



Course Name: Data Communication  
Course Code: CSE-350  
Lab Report - 1

**Submitted To**

Course Instructor: Md. Mahir Ashhab  
Lecturer  
Department of Computer Science and Engineering

**Submitted By**

Md. Arif Mahmud Rakib  
Student ID: 2019-1-60-070

**Introduction:**

Block coding is a technique that enhances the performance of line coding by introducing redundancy. This redundancy serves the purpose of synchronization and inherent error detection. The process involves transforming a block of  $m$  bits into a larger block of  $n$  bits, a concept known as  $mB/nB$  encoding.

The process of block coding typically comprises three key stages: division, substitution, and combination. During division, a sequence of bits is grouped into segments of  $m$  bits each. The core of block coding lies in the substitution step. Here, the  $m$ -bit segments are replaced with  $n$ -bit segments. Finally, these  $n$ -bit segments are combined to create a new bit stream. As a result of this process, the resulting stream contains more bits compared to the original stream.

**NRZ-I**

NRZ-Invert is a signal modulation technique based on a simple principle: when a logical 1 is encountered, the signal's polarity is flipped, while a logical 0 maintains the current polarity. In other words, the signal switches its state for 1s and maintains its state for 0s. This approach characterizes the NRZ-Invert encoding method.

**NRZ-L**

In this form of Polar signaling, a positive pulse signifies a High in the data, whereas a negative pulse signifies a Low in the data. The concept is straightforward, and there are no low-frequency elements present in the representation.

**Rz**

The underlying concept of the aforementioned representation is that a logical 1 is depicted as a pulse occupying half of the positive voltage level and half of the zero voltage level. Conversely, a logical 0 is portrayed as a pulse encompassing half of the negative voltage level and half of the zero voltage level.

**Manchester**

Manchester coding, also referred to as Phase Encoding (PE), is a line coding technique designed to ensure reliable data transmission. In this method, each data bit is encoded in a way that guarantees the presence of at least one transition and occupies the same time duration. It can be viewed as a hybrid of RZ (Return-to-Zero) and NRZ-L (Non-Return-to-Zero Level) coding approaches.

In Manchester coding, a logical 1 is divided into two equal halves. The initial half is denoted by a negative voltage, followed by the latter half represented by a positive voltage. Similarly, a logical 0 is also divided into two equal halves, with the initial half being represented by a positive voltage and the latter half by a negative voltage. The significant characteristic of this coding technique is the transition occurring in the middle of each bit interval, which serves the purpose of synchronization and facilitates accurate data recovery.

### **Differential Manchester**

Bi-Phase encoding stands out as a superior choice when it comes to achieving synchronization in data transmission. This method ensures the presence of at least one transition within each bit interval and potentially represents two bits. Its signal modulation rate is double that of NRZ (Non- Return-to-Zero), resulting in a requirement for increased bandwidth. This technique can be visualized as a fusion of RZ (Return-to-Zero) and NRZ-I (Non-Return-to-Zero Inverted) strategies.

In Bi-Phase encoding, a notable feature is the presence of a transition precisely at the midpoint of every bit interval. However, the actual bit values are determined right from the beginning of each bit interval. This combination of timing and value determination contributes to its effectiveness in ensuring synchronization and accurate data retrieval.

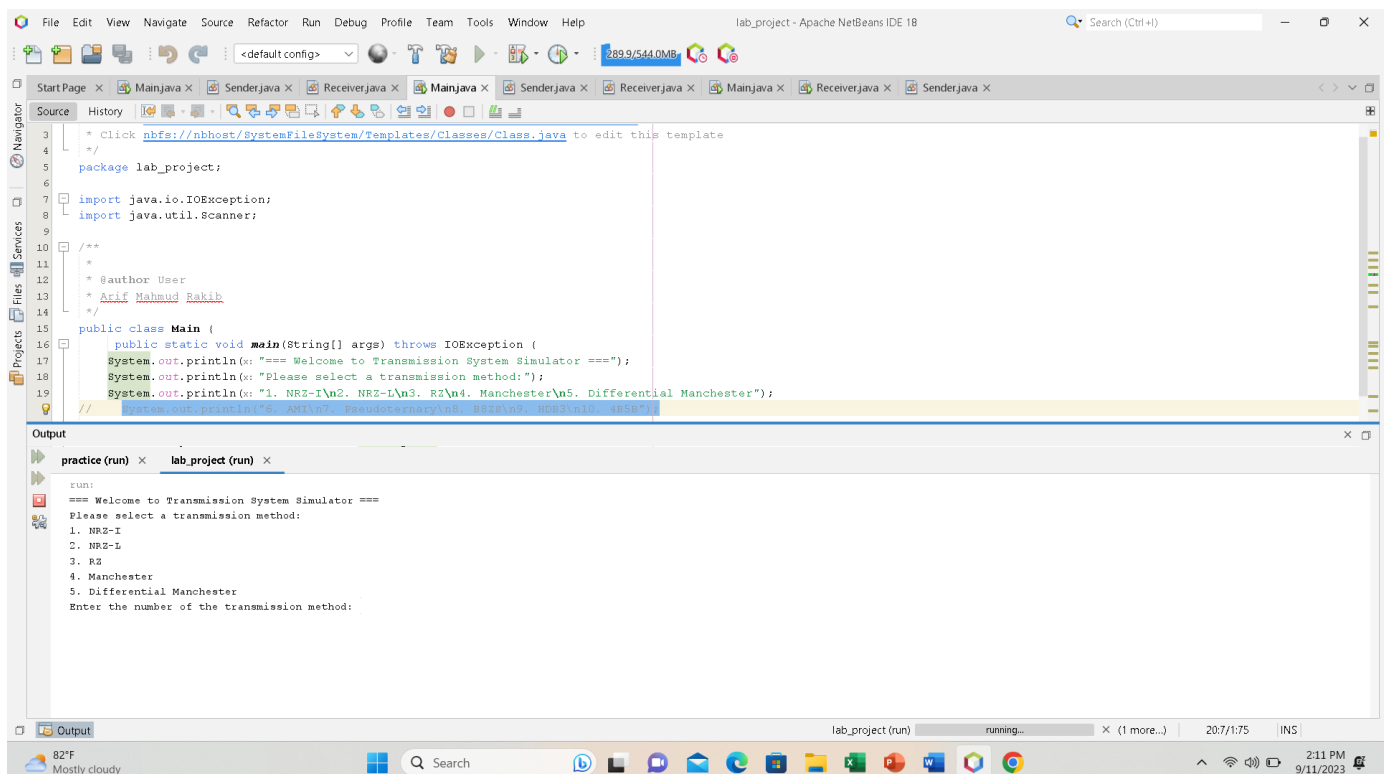
## Lab task:

The laboratory assignment focuses on the process of line coding, which involves the conversion of digital data into a digital signal. The procedure encompasses various stages, starting with the retrieval of input data from a file. Subsequently, this data is translated into a digital signal through encoding in the sender segment.

Upon reaching the receiver side, the encoded signal is decoded to recover the original data. The essential steps of reading input, encoding in the sender, decoding in the receiver, and producing the output are integral to the entire process.

In essence, the assignment delves into the intricacies of digital-to-digital conversion, whereby data undergoes transformation into a signal for transmission and then reverts to its original form through decoding at the receiving end.

## Simulation Design:



```
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
lab_project - Apache NetBeans IDE 18
Search (Ctrl+I)

Start Page X Mainjava X Senderjava X Receiverjava X Mainjava X Senderjava X Receiverjava X Mainjava X Receiverjava X Senderjava X
Source History
Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
package lab_project;

import java.io.IOException;
import java.util.Scanner;

/**
 * @author User
 * Arif Mahmud Rakib
 */
public class Main {
    public static void main(String[] args) throws IOException {
        System.out.println("=== Welcome to Transmission System Simulator ===");
        System.out.println("Please select a transmission method:");
        System.out.println("1. NRZ-I\n2. NRZ-L\n3. RZ\n4. Manchester\n5. Differential Manchester");
        // Scanner scanner = new Scanner(System.in);
    }
}
```

