**Assignment 1**

## Name : Shrinivas Hatyalikar

## Div: TY-B (B2)

## Roll No: 26

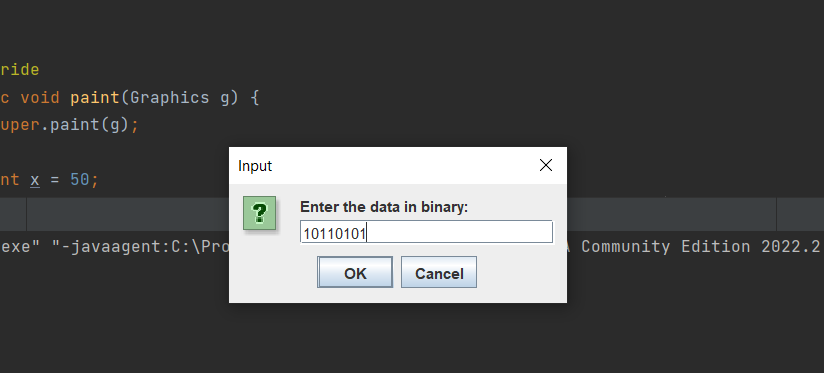
Q) Write a program in C++/JAVA to implement - Unipolar, NRZ Level, NRZ Inverted, Bipolar Encoding, Manchester Encoding, and Differential Manchester Encoding.

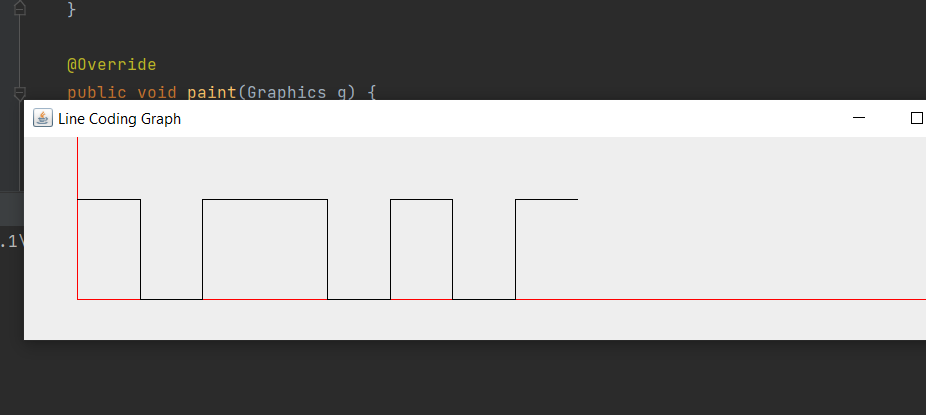
1. Unipolar

Code:

package javaprog;  
  
import javax.swing.\*;  
import java.awt.\*;  
  
public class Unipolar extends JFrame {  
 private String data;  
  
 public Unipolar(String data) {  
 this.data = data;  
  
 setTitle("Line Coding Graph");  
 setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
 setSize(800, 200);  
 setLocationRelativeTo(null);  
 }  
  
 @Override  
 public void paint(Graphics g) {  
 super.paint(g);  
  
 int x = 50;  
 int y;  
 int lineWidth = 50;  
 g.setColor(Color.*RED*);  
 g.drawLine(50, 0, 50, 160);  
 g.drawLine(50, 160, 800, 160);  
 g.setColor(Color.*BLACK*);  
 char m = data.charAt(0);  
 for (char bit : data.toCharArray()) {  
 if (bit == '0') {  
 if (bit == m) {  
 y = 160;  
 g.drawLine(x, y, x + lineWidth, y);  
 x += lineWidth;  
 }  
 else{  
 g.drawLine(x, 80, x, 160);  
 y = 160;  
 g.drawLine(x, y, x + lineWidth, y);  
 x += lineWidth;  
 m = bit;  
 }  
 } else if (bit == '1') {  
 if (bit == m) {  
 y = 80;  
 g.drawLine(x, y, x + lineWidth, y);  
 x += lineWidth;  
 }  
 else{  
 g.drawLine(x, 80, x, 160);  
 y = 80;  
 g.drawLine(x, y, x + lineWidth, y);  
 x += lineWidth;  
 m = bit;  
 }  
 }  
 }  
 }  
  
 public static void main(String[] args) {  
 SwingUtilities.*invokeLater*(() -> {  
 String data = JOptionPane.*showInputDialog*("Enter the data in binary:");  
  
