1's
3. **BitsMain** – This Java file is responsible for taking the input from the user and then passing it to the another class methods for further processing.