Output

practice (run) x lab\_project (run) x

run:

```
=== Welcome to Transmission System Simulator ===
Please select a transmission method:
1. NRZ-I
2. NRZ-L
3. RZ
4. Manchester
5. Differential Manchester
Enter the number of the transmission method:
```

lab\_project (run) running... x (1 more...) 20:7:175 INS

82°F Mostly cloudy 2:11 PM 9/11/2023

## **Implementation Details:**

### **NRZ-I**

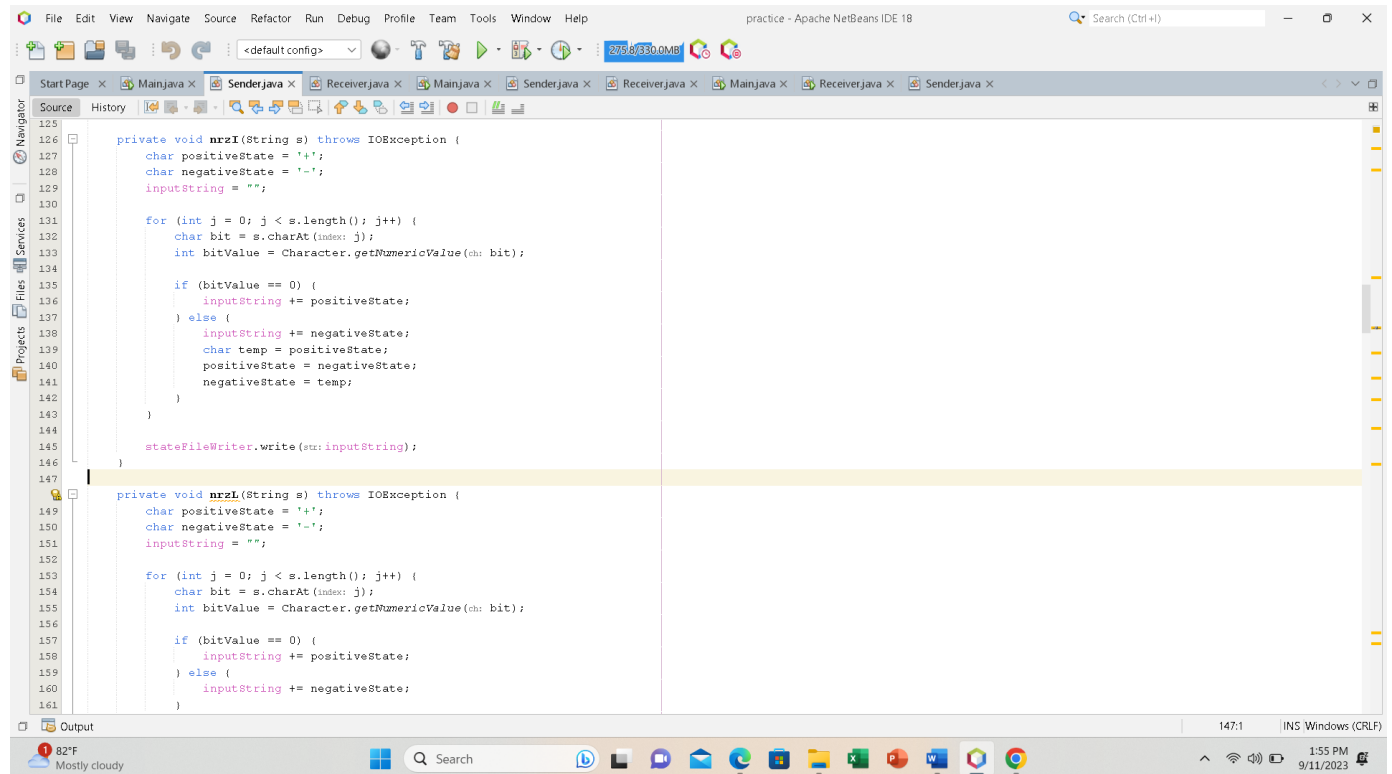
#### **Sender**

```
private void nrZI(String s) throws IOException {
    char positiveState = '+';
    char negativeState = '-';
    inputString = "";

    for (int j = 0; j < s.length(); j++) {
        char bit = s.charAt(j);
        int bitValue = Character.getNumericValue(bit);

        if (bitValue == 0) {
            inputString += positiveState;
        } else {
            inputString += negativeState;
            char temp = positiveState;
            positiveState = negativeState;
            negativeState = temp;
        }
    }

    stateFileWriter.write(inputString);
}
```



## Receiver

```
private void nrzI() throws IOException {
    char prevState = '+';
    char state = '0';
    char antiState = '1';
    BufferedReader brr = new BufferedReader(frr);
    int i = 0;
    inputString = "";
```

```
while (true) {
    int x = brr.read();
    i++;
```

```
    if (x == -1) {
        fww.write(inputString);
        break;
    }
```

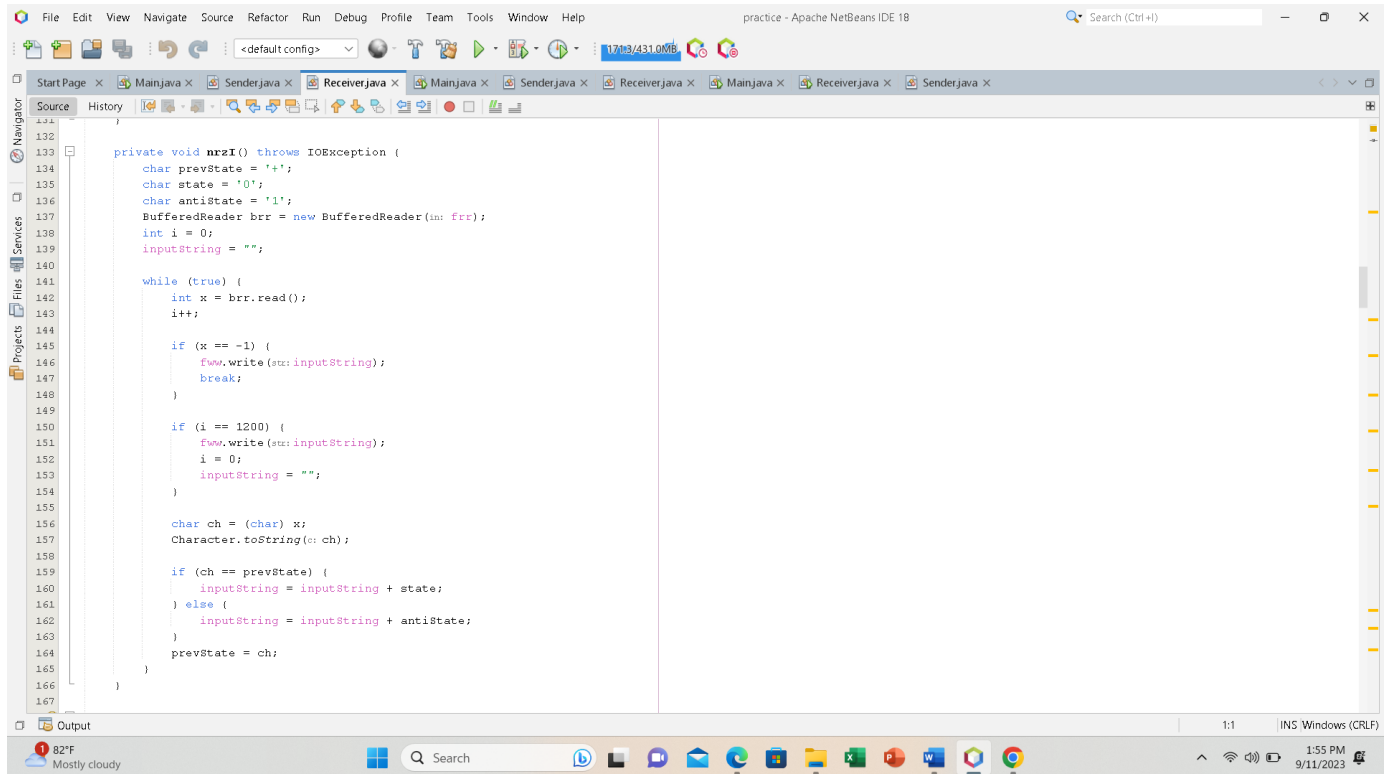
```
    if (i == 1200) {
        fww.write(inputString);
        i = 0;
        inputString = "";
    }
```

```
    char ch = (char) x;
    Character.toString(ch);
```