 Unipolar obj = new Unipolar(data);  
 obj.setVisible(true);  
 });  
 }  
}

Output:



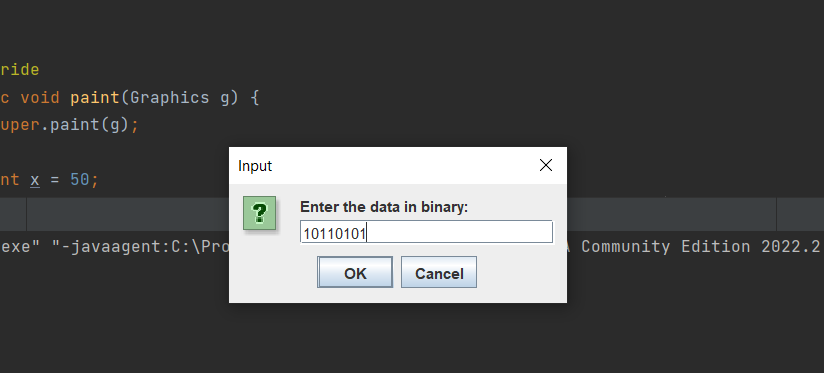


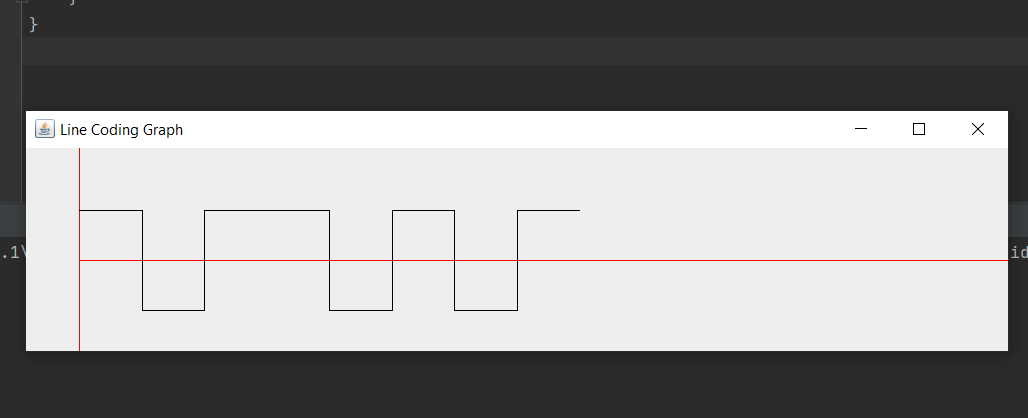
1. NRZ Level

Code:

import java.awt.Color;  
import java.awt.Component;  
import java.awt.Graphics;  
import javax.swing.JFrame;  
import javax.swing.JOptionPane;  
import javax.swing.SwingUtilities;  
  
public class NRZL extends JFrame {  
 private String data;  
  
 public NRZL(String data) {  
 this.data = data;  
 this.setTitle("Line Coding Graph");  
 this.setDefaultCloseOperation(3);  
 this.setSize(800, 200);  
 this.setLocationRelativeTo((Component)null);  
 }  
  
 public void paint(Graphics g) {  
 super.paint(g);  
 int x = 50;  
 int lineWidth = 50;  
 g.setColor(Color.*RED*);  
 g.drawLine(50, 0, 50, 200);  
 g.drawLine(50, 120, 800, 120);  
 g.setColor(Color.*BLACK*);  
 char m = this.data.charAt(0);  
 char[] var6 = this.data.toCharArray();  
 int var7 = var6.length;  
  
 for(int var8 = 0; var8 < var7; ++var8) {  
 char bit = var6[var8];  
 if (bit == '0') {  
 short y;  
 if (bit == m) {  
 y = 160;  
 g.drawLine(x, y, x + lineWidth, y);  
 x += lineWidth;  
 } else {  
 g.drawLine(x, 80, x, 160);  
 y = 160;  
 g.drawLine(x, y, x + lineWidth, y);  
 x += lineWidth;  
 m = bit;  
 }  
 } else if (bit == '1') {  
 byte y;  
 if (bit == m) {  
 y = 80;  
 g.drawLine(x, y, x + lineWidth, y);  
 x += lineWidth;  
 } else {  
 g.drawLine(x, 80, x, 160);  
 y = 80;  
 g.drawLine(x, y, x + lineWidth, y);  
 x += lineWidth;  
 m = bit;  
 }  
 }  
 }  
  
 }  
  
 public static void main(String[] args) {  
 SwingUtilities.*invokeLater*(() -> {  
 String data = JOptionPane.*showInputDialog*("Enter the data elements (0s and 1s):");  
 NRZL obj = new NRZL(data);  
 obj.setVisible(true);  
 });  
 }  
}

Output:



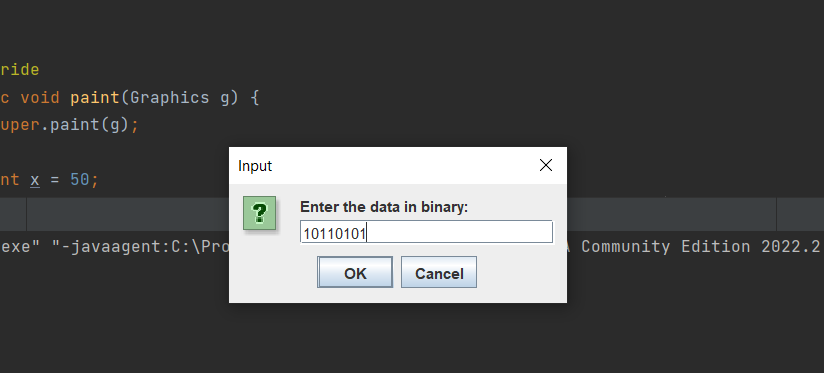


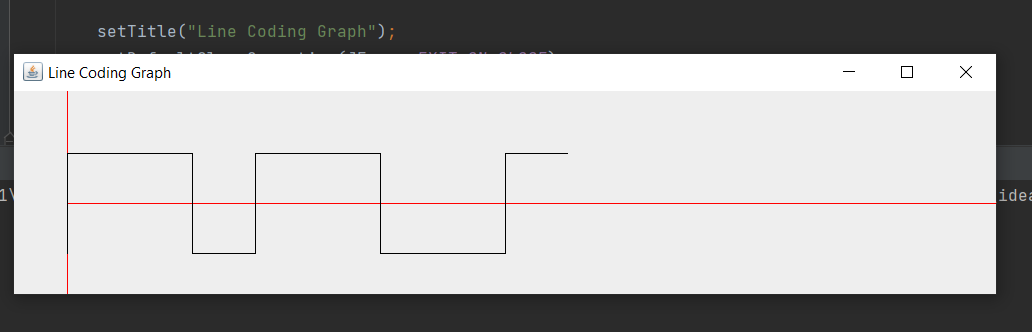
3)NRZ Inverted

Code:

package javaprog;  
  
import javax.swing.\*;  
import java.awt.\*;  
  
public class NRZI extends JFrame {  
 private String data;  
  
 public NRZI(String data) {  
 this.data = data;  
  
 setTitle("Line Coding Graph");  
 setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
 setSize(800, 200);  
 setLocationRelativeTo(null);  
 }  
  
 @Override  
 public void paint(Graphics g) {  
 super.paint(g);  
  
 int x = 50;  
 int y;  
 int lineWidth = 50;  
 g.setColor(Color.*RED*);  
 g.drawLine(50, 0, 50, 200);  
 g.drawLine(50, 120, 800, 120);  
 g.setColor(Color.*BLACK*);  
 int m = 0;  
 for (char bit : data.toCharArray()) {  
 if (bit == '1') {  
 if (m == 1) {  
 m = 0;  
 } else {  
 m = 1;   
 }  
 }  
 if (bit == '1') {  
 g.drawLine(x, 80, x, 160);  
 }  
 if (m == 0) {  
 y = 160;  
 g.drawLine(x, y, x + lineWidth, y);  
 x += lineWidth;  
 }  
 else{  
 y = 80;  
 g.drawLine(x, y, x + lineWidth, y);  
 x += lineWidth;  
 }  
 }  
 }  
  
 public static void main(String[] args) {  
 SwingUtilities.*invokeLater*(() -> {  
 String data = JOptionPane.*showInputDialog*("Enter the data elements (0s and 1s):");  
  