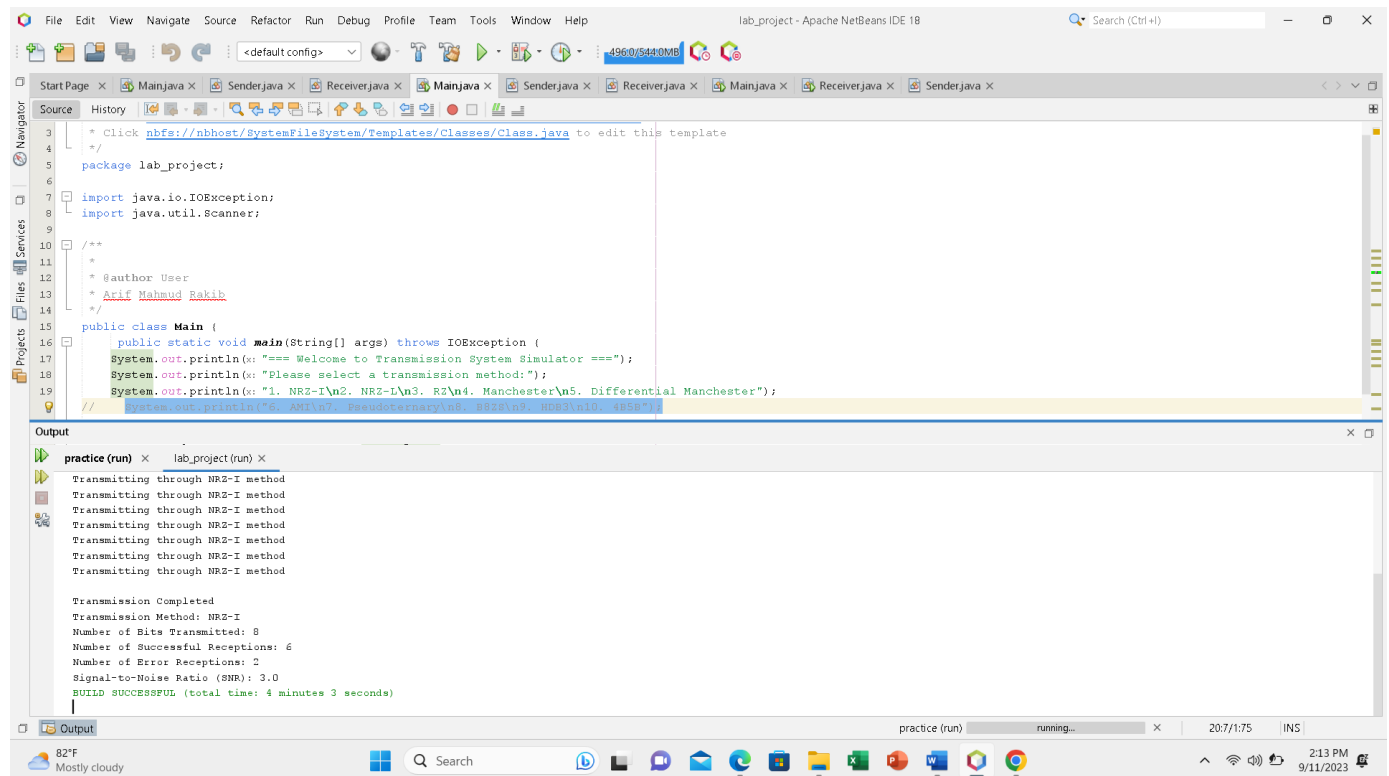
```

        if (ch == prevState) {
            inputString = inputString + state;
        } else {
            inputString = inputString + antiState;
        }
        prevState = ch;
    }
}

```



## Output NRZ-I:



The screenshot shows the Apache NetBeans IDE interface. The main editor displays the source code of a Java application. The code includes package declarations, imports for `IOException` and `Scanner`, and a `Main` class with a `main` method. The `main` method prints a welcome message and a list of transmission methods: NRZ-I, NRZ-L, Manchester, and Differential Manchester. The output window at the bottom shows the execution results of the `practice (run)` task. It displays a series of "Transmitting through NRZ-I method" messages, followed by a "Transmission Completed" message and summary statistics: Transmission Method: NRZ-I, Number of Bits Transmitted: 8, Number of Successful Receptions: 6, Number of Error Receptions: 2, Signal-to-Noise Ratio (SNR): 3.0, and BUILD SUCCESSFUL (total time: 4 minutes 3 seconds).

```
3  // * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
4  */
5  package lab_project;
6
7  import java.io.IOException;
8  import java.util.Scanner;
9
10 /**
11  *
12  * @author User
13  * @Arif Mahmud Rakib
14  */
15 public class Main {
16     public static void main(String[] args) throws IOException {
17         System.out.println("=== Welcome to Transmission System Simulator ===");
18         System.out.println("Please select a transmission method:");
19         System.out.println("1. NRZ-I\n2. NRZ-L\n3. RZ\n4. Manchester\n5. Differential Manchester");
20         // System.out.println("6. AM\n7. Pseudoternary\n8. BPSK\n9. QPSK\n10. 4FSK");
21     }
22 }
```

practice (run) x lab\_project (run) x

Transmitting through NRZ-I method  
Transmitting through NRZ-I method  
Transmitting through NRZ-I method  
Transmitting through NRZ-I method  
Transmitting through NRZ-I method  
Transmitting through NRZ-I method  
Transmitting through NRZ-I method

Transmission Completed  
Transmission Method: NRZ-I  
Number of Bits Transmitted: 8  
Number of Successful Receptions: 6  
Number of Error Receptions: 2  
Signal-to-Noise Ratio (SNR): 3.0  
BUILD SUCCESSFUL (total time: 4 minutes 3 seconds)

## NRZ-L

## Sender

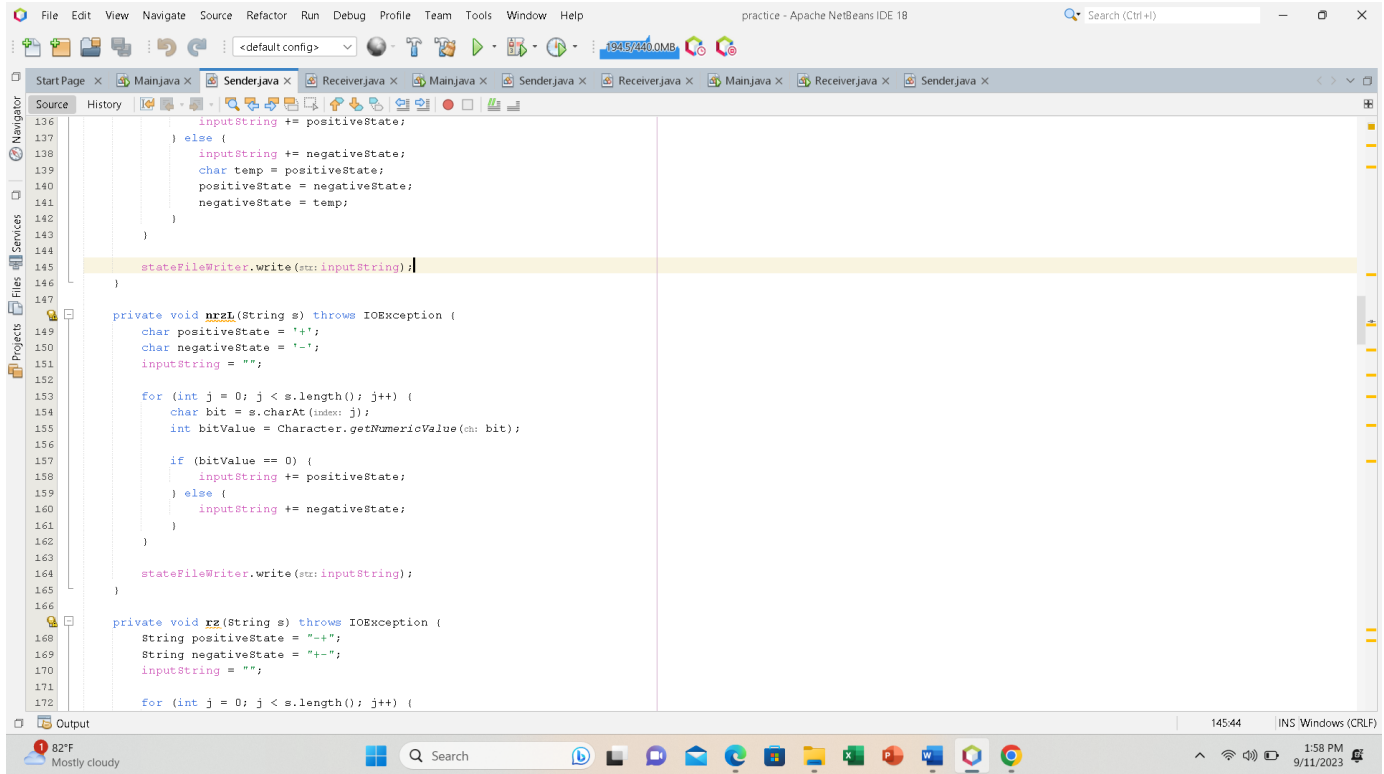
```
private void nrzL(String s) throws IOException {
    char positiveState = '+';
    char negativeState = '-';
    inputString = "";

    for (int j = 0; j < s.length(); j++) {
        char bit = s.charAt(j);
        int bitValue = Character.getNumericValue(bit);

        if (bitValue == 0) {
            inputString += positiveState;
        } else {
            inputString += negativeState;
        }
    }

    stateFileWriter.write(inputString);
}
```