 NRZI obj = new NRZI(data);  
 obj.setVisible(true);  
 });  
 }  
}

Output:



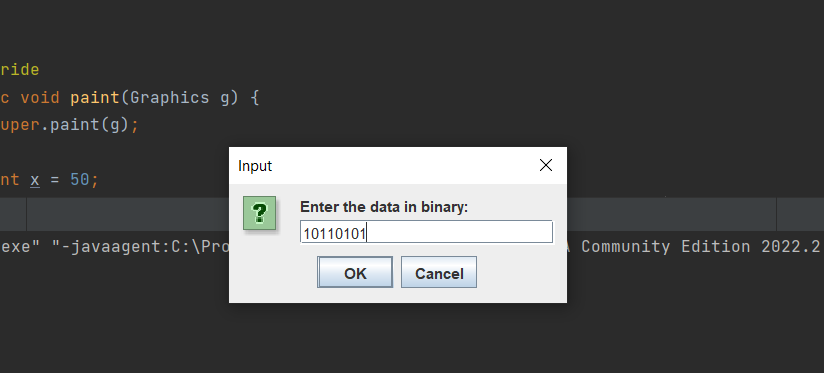


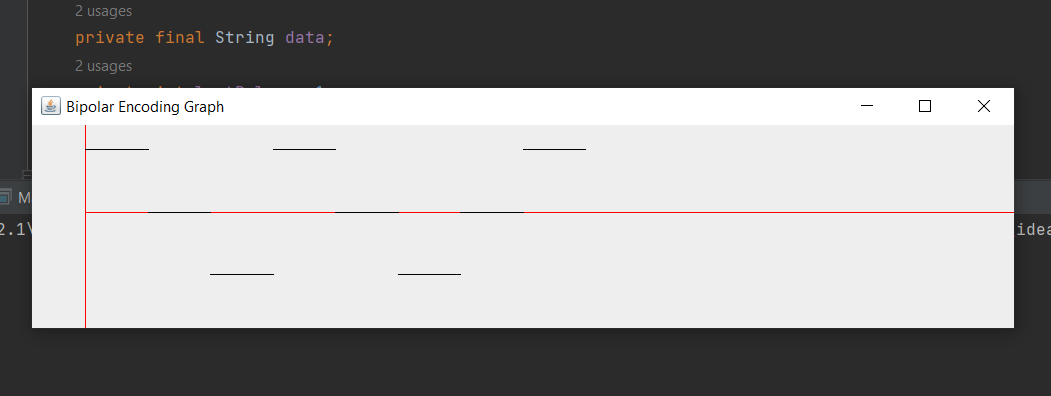
4)Bipolar

Code:

package javaprog;  
  
import javax.swing.\*;  
import java.awt.\*;  
  
public class Bipolar extends JFrame {  
 private final String data;  
 private int lastPulse = 1;  
  
 public Bipolar(String data) {  
 this.data = data;  
  
 setTitle("Bipolar Encoding Graph");  
 setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
 setSize(800, 200);  
 setLocationRelativeTo(null);  
 }  
  
 @Override  
 public void paint(Graphics g) {  
 super.paint(g);  
  
 int x = 50;  
 int y;  
 int lineWidth = 50;  
 g.setColor(Color.*RED*);  
 g.drawLine(50, 0, 50, 200);  
 g.drawLine(50, 100, 800, 100); // Central line  
 g.setColor(Color.*BLACK*);  
  
 for (char bit : data.toCharArray()) {  
 if (bit == '0') {  
 y = 100;  
 g.drawLine(x, y, x + lineWidth, y);  
 x += lineWidth;  
 } else if (bit == '1') {  
 y = (lastPulse == 1) ? 50 : 150;  
 g.drawLine(x, y, x + lineWidth, y);  
 lastPulse = (y == 50) ? -1 : 1;  
 x += lineWidth;  
 }  
 }  
 }  
  
 public static void main(String[] args) {  
 SwingUtilities.*invokeLater*(() -> {  
 String data = JOptionPane.*showInputDialog*("Enter the data in binary:");  
  
 if (data.matches("[01]+")) { // Check if the input is binary  
 Bipolar obj = new Bipolar(data);  
 obj.setVisible(true);  
 } else {  
 JOptionPane.*showMessageDialog*(null, "Please enter a valid binary sequence.");  
 }  
 });  
 }  
}

Output:



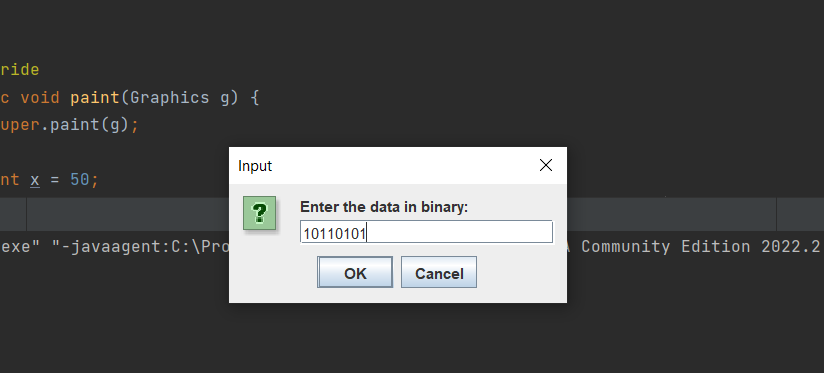


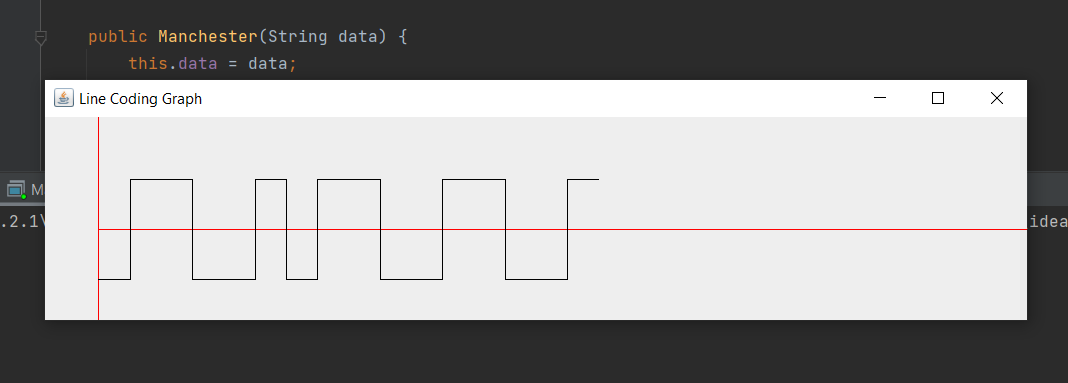
5)Manchester

Code:

import java.awt.Color;  
import java.awt.Component;  
import java.awt.Graphics;  
import javax.swing.JFrame;  
import javax.swing.JOptionPane;  
import javax.swing.SwingUtilities;  
  
public class Manchester extends JFrame {  
 private String data;  
  
 public Manchester(String data) {  
 this.data = data;  
 this.setTitle("Line Coding Graph");  
 this.setDefaultCloseOperation(3);  
 this.setSize(800, 200);  
 this.setLocationRelativeTo((Component)null);  
 }  
  
 public void paint(Graphics g) {  
 super.paint(g);  
 int x = 50;  
 int lineWidth = 50;  
 g.setColor(Color.*RED*);  
 g.drawLine(50, 0, 50, 200);  
 g.drawLine(50, 120, 800, 120);  
 g.setColor(Color.*BLACK*);  
 char m = this.data.charAt(0);  
 if (m == '0') {  
 m = '1';  
 } else {  
 m = '0';  
 }  
  
 char[] var6 = this.data.toCharArray();  
 int var7 = var6.length;  
  