## **Receiver**

```
private void nrzL() throws IOException {
    char state = '0';
    char antiState = '1';
    BufferedReader brr = new BufferedReader(frr);
    int i = 0;
    inputString = "";

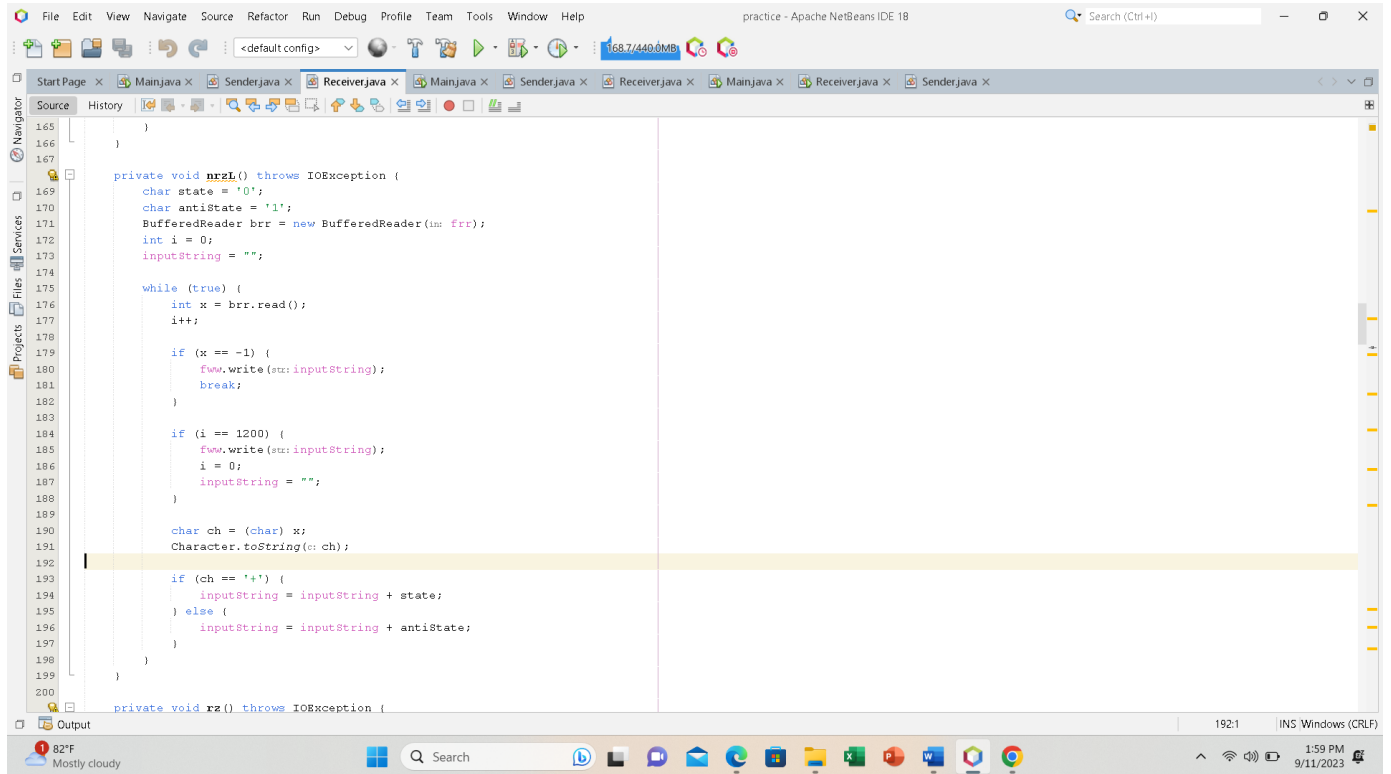
    while (true) {
        int x = brr.read();
        i++;

        if (x == -1) {
            fww.write(inputString);
            break;
        }

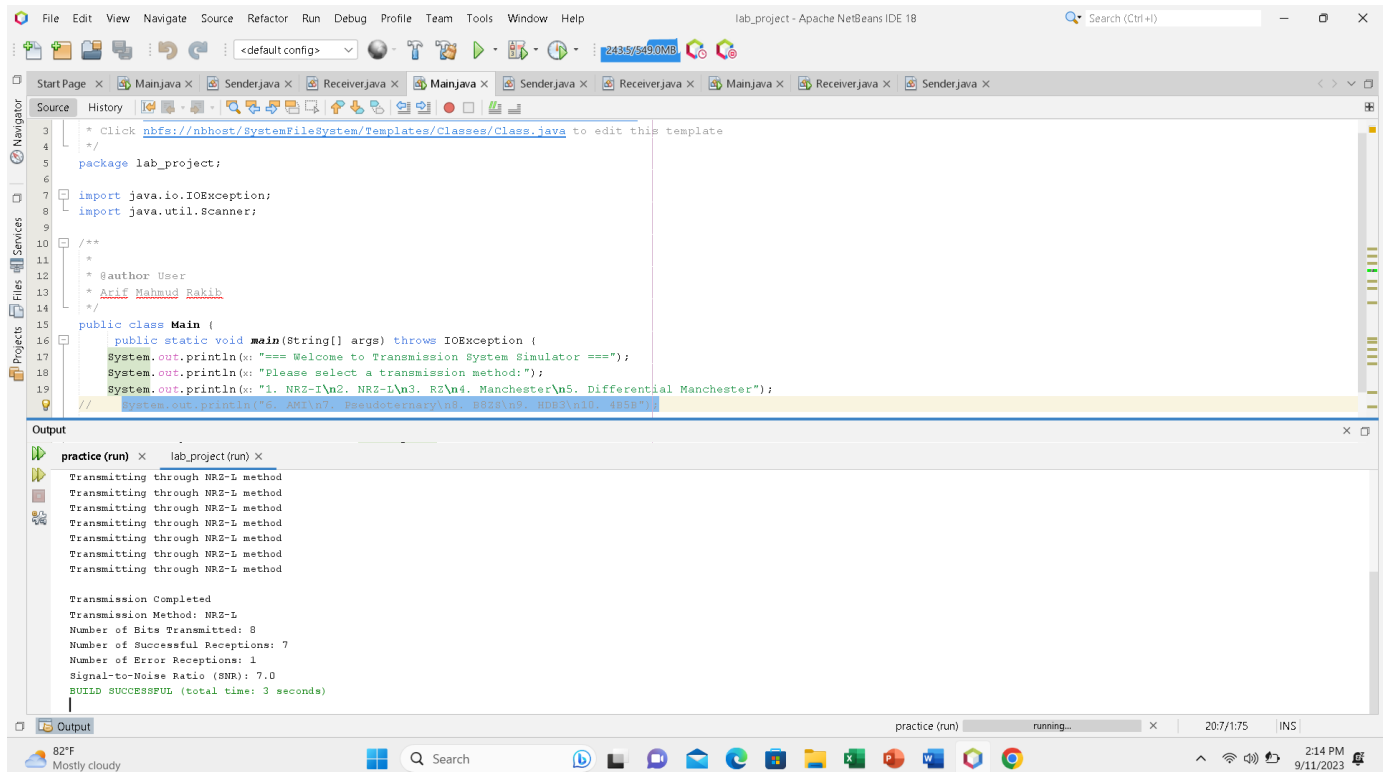
        if (i == 1200) {
            fww.write(inputString);
            i = 0;
            inputString = "";
        }

        char ch = (char) x;
        Character.toString(ch);

        if (ch == '+') {
            inputString = inputString + state;
        } else {
            inputString = inputString + antiState;
        }
    }
}
```



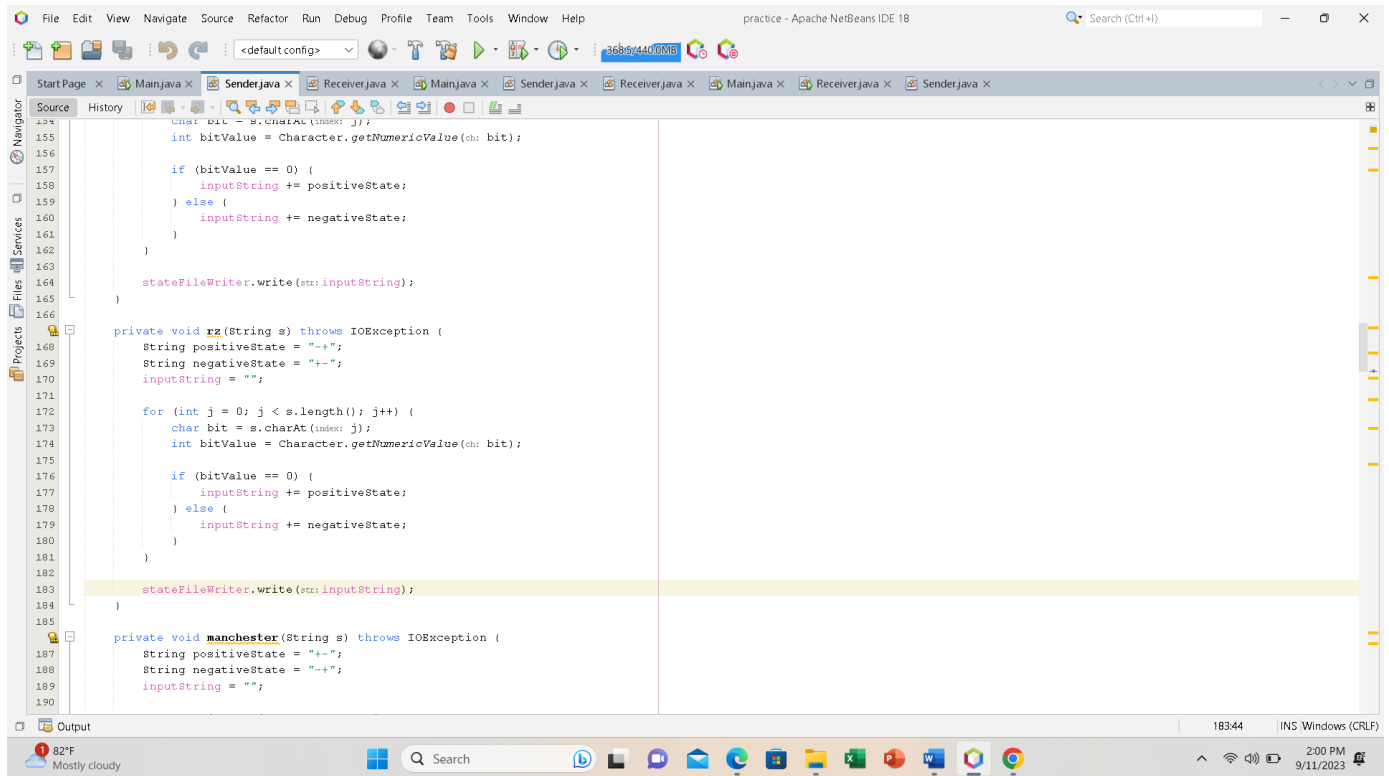
## Output NRZ-L:



## RZ

### Sender

```
private void rz(String s) throws IOException {  
    String positiveState = "-+";  
    String negativeState = "+-";  
    inputString = "";  
  
    for (int j = 0; j < s.length(); j++) {  
        char bit = s.charAt(j);  
        int bitValue = Character.getNumericValue(bit);  
  
        if (bitValue == 0) {  
            inputString += positiveState;  
        } else {  
            inputString += negativeState;  
        }  
    }  
  
    stateFileWriter.write(inputString);  
}
```



## Receiver

```
private void rz() throws IOException {
    char state = '0';
    char antiState = '1';
    BufferedReader brr = new BufferedReader(frr);
    int i = 0;
    inputString = "";

    while (true) {
        int x = brr.read();
        i++;

        if (x == -1) {
            fww.write(inputString);
            break;
        }

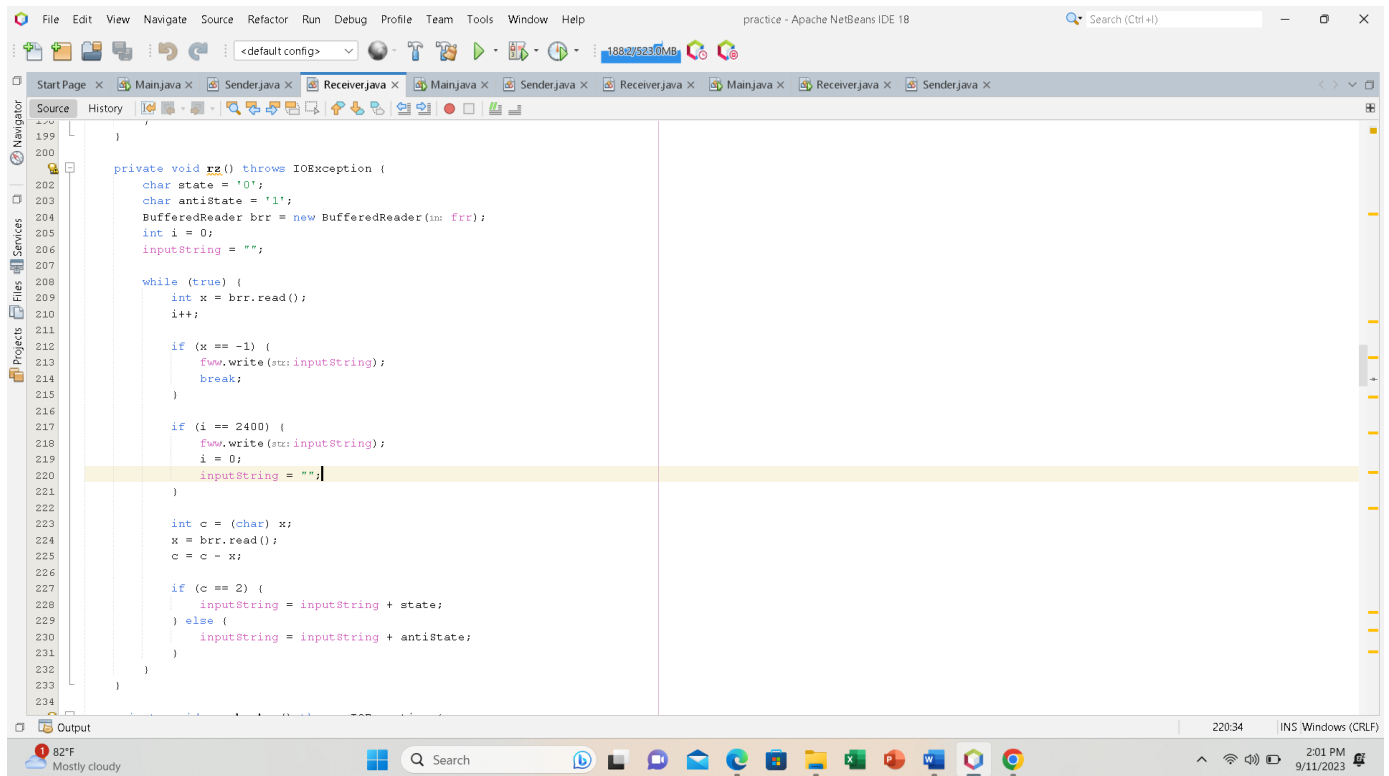
        if (i == 2400) {
            fww.write(inputString);
            i = 0;
            inputString = "";
        }

        int c = (char) x;
        x = brr.read();
    }
}
```

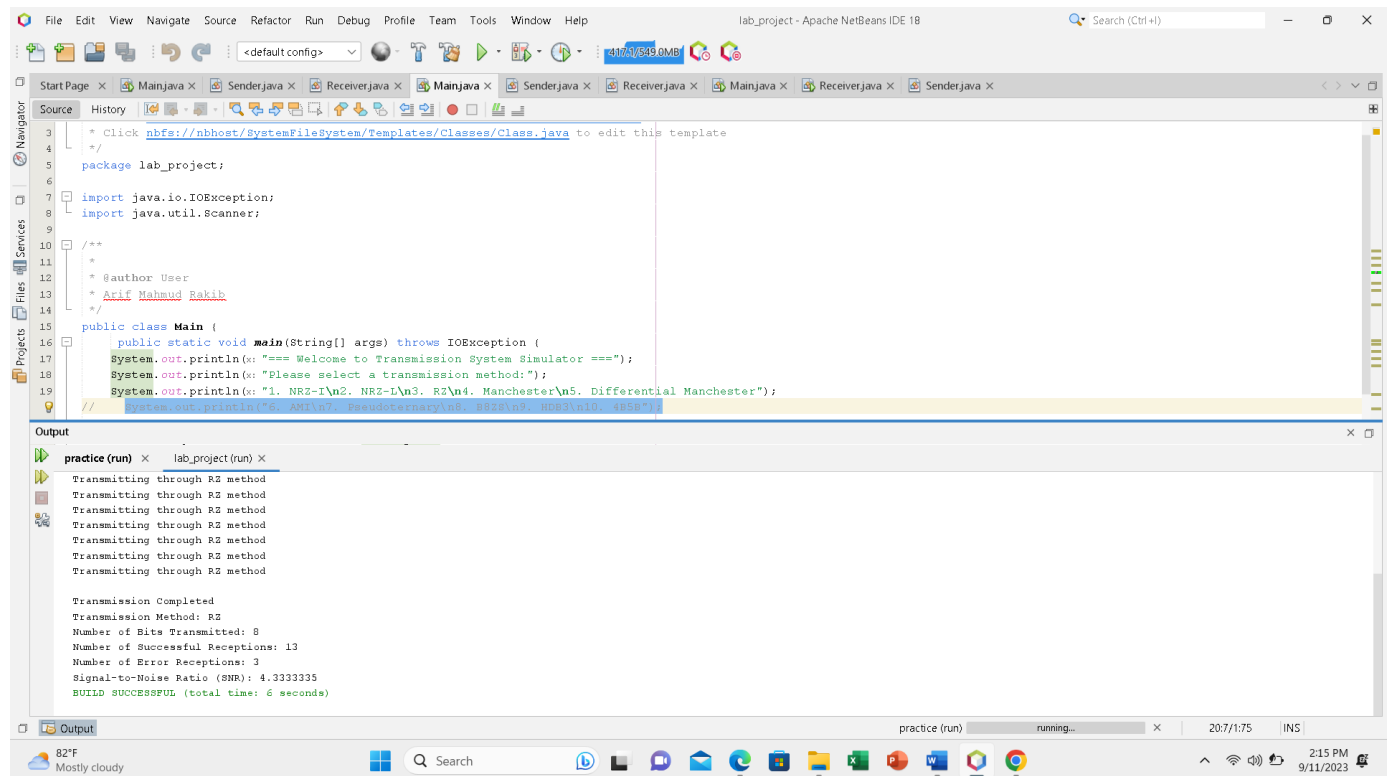
```
c = c - x;
```

```
if (c == 2) {  
    inputString = inputString + state;  
} else {  
    inputString = inputString + antiState;  
}  
}
```

```
}
```



## Output NRZ-L:



```
3  * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
4  */
5  package lab_project;
6
7  import java.io.IOException;
8  import java.util.Scanner;
9
10 /**
11  * @author User
12  * @Arif Mahmud Rakib
13  */
14
15 public class Main {
16     public static void main(String[] args) throws IOException {
17         System.out.println("=== Welcome to Transmission System Simulator ===");
18         System.out.println("Please select a transmission method:");
19         System.out.println("1. NRZ-I\n2. NRZ-L\n3. NRZ\n4. Manchester\n5. Differential Manchester");
20         // System.out.println("6. AM\n7. Pseudoternary\n8. RZ\n9. HD\n10. BPSK");
21     }
22 }
```

practice (run) x lab\_project (run) x

Transmitting through RZ method  
Transmitting through RZ method  
Transmitting through RZ method  
Transmitting through RZ method  
Transmitting through RZ method  
Transmitting through RZ method  
Transmitting through RZ method

Transmission Completed  
Transmission Method: RZ  
Number of Bits Transmitted: 8  
Number of Successful Receptions: 13  
Number of Error Receptions: 3  
Signal-to-Noise Ratio (SNR): 4.3333335  
BUILD SUCCESSFUL (total time: 6 seconds)

## Manchester:

### Sender:

```
private void manchester(String s) throws IOException {
```

```
    String positiveState = "+-";
```

```
    String negativeState = "-+";
```

```
    inputString = "";
```

```
    for (int j = 0; j < s.length(); j++) {
```

```
        char bit = s.charAt(j);
```

```
        int bitValue = Character.getNumericValue(bit);
```

```
        if (bitValue == 0) {
```

```
            inputString += positiveState;
```

```
        } else {
```

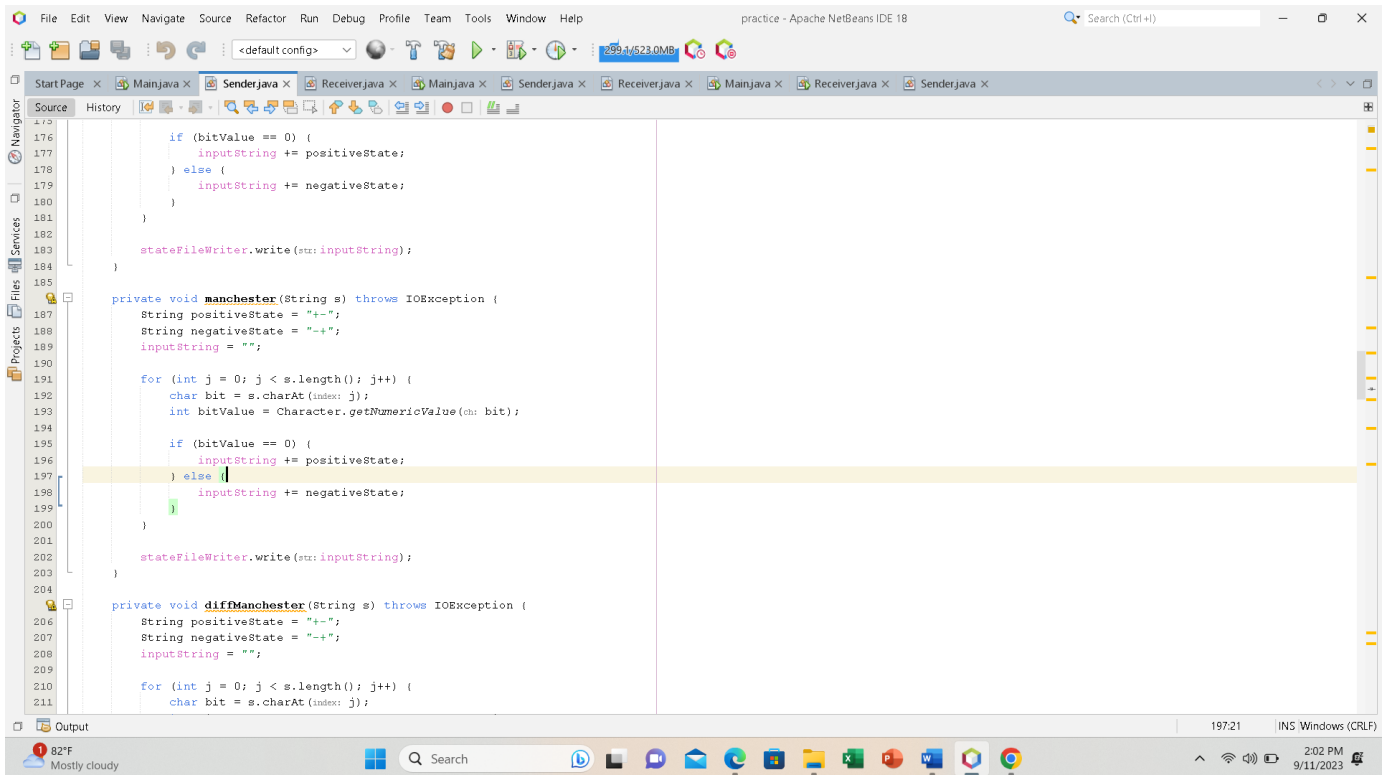
```
            inputString += negativeState;
```

```
        }
```

```
}
```

```
stateFileWriter.write(inputString);
```

```
}
```



## Receiver:

```
private void manchester() throws IOException {  
    char state = '0';  
    char antiState = '1';  
    BufferedReader brr = new BufferedReader(frr);  
    int i = 0;  
    inputString = "";  
  
    while (true) {  
        int x = brr.read();  
        i++;  
  
        if (x == -1) {  
            fww.write(inputString);  
        }  
    }  
}
```

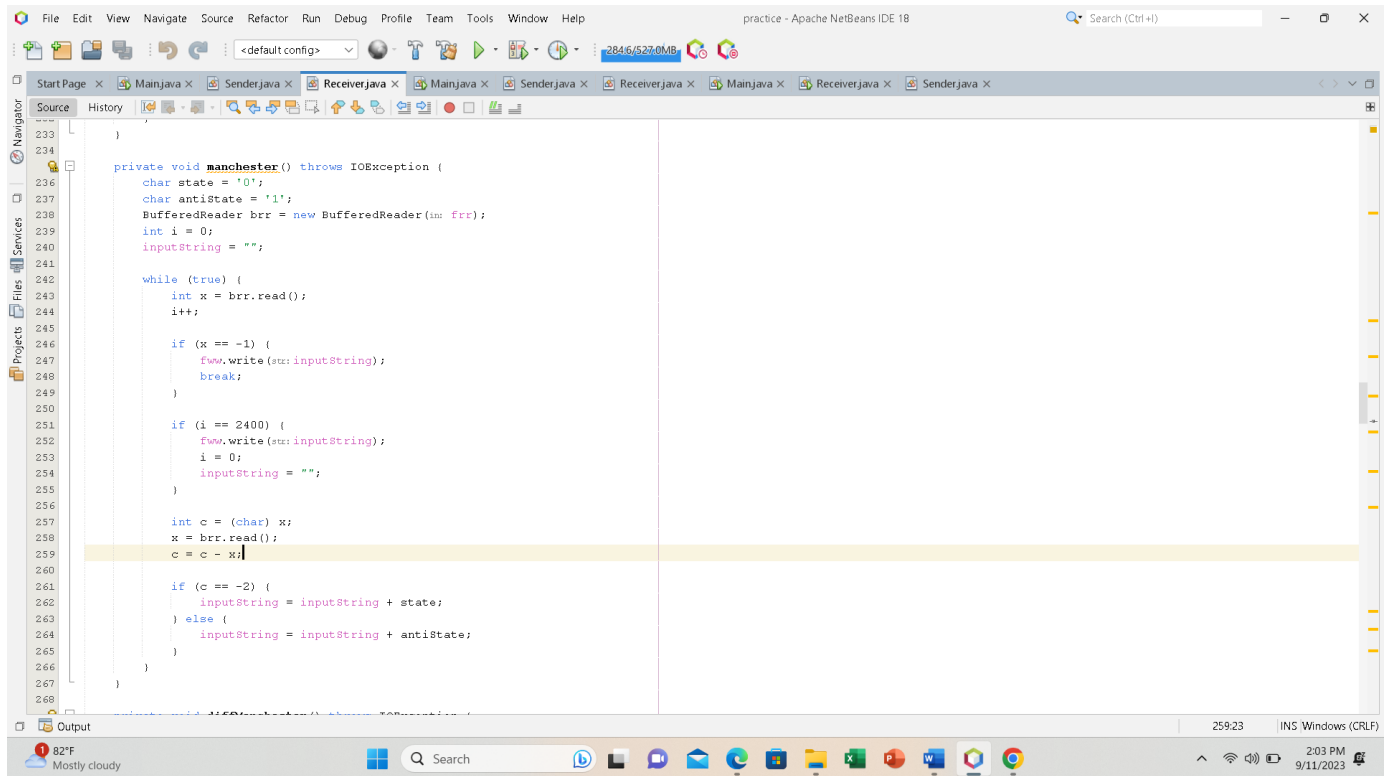


```
        break;
    }

    if (i == 2400) {
        fww.write(inputString);
        i = 0;
        inputString = "";
    }

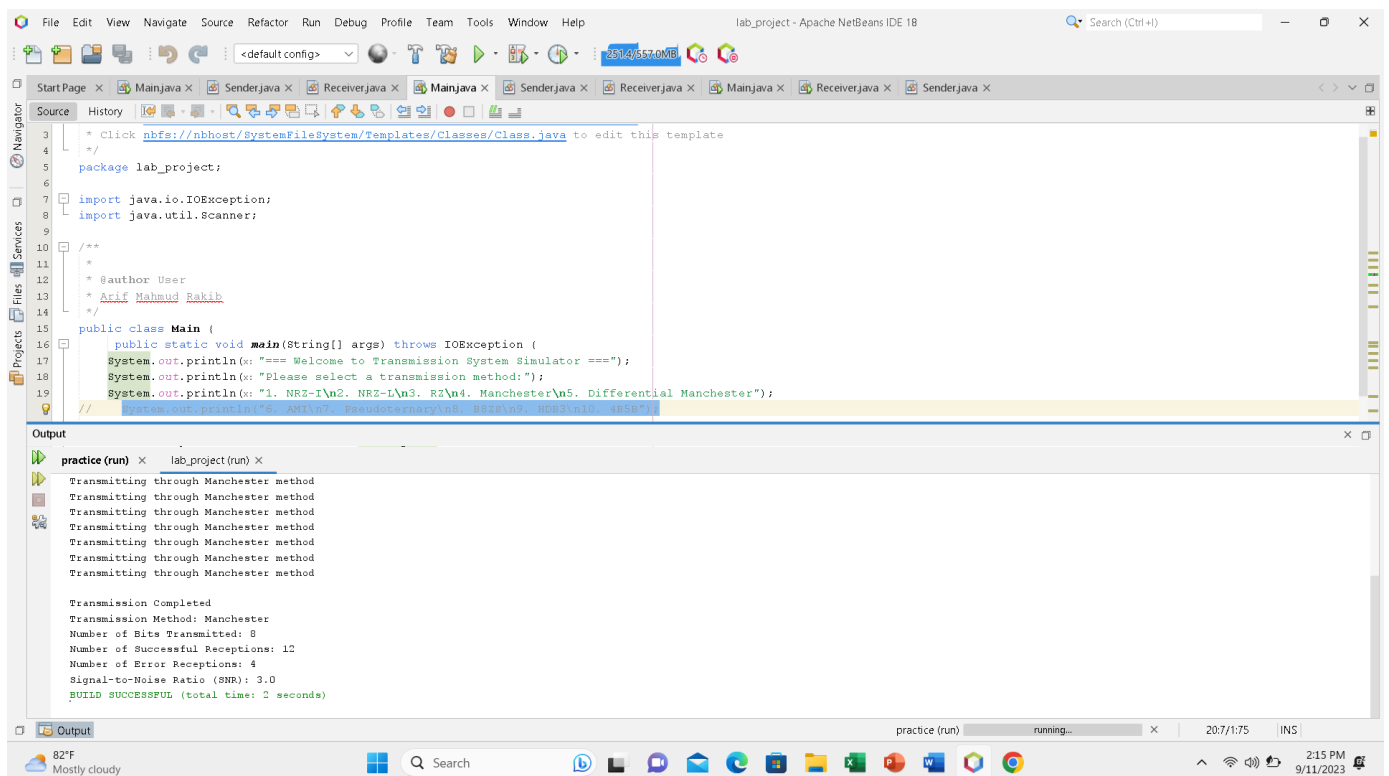
    int c = (char) x;
    x = brr.read();
    c = c - x;

    if (c == -2) {
        inputString = inputString + state;
    } else {
        inputString = inputString + antiState;
    }
}
}
```



```
233 }
234
235 private void manchester() throws IOException {
236     char state = '0';
237     char antiState = '1';
238     BufferedReader brr = new BufferedReader(new FileReader("Receiver.txt"));
239     int i = 0;
240     inputString = "";
241
242     while (true) {
243         int x = brr.read();
244         i++;
245
246         if (x == -1) {
247             fww.write(state + inputString);
248             break;
249         }
250
251         if (i == 2400) {
252             fww.write(state + inputString);
253             i = 0;
254             inputString = "";
255         }
256
257         int c = (char) x;
258         x = brr.read();
259         c = c - x;
260
261         if (c == -2) {
262             inputString = inputString + state;
263         } else {
264             inputString = inputString + antiState;
265         }
266     }
267 }
```

## Manchester Output :



```
3 4
5 package lab_project;
6
7 import java.io.IOException;
8 import java.util.Scanner;
9
10 /**
11  * @author User
12  * Arif Mahmud Rakib
13  */
14
15 public class Main {
16     public static void main(String[] args) throws IOException {
17         System.out.println("=== Welcome to Transmission System Simulator ===");
18         System.out.println("Please select a transmission method:");
19         System.out.println("1. NRZ-L\n2. NRZ-I\n3. RS\n4. Manchester\n5. Differential Manchester");
20         // System.out.println("6. AM\n7. PSK\n8. QAM\n9. BPSK\n10. FSK\n11. LFSR");
21     }
22 }
```

practice (run) × lab\_project (run) ×

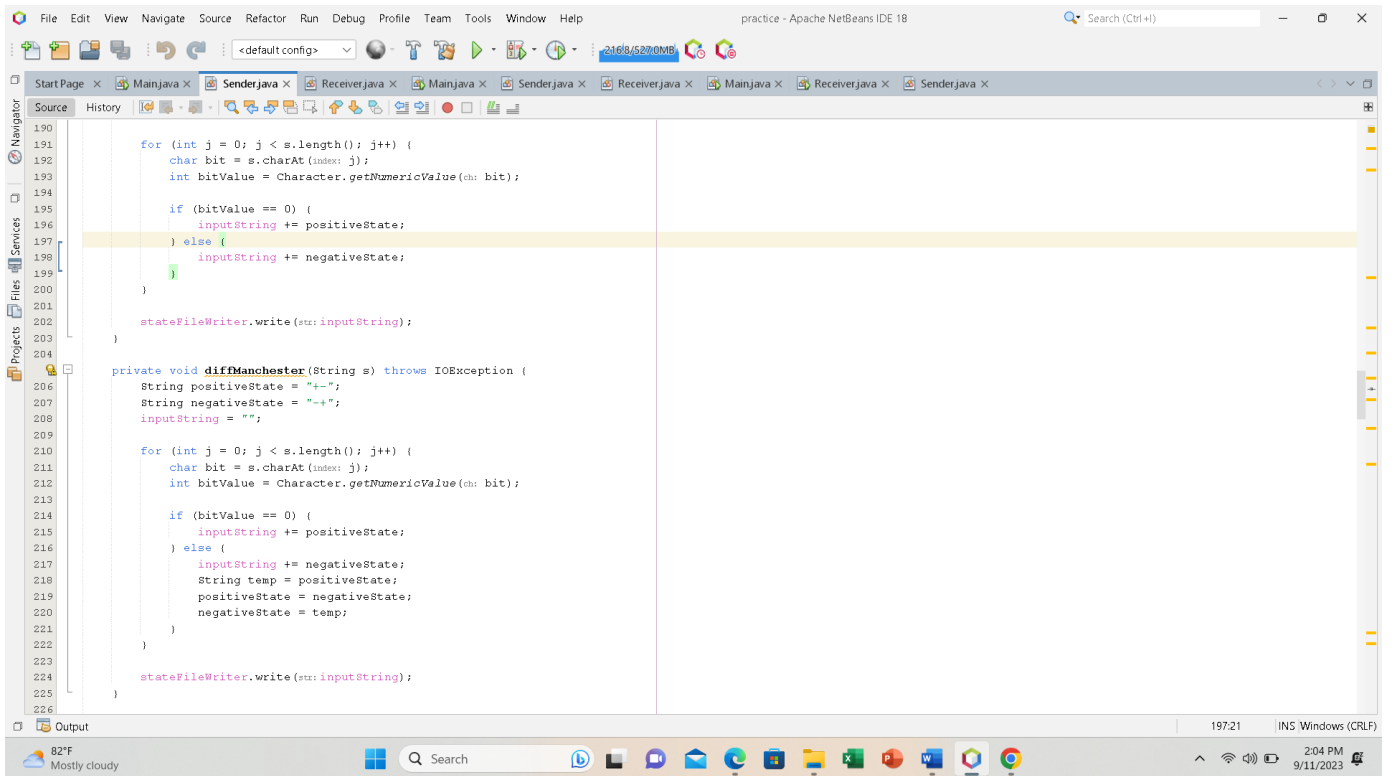
Transmitting through Manchester method  
Transmitting through Manchester method  
Transmitting through Manchester method  
Transmitting through Manchester method  
Transmitting through Manchester method  
Transmitting through Manchester method  
Transmitting through Manchester method

Transmission Completed  
Transmission Method: Manchester  
Number of Bits Transmitted: 0  
Number of Successful Receptions: 12  
Number of Error Receptions: 4  
Signal-to-Noise Ratio (SNR): 3.0  
BUILD SUCCESSFUL (total time: 2 seconds)

## **Differential Manchester:**

### **Sender:**

```
private void diffManchester(String s) throws IOException {  
    String positiveState = "+-";  
    String negativeState = "-+";  
    inputString = "";  
  
    for (int j = 0; j < s.length(); j++) {  
        char bit = s.charAt(j);  
        int bitValue = Character.getNumericValue(bit);  
  
        if (bitValue == 0) {  
            inputString += positiveState;  
        } else {  
            inputString += negativeState;  
            String temp = positiveState;  
            positiveState = negativeState;  
            negativeState = temp;  
        }  
    }  
  
    stateFileWriter.write(inputString);  
}
```



**Receiver:**

```
private void diffManchester() throws IOException {
    int prevState = -2;
    char state = '0';
    char antiState = '1';
    BufferedReader brr = new BufferedReader(frr);
    int i = 0;
    inputString = "";

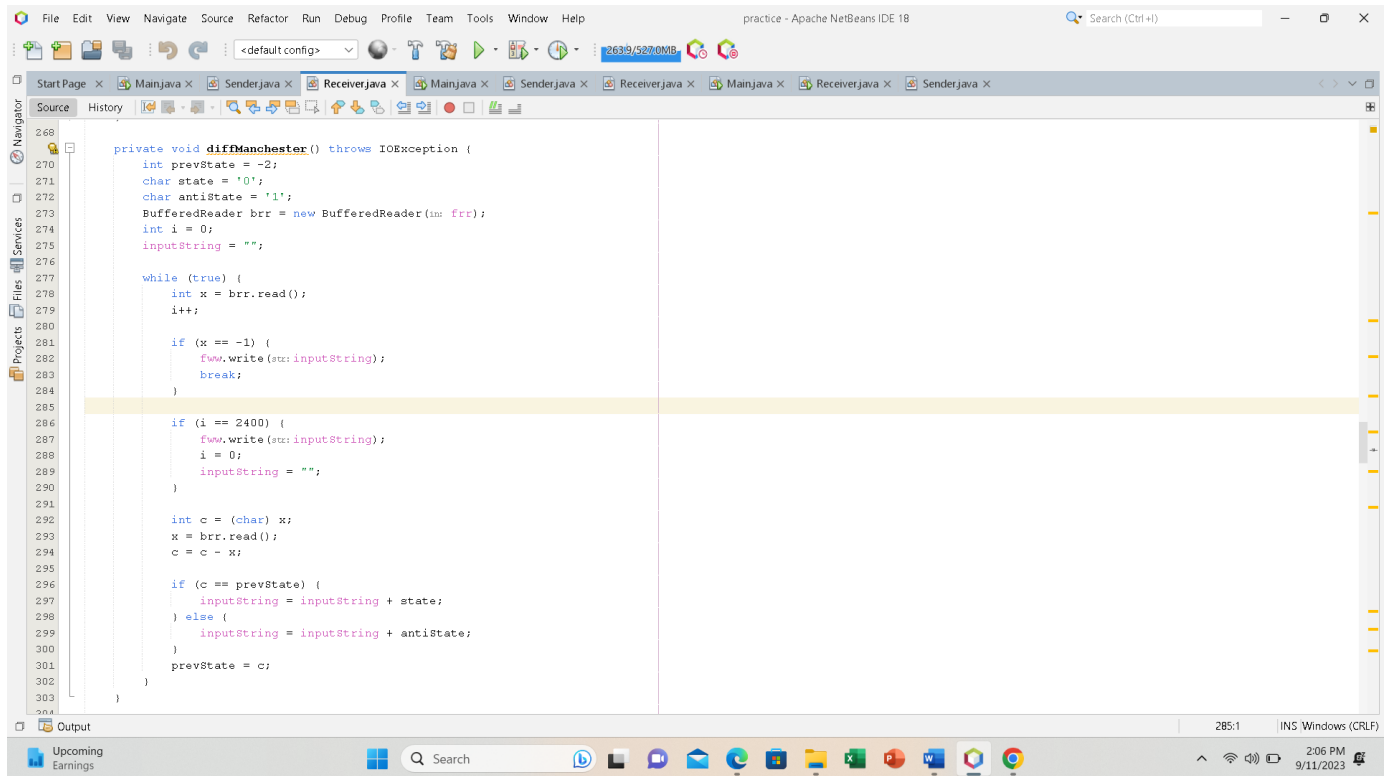
    while (true) {
        int x = brr.read();
        i++;

        if (x == -1) {
            fww.write(inputString);
            break;
        }

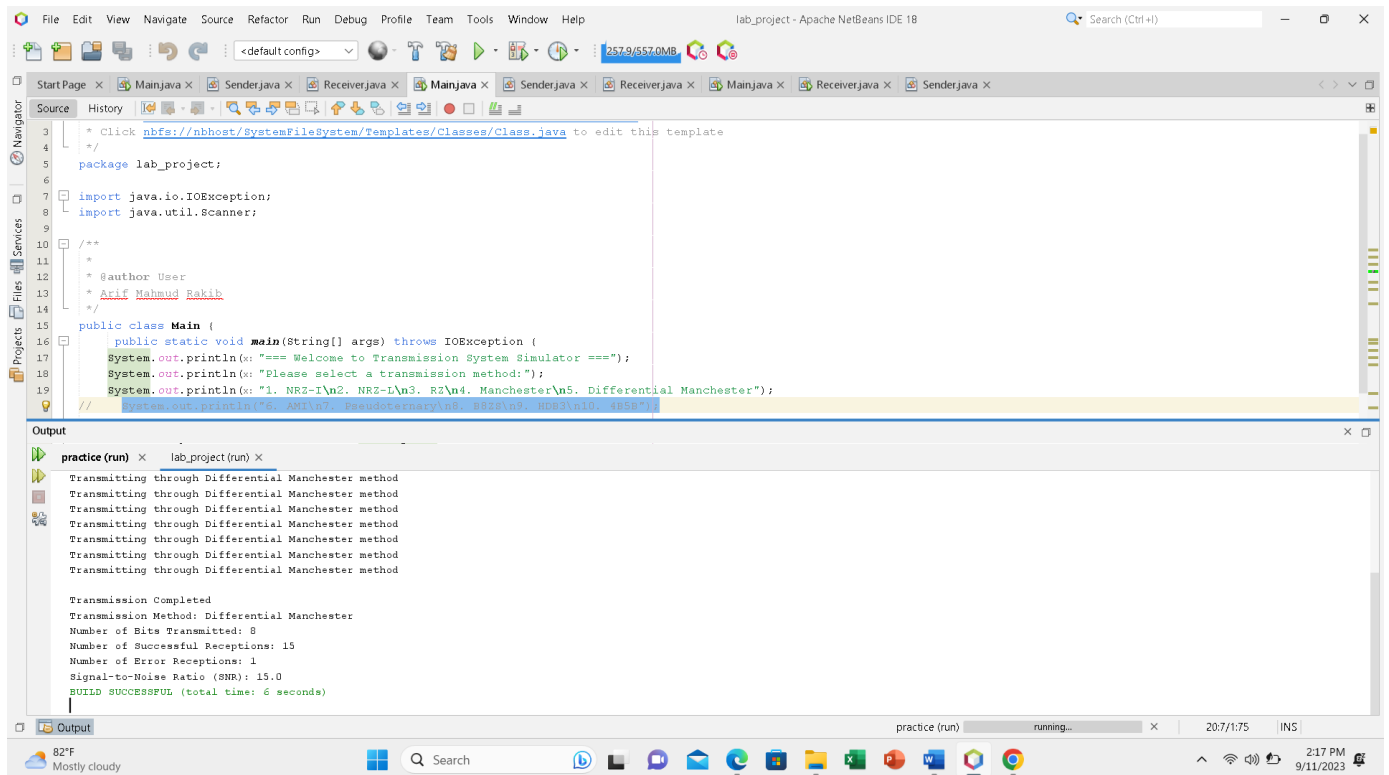
        if (i == 2400) {
            fww.write(inputString);
            i = 0;
            inputString = "";
        }

        int c = (char) x;
        x = brr.read();
        c = c - x;

        if (c == prevState) {
            inputString = inputString + state;
        } else {
            inputString = inputString + antiState;
        }
        prevState = c;
    }
}
```



## Output Differential Manchester:



## Discussion :

The provided content serves as a concise guide to different line coding techniques and their implementation. It showcases the practical aspects of encoding and decoding digital signals, highlighting the steps and coding involved in each technique. This lab task offers us hands-on experience in understanding how different encoding schemes impact data transmission and synchronization. It also underscores the significance of proper encoding and decoding for reliable communication in digital systems.