 for(int var8 = 0; var8 < var7; ++var8) {  
 char bit = var6[var8];  
 short y;  
 byte y;  
 if (bit == '0') {  
 if (bit != m) {  
 y = 80;  
 g.drawLine(x, y, x + lineWidth / 2, y);  
 x += lineWidth / 2;  
 g.drawLine(x, 80, x, 160);  
 y = 160;  
 g.drawLine(x, y, x + lineWidth / 2, y);  
 x += lineWidth / 2;  
 m = bit;  
 } else {  
 g.drawLine(x, 80, x, 160);  
 y = 80;  
 g.drawLine(x, y, x + lineWidth / 2, y);  
 x += lineWidth / 2;  
 g.drawLine(x, 80, x, 160);  
 y = 160;  
 g.drawLine(x, y, x + lineWidth / 2, y);  
 x += lineWidth / 2;  
 }  
 } else if (bit == '1') {  
 if (bit != m) {  
 y = 160;  
 g.drawLine(x, y, x + lineWidth / 2, y);  
 x += lineWidth / 2;  
 g.drawLine(x, 80, x, 160);  
 y = 80;  
 g.drawLine(x, y, x + lineWidth / 2, y);  
 x += lineWidth / 2;  
 m = bit;  
 } else {  
 g.drawLine(x, 80, x, 160);  
 y = 160;  
 g.drawLine(x, y, x + lineWidth / 2, y);  
 x += lineWidth / 2;  
 g.drawLine(x, 80, x, 160);  
 y = 80;  
 g.drawLine(x, y, x + lineWidth / 2, y);  
 x += lineWidth / 2;  
 }  
 }  
 }  
  
 }  
  
 public static void main(String[] args) {  
 SwingUtilities.*invokeLater*(() -> {  
 String data = JOptionPane.*showInputDialog*("Enter the data elements (0s and 1s):");  
 Manchester obj = new Manchester(data);  
 obj.setVisible(true);  
 });  
 }  
}

Output:





6) Differential Manchester

Code:

import java.awt.Color;  
import java.awt.Component;  
import java.awt.Graphics;  
import javax.swing.JFrame;  
import javax.swing.JOptionPane;  
import javax.swing.SwingUtilities;  
  
public class DiffManchester extends JFrame {  
 private String data;  
  
 public DiffManchester(String data) {  
 this.data = data;  
 this.setTitle("Line Coding Graph");  
 this.setDefaultCloseOperation(3);  
 this.setSize(800, 200);  
 this.setLocationRelativeTo((Component)null);  
 }  
  
 public void paint(Graphics g) {  
 super.paint(g);  
 int x = 50;  
 int lineWidth = 50;  
 g.setColor(Color.*RED*);  
 g.drawLine(50, 0, 50, 200);  
 g.drawLine(50, 120, 800, 120);  
 g.setColor(Color.*BLACK*);  
 int m = false;  
 char[] var6 = this.data.toCharArray();  
 int var7 = var6.length;  
  
 for(int var8 = 0; var8 < var7; ++var8) {  
 char bit = var6[var8];  
 if (bit == '1') {  
 if (m) {  
 m = false;  
 } else {  
 m = true;  
 }  
 }  
  
 if (bit == '0') {  
 g.drawLine(x, 80, x, 160);  
 }  
  
 short y;  
 byte y;  
 if (!m) {  
 y = 160;  
 g.drawLine(x, y, x + lineWidth / 2, y);  
 x += lineWidth / 2;  
 g.drawLine(x, 80, x, 160);  
 y = 80;  
 g.drawLine(x, y, x + lineWidth / 2, y);  
 x += lineWidth / 2;  
 } else {  
 y = 80;  
 g.drawLine(x, y, x + lineWidth / 2, y);  
 x += lineWidth / 2;  
 g.drawLine(x, 80, x, 160);  
 y = 160;  
 g.drawLine(x, y, x + lineWidth / 2, y);  
 x += lineWidth / 2;  
 }  
 }  
  
 }  
  
 public static void main(String[] args) {  
 SwingUtilities.*invokeLater*(() -> {  
 String data = JOptionPane.*showInputDialog*("Enter the data elements (0s and 1s):");  
 DiffManchester diffmanchester = new DiffManchester(data);  
 diffmanchester.setVisible(true);  
 });  
 }  
}

Output:

