COM329 ADVANCED PROGRAMMING

FINAL PROJECT

BATUHAN SATILMIS

B1805.010039

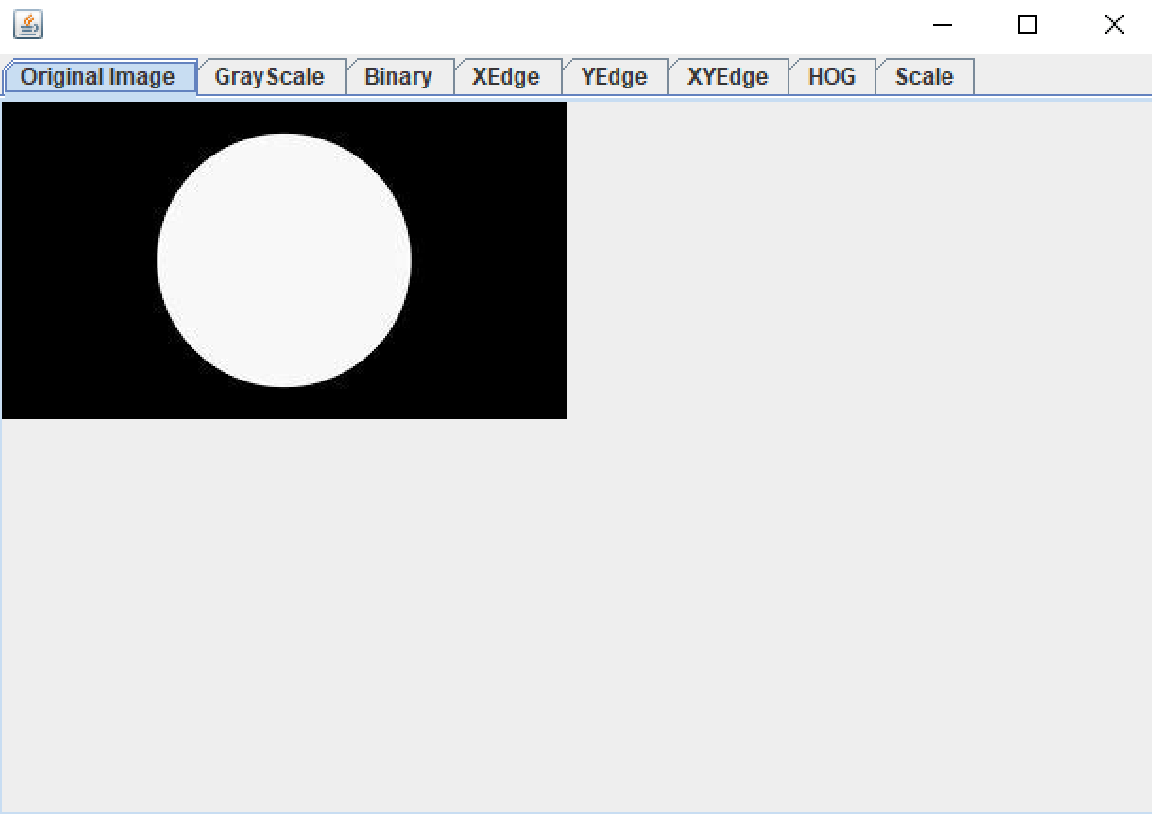
MUHAMMAD RIVALSYAH

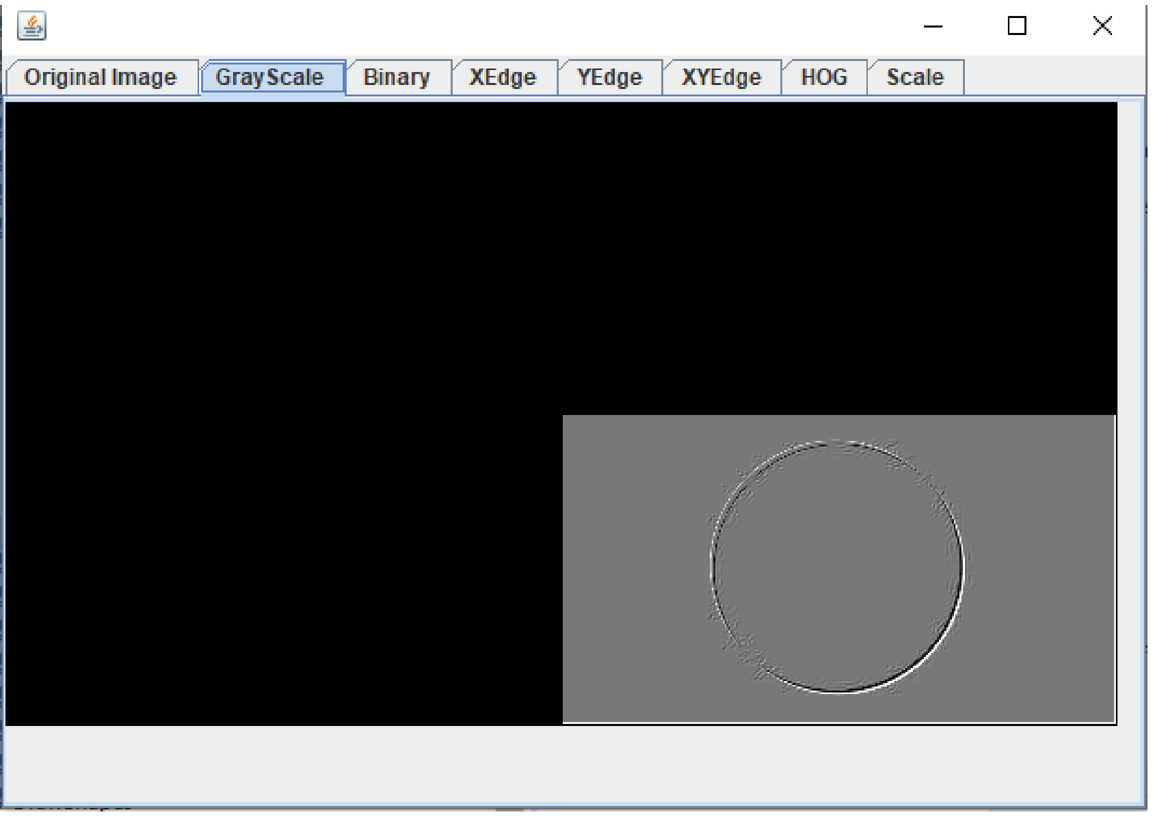
B1805.010040

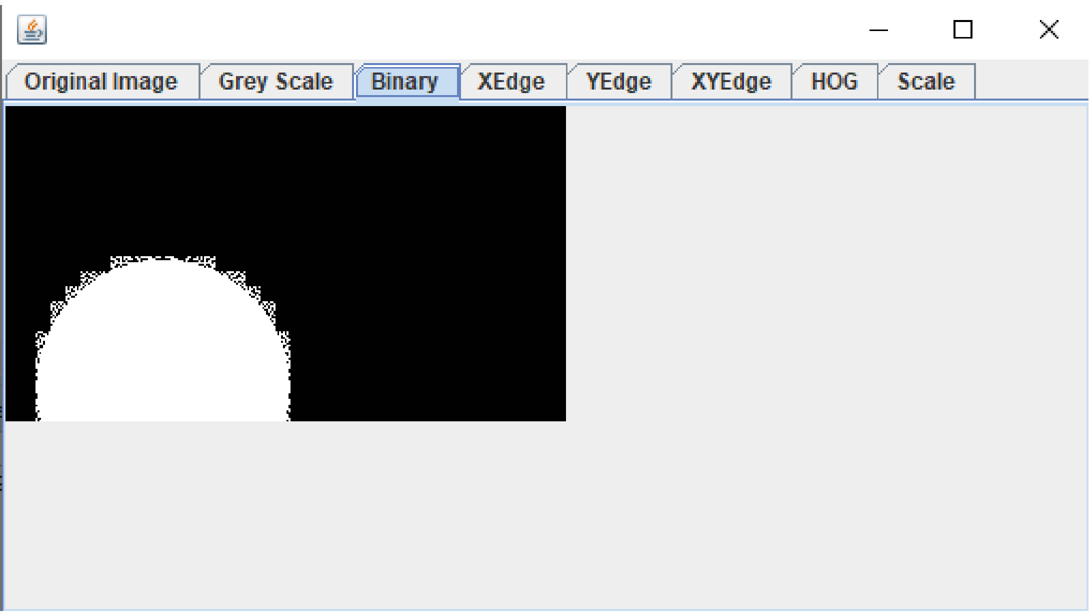
TEACHER

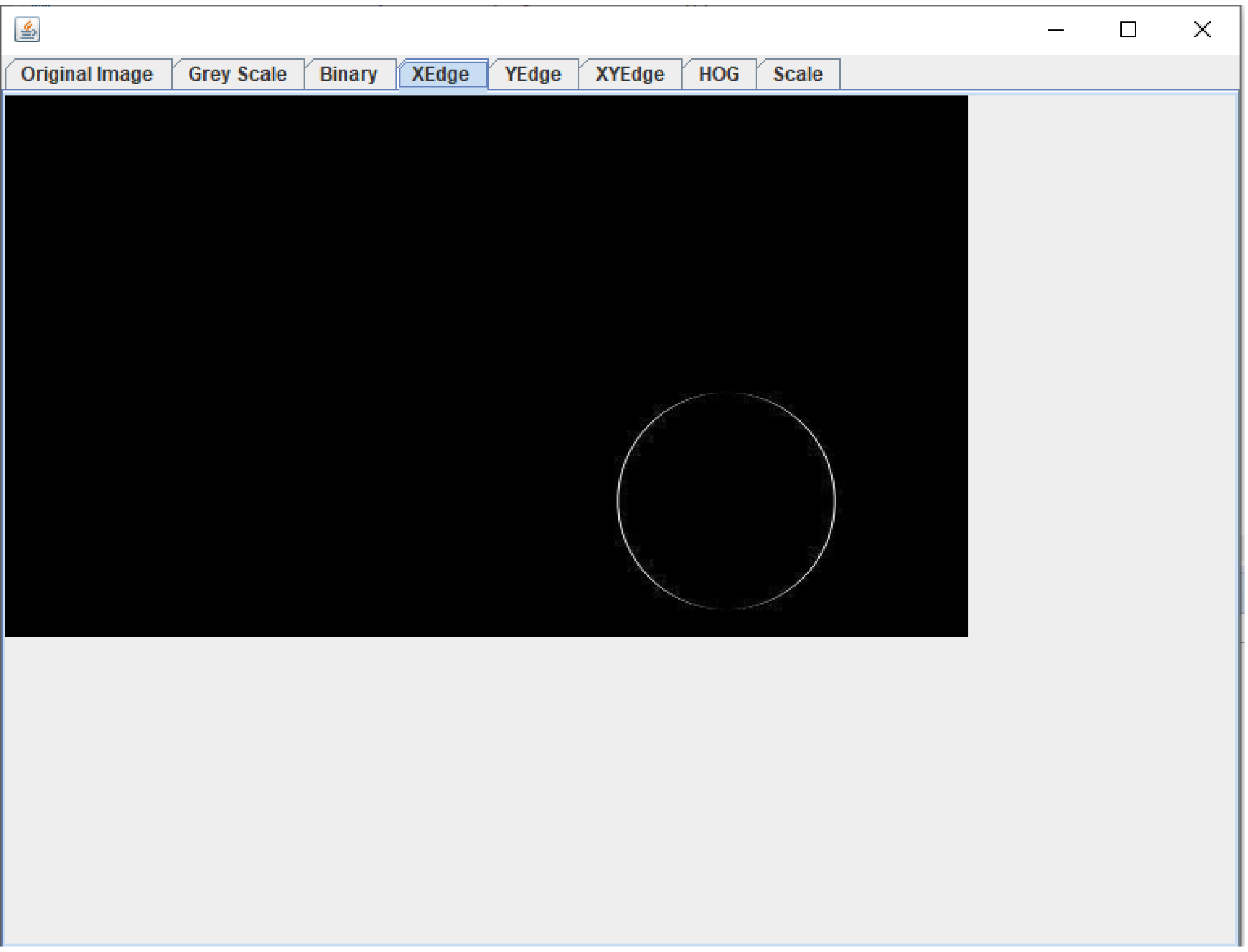
DR. ADEM OZYAVAS

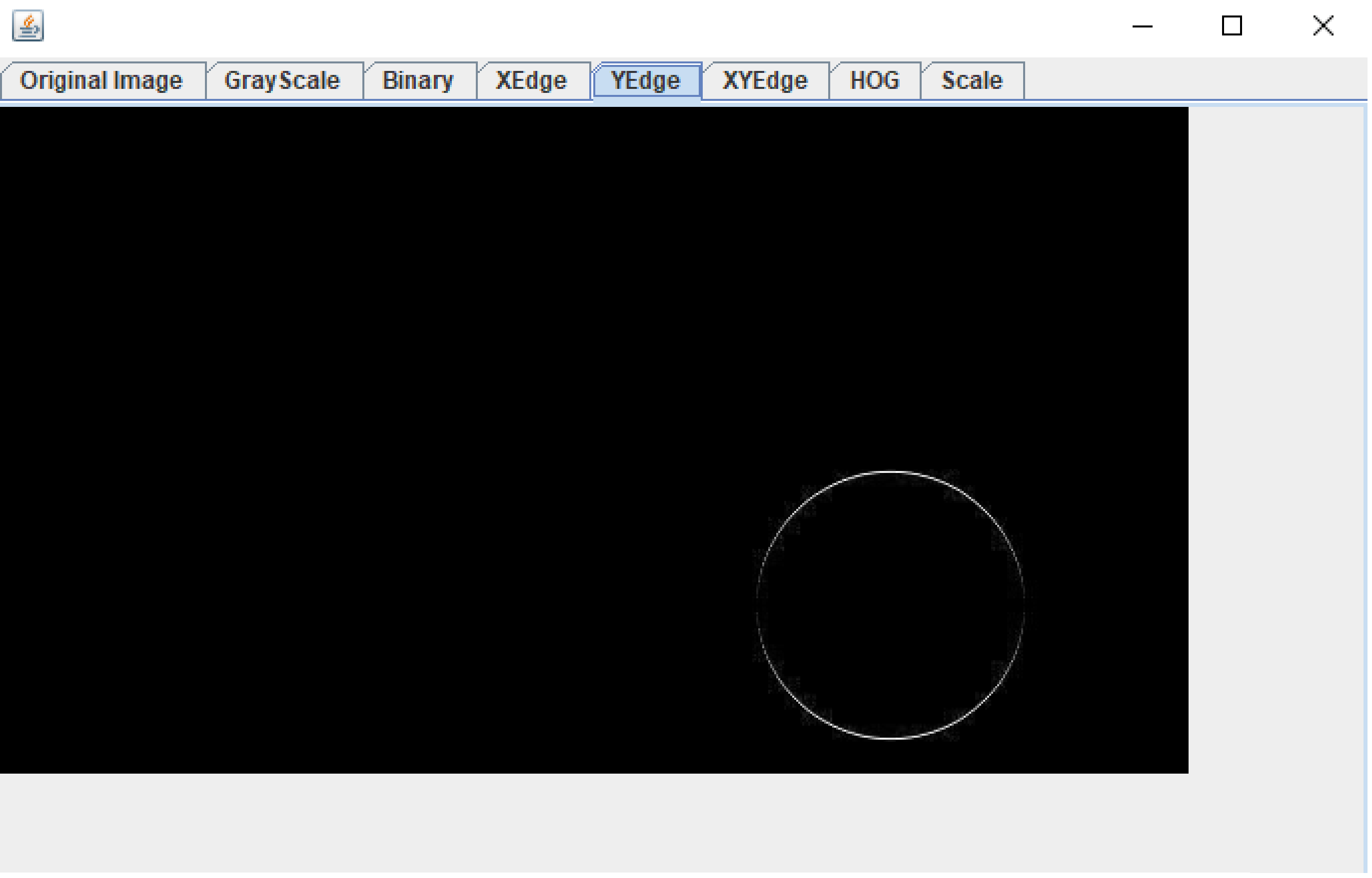
PART 1 OUTPUTS

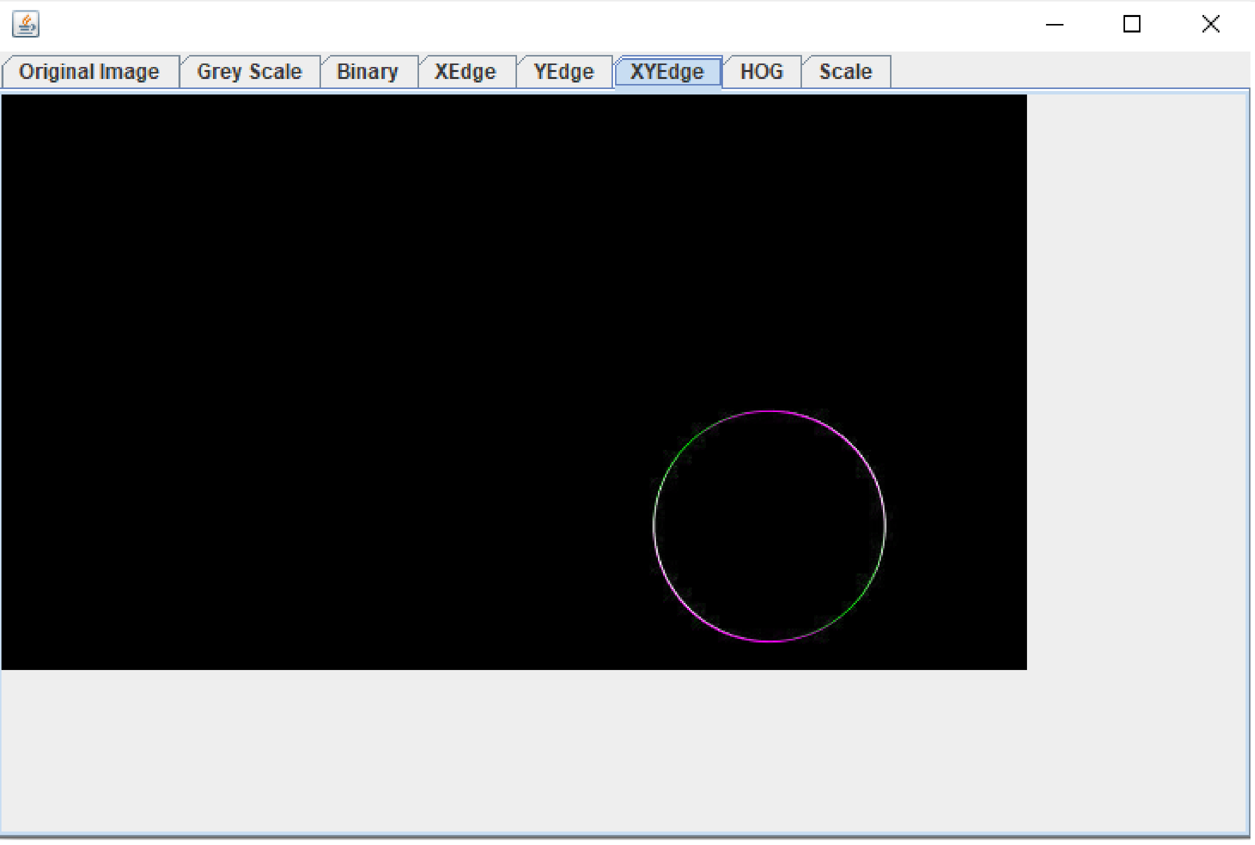


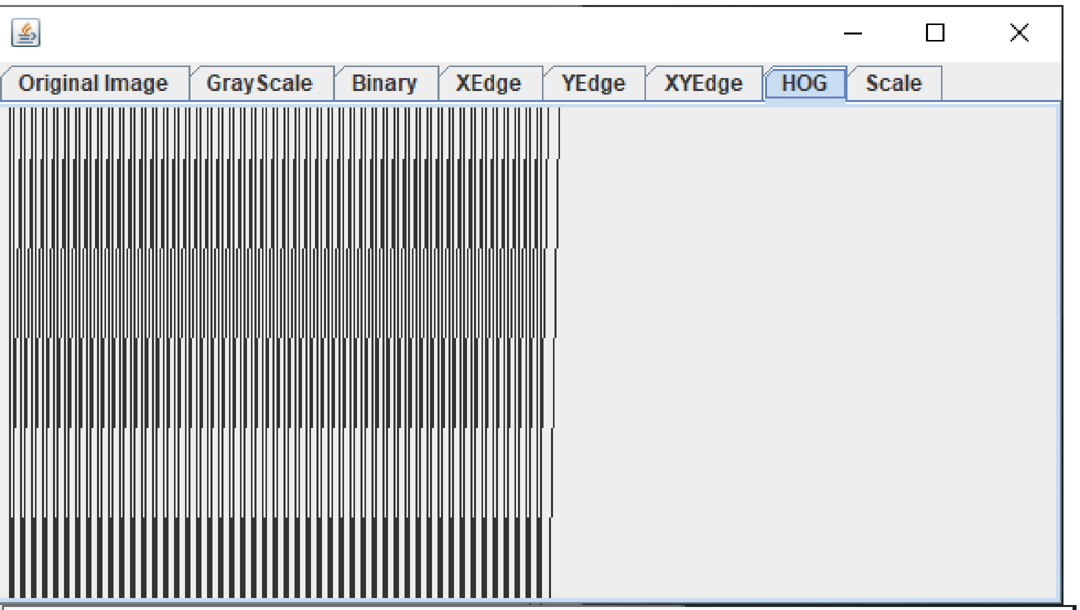


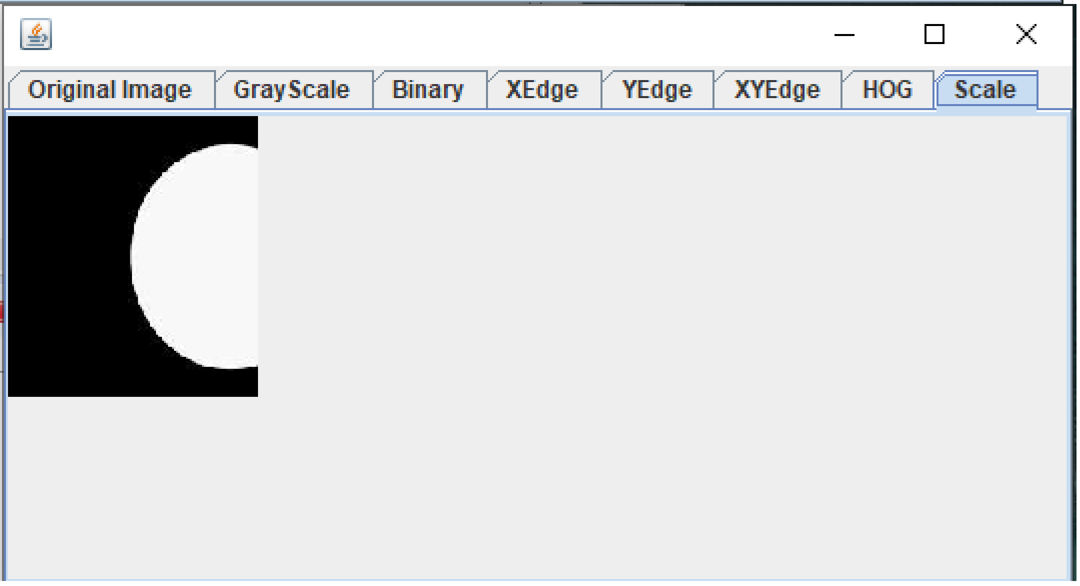

Original Image
Grey Scale
Binary
XEdge
YEdge
XYEdge
HOG
Scale


Original Image
Grey Scale
Binary
XEdge
YEdge
XYEdge
HOG
Scale


Original Image
GrayScale
Binary
XEdge
YEdge
XYEdge
HOG
Scale


Original Image
Grey Scale
Binary
XEdge
YEdge
XYEdge
HOG
Scale





Source Codes

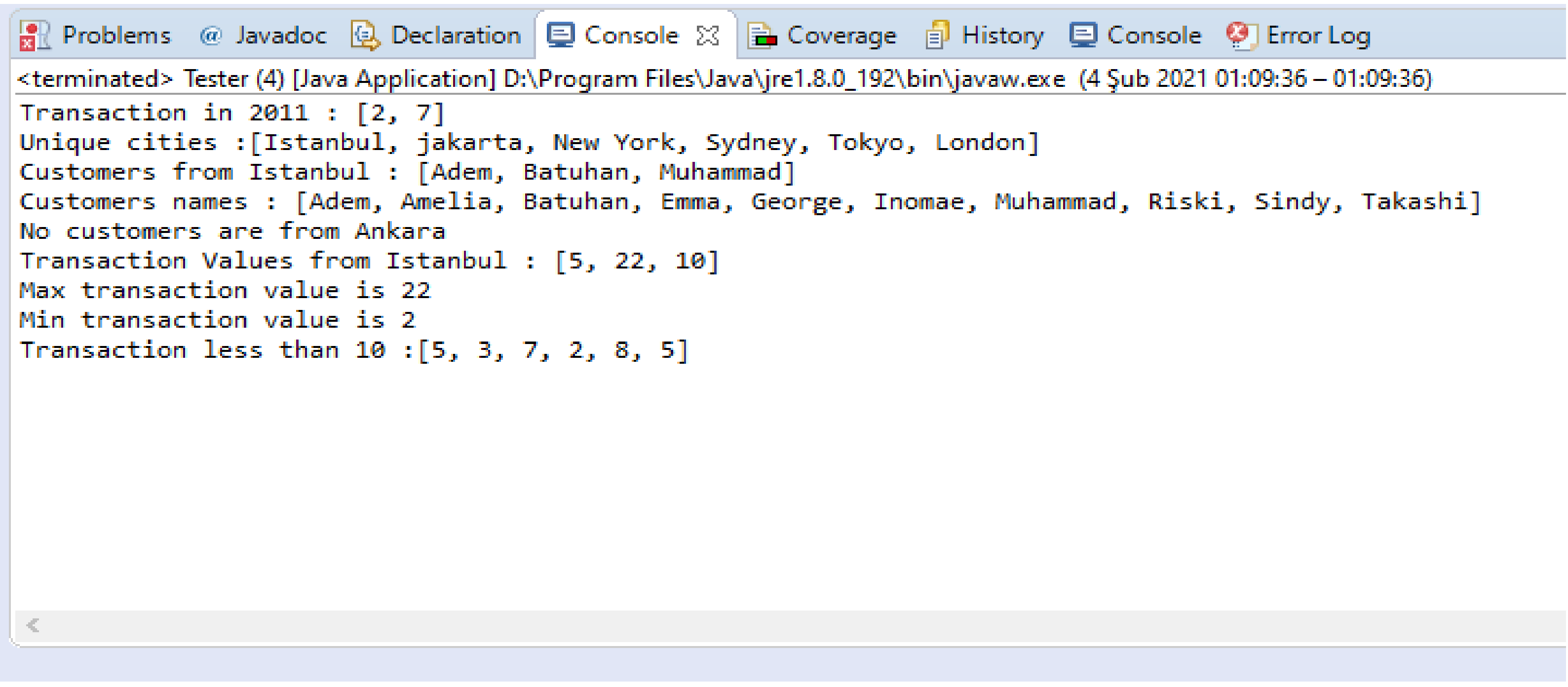
```java
package finalpartone;

import java.awt.Color;
import java.awt.Graphics;
import java.awt.image.BufferedImage;
import java.awt.image.DataBufferByte;
import java.awt.image.Raster;
import java.io.File;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

import javax.imageio.ImageIO;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JTabbedPane;

public class FinalPartOne extends JFrame{
    private int [][] pixels;
    private int width, height;
    private TabOne tabOne;
    private TabTwo tabTwo;
    private TabThree tabThree;
    private TabFour tabFour;
    private TabFive tabFive;
    private TabSix tabSix;
    private TabSeven tabSeven;
    private TabEight tabEight;
    private BufferedImage img = null;
    private int[][][] rgb_buffer;
    private byte[] p;
      FinalPartOne(){
            readImage();
            JTabbedPane jtp = new JTabbedPane();
            tabOne = new TabOne();
            jtp.add("Original Image", tabOne);
            tabTwo = new TabTwo();
            jtp.add("GrayScale", tabTwo);
            tabThree = new TabThree();
            jtp.add("Binary", tabThree);
            tabFour = new TabFour();
            jtp.add("XEdge" ,tabFour);
            tabFive = new TabFive();
            jtp.add("YEdge" ,tabFive);
            tabSix = new TabSix();
            jtp.add("XYEdge" , tabSix);
            tabSeven = new TabSeven();
            jtp.add("HOG" , tabSeven);
            tabEight = new TabEight();
            jtp.add("Scale" ,tabEight);
            this.add(jtp);
            this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            this.setSize(598,335);
            this.setVisible(true);
      }


      private void readImage() {

            try {
                  File f = new File("circle1.jpg");


                  img = ImageIO.read(f);
                  width = img.getWidth();
                  height = img.getHeight();
                  pixels = new int[300][500];
                  System.out.printf("Width : %d, height : %d ", width, height);
                  Raster raster = img.getData();

                  DataBufferByte data   = (DataBufferByte) raster.getDataBuffer();

           p = data.getData();

                  for(int row = 0; row < height; row++) {
                        for(int col = 0; col < width; col++) {
                              //pixels[col][row] = image.getRGB(0, 0);
                              pixels[col][row] = raster.getSample(col, row, 0);
                        }

                }
        } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
        }
                System.out.println("Reading complete.");
                //System.out.println(p.length);
}

class TabOne extends JPanel{
        @Override
        public void paintComponent(Graphics g) {
                super.paintComponent(g);
                for(int row = 0; row < height; row++)
                        for(int col = 0; col < width; col++) {
                                g.setColor(new Color(pixels[col][row],
                                                pixels[col][row],
                                                pixels[col][row]));
                                g.fillRect(col, row, 1, 1);
                        }
        }
}
class TabTwo extends JPanel{

        @Override
        public void paintComponent(Graphics g) {
                 rgb_buffer = new int[3][img.getHeight()][img.getWidth()];
                super.paintComponent(g);
                for(int row = 0; row < height; row++) {
                        for(int col = 0; col < width; col++) {
                                g.setColor(new Color(pixels[col][row],
                                                pixels[col][row],
                                                pixels[col][row]));
                        //      g.fillRect(col, row, 1, 1);
                                g.drawImage(img, col, row , null);

                        Color c= new Color(img.getRGB(col, row));
                        rgb_buffer[0][row][col]=c.getRed();
                        rgb_buffer[1][row][col]=c.getGreen();
                        rgb_buffer[2][row][col]=c.getBlue();

                        }

                        }

 for(int row = 1; row < height-1; row++) {
   for(int col = 1; col < width-1; col++) {
     int r =0,gr=0,b=0;
r = Math.min(Math.abs((rgb_buffer[0][row][col]-rgb_buffer[0][row+1][col+1])+120),255);
  gr = Math.min(Math.abs((rgb_buffer[1][row][col]-rgb_buffer[0][row+1][col+1])+120),255);
    b = Math.min(Math.abs((rgb_buffer[2][row][col]-rgb_buffer[0][row+1][col+1])+120),255);

 Color c= new Color(r,gr,b);
  img.setRGB(col,row,c.getRGB());

}

                                }

                        }
        }
```

```
class TabThree extends JPanel{
      @Override
      public void paintComponent(Graphics g) {
            super.paintComponent(g);
            for(int row = 0; row < height; row++)
                  for(int col = 0; col <width ; col++){

                        if(pixels[row][col] == 0)
                        {
                              g.setColor(new Color(1, 1, 1));
                        }
                        else
                        {
                              g.setColor(new Color(255, 255, 255));
                        }

                        g.fillRect(col, row, 1, 1);

                  }

      }
}
class TabFour extends JPanel{
      @Override

      public void paintComponent(Graphics g) {
             rgb_buffer = new int[3][img.getHeight()][img.getWidth()];
            super.paintComponent(g);
            for(int row = 0; row < height; row++) {
                  for(int col = 0; col < width; col++) {
                        g.setColor(new Color(pixels[col][row],
                                    pixels[col][row],
                                    pixels[col][row]));

                        //    g.fillRect(col, row, 1, 1);
                        g.drawImage(img, col, row, null);

                  Color c= new Color(img.getRGB(col, row));
                  rgb_buffer[0][row][col]=c.getRed();
                  rgb_buffer[1][row][col]=c.getGreen();
                  rgb_buffer[2][row][col]=c.getBlue();

                  }

                  }

    for(int row = 1; row < height-1; row++) {
      for(int col = 1; col < width-1; col++) {
      int r =0,gr=0,b=0;
    r = Math.min(Math.abs((rgb_buffer[0][row][col]-rgb_buffer[0][row][col+1])+0),256);
    gr = Math.min(Math.abs((rgb_buffer[1][row][col]-rgb_buffer[0][row][col+1])+0),256);
   b = Math.min(Math.abs((rgb_buffer[2][row][col]-rgb_buffer[0][row][col+1])+0),256);

                              Color c= new Color(r,gr,b);
                              img.setRGB(col,row,c.getRGB());

                              }

                        }

                  }
}
class TabFive extends JPanel{
      @Override
      public void paintComponent(Graphics g) {
              rgb_buffer = new int[3][img.getHeight()][img.getWidth()];
             super.paintComponent(g);
             for(int row = 0; row < height; row++) {
                   for(int col = 0; col < width; col++) {
                         g.setColor(new Color(pixels[col][row],
                                      pixels[col][row],
                                      pixels[col][row]));

                         //     g.fillRect(col, row, 1, 1);
                         g.drawImage(img, col, row, null);

                   Color c= new Color(img.getRGB(col, row));
                   rgb_buffer[0][row][col]=c.getRed();
                   rgb_buffer[1][row][col]=c.getGreen();
                   rgb_buffer[2][row][col]=c.getBlue();

                   }

                   }

    for(int row = 1; row < height-1; row++) {
      for(int col = 1; col < width-1; col++) {
       int r =0,gr=0,b=0;
        r = Math.min(Math.abs((rgb_buffer[0][row][col]-rgb_buffer[0][row+1][col])+0),256);
       gr = Math.min(Math.abs((rgb_buffer[1][row][col]-rgb_buffer[0][row+1][col])+0),256);
        b = Math.min(Math.abs((rgb_buffer[2][row][col]-rgb_buffer[0][row+1][col])+0),256);

                                Color c= new Color(r,gr,b);
                               img.setRGB(col,row,c.getRGB());

                               }

                         }

                   }
}
class TabSix extends JPanel{
      public void paintComponent(Graphics g) {
              rgb_buffer = new int[3][img.getHeight()][img.getWidth()];
             super.paintComponent(g);
             for(int row = 0; row < height; row++) {
                   for(int col = 0; col < width; col++) {
                         g.setColor(new Color(pixels[col][row],
                                      pixels[col][row],
                                      pixels[col][row]));

                         //     g.fillRect(col, row, 1, 1);
                         g.drawImage(img, col, row, null);

                   Color c= new Color(img.getRGB(col, row));
                   rgb_buffer[0][row][col]=c.getRed();
                   rgb_buffer[1][row][col]=c.getGreen();
                   rgb_buffer[2][row][col]=c.getBlue();

                   }

                   }

   for(int row = 2; row < height-2; row++) {
    for(int col = 2; col < width-2; col++) {
     int r =0,gr=0,b=0;
   r = Math.min(Math.abs((rgb_buffer[0][row][col]-rgb_buffer[0][row-1][col+1])+0),256);
```

```
    gr = Math.min(Math.abs((rgb_buffer[0][row][col]-rgb_buffer[0][row][col+1])+0),256);
    b = Math.min(Math.abs((rgb_buffer[0][row][col]-rgb_buffer[0][row-1][col+1])+0),256);

                                Color c= new Color(r,gr,b);
                               img.setRGB(col,row,c.getRGB());

                               }

                        }

                 }
}
class TabSeven extends JPanel{
 @Override
 public void paintComponent(Graphics g) {
     super.paintComponent(g);

     int[][] filter1 = {
             { -1,  0,  1 },
             { -2,  0,  2 },
             { -1,  0,  1 }
         };

         int[][] filter2 = {
             {  1,  2,  1 },
             {  0,  0,  0 },
             { -1, -2, -1 }
         };

         Integer horizontal[] = new Integer[height];

         List<Integer> arrList = new ArrayList<Integer>();

         for (int y = 1; y < height - 1; y++) {
             for (int x = 1; x < width - 1; x++) {
                 int[][] gray = new int[3][3];
                 for (int i = 0; i < 3; i++) {
                     for (int j = 0; j < 3; j++) {
                         gray[i][j] = (int)(img.getRGB(x-1+i, y-1+j));

                     }
                 }

                 int gray1 = 0, gray2 = 0;
                 for (int i = 0; i < 3; i++) {
                     for (int j = 0; j < 3; j++) {

                         gray1 += gray[i][j] * filter1[i][j];
                         gray2 += gray[i][j] * filter2[i][j];

                     }
                 }
                arrList.add(gray1);
                  arrList.add(gray2);
                int magnitude = 255 - ((int) Math.sqrt(gray1*gray1 + gray2*gray2));


                Color c = new Color((int) magnitude);
                img.setRGB(x, y, c.getRGB());

                horizontal = arrList.toArray(horizontal);
                g.drawLine(x, 400, x+10, 400-horizontal[x]);

             }
```

```
                    }

            }
        }

        class TabEight extends JPanel{
                public void paintComponent(Graphics g) {
                        super.paintComponent(g);

                            g.drawImage(img, 0, 0, 125,250 , 0, 0, height , width, null);
                            g.dispose();

                               }

                }



        public static void main(String[] args) {
                // TODO Auto-generated method stub
                new FinalPartOne();
        } }
```

PART 2 OUTPUT



```
<terminated> Tester (4) [Java Application] D:\Program Files\Java\jre1.8.0_192\bin\javaw.exe (4 Şub 2021 01:09:36 – 01:09:36)
Transaction in 2011 : [2, 7]
Unique cities :[Istanbul, jakarta, New York, Sydney, Tokyo, London]
Customers from Istanbul : [Adem, Batuhan, Muhammad]
Customers names : [Adem, Amelia, Batuhan, Emma, George, Inomae, Muhammad, Riski, Sindy, Takashi]
No customers are from Ankara
Transaction Values from Istanbul : [5, 22, 10]
Max transaction value is 22
Min transaction value is 2
Transaction less than 10 :[5, 3, 7, 2, 8, 5]
```

PART 2 SOURCE CODES

```
package finalparttwo;

import java.util.ArrayList;
import java.util.List;
import java.util.function.Predicate;
import java.util.stream.Collectors;
```

```
public class Tester {

      public static void main(String[] args) {

            List<Customer> cus = new ArrayList<Customer>();
            cus.add(new Customer("Muhammad","Rivalsyah", 2020, "Istanbul", 5));
            cus.add(new Customer("Batuhan","Satilmis", 2018, "Istanbul", 22));
            cus.add(new Customer("Riski","Mudafarsyah", 2018, "jakarta", 3));
            cus.add(new Customer("George","Petterson", 2011, "New York", 7));
            cus.add(new Customer("Emma","Watson", 2011, "Sydney", 2));
            cus.add(new Customer("Takashi","Oshiro", 2006, "Tokyo", 11));
            cus.add(new Customer("Amelia","Watson", 2020, "London", 8));
            cus.add(new Customer("Inomae","Ina", 2020, "Tokyo", 17));
            cus.add(new Customer("Sindy","Barbie", 2010, "Sydney", 5));
            cus.add(new Customer("Adem","Ozyavas", 2003, "Istanbul", 10));

            List<Customer> trancinyear2011 = new ArrayList<Customer>();
            List<Integer> value = new ArrayList<Integer>();

//1
            for(Customer d: cus)
                  if(d.getYear() == 2011)
                        trancinyear2011.add(d);

            for(Customer d: cus)
                  value.add(d.getTransaction());


            List<Integer> integ =
                        cus.stream()
                        .filter(d-> d.getYear() == 2011)
                        .map(d-> d.getTransaction())
                        .sorted((a,b) -> a.compareTo(b))
                        .collect(Collectors.toList());
            System.out.println("Transaction in 2011 : " + integ);





//2
            List<String> cities = new ArrayList<String>();
            for(Customer d: cus)
                  cities.add(d.getCity());

            List<String> uniquecities =
                        cus.stream()
                        .map(d-> d.getCity())
                        .distinct()
                        .collect(Collectors.toList());
            System.out.println("Unique cities :" + uniquecities);

//3


            List<String> istanbulcus =
                        cus.stream()
                        .filter(d -> d.getCity() == "Istanbul")
                        .map(d-> d.getName())
                        .sorted((a,b) -> a.compareTo(b))
                        .collect(Collectors.toList());
            System.out.println("Customers from Istanbul : "+ istanbulcus);

//4
```

```
		List<Customer> cusname = new ArrayList<Customer>();
		for(Customer d : cus)
			cusname.add(d);

		List<String> allnames =
				cus.stream()
				.map(d -> d.getName())
				.sorted((a,b) -> a.compareTo(b))
				.collect(Collectors.toList());

		System.out.println("Customers names : "+ allnames);

//5
		List<String> ankara = new ArrayList<String>();

		for(Customer d: cus)
			if(d.getCity() == "Ankara")
				ankara.add(d.getCity());

		List<String> cusankara =
				cus.stream()
				.filter(d -> d.getCity() == "Ankara")
				.map(d -> d.getName())
				.collect(Collectors.toList());

		if (cusankara != null)
			System.out.println("No customers are from Ankara");
		else
			System.out.println("Customers from Ankara : " + cusankara);


//6
		List<Integer> valuesfromistanbul =
				cus.stream()
				.filter(d -> d.getCity() == "Istanbul")
				.map(Customer::getTransaction)
				.collect(Collectors.toList());
		System.out.println("Transaction Values from Istanbul : " + valuesfromistanbul);

//7

		int max = value.stream()
				.collect(Collectors.summarizingInt(Integer::intValue)).getMax();
		System.out.println("Max transaction value is "+ max);

//8

		int min = value.stream()
				.collect(Collectors.summarizingInt(Integer::intValue)).getMin();
		System.out.println("Min transaction value is "+ min);

//9

		List<Customer> valuelessthan10 = new ArrayList<Customer>();
		for (Customer c : cus)
			if (c.getTransaction() < 10)
				valuelessthan10.add(c);

		List<Integer> lessthan10 =
				cus.stream()
				.filter(d -> d.getTransaction() < 10)
				.map(d -> d.getTransaction())
				.collect(Collectors.toList());

		System.out.println("Transaction less than 10 :" + lessthan10);
```

```
	}


}

@FunctionalInterface
interface MyPredicate<T>{
	boolean mytest(T arg);
}

class SomeFilter<T> implements Predicate<T>{
	public boolean mytest(T arg) {
		return true;
	}

	@Override
	public boolean test(T t) {
		// TODO Auto-generated method stub
		return false;
	}
}

class Customer {
	private String name;
	private String surname;
	private int year;
	private String city;
	private int transaction;
	public Customer (String n, String s, int y, String c, int t)
	{
		this.name = n;
		this.surname = s;
		this.year = y;
		this.city = c;
		this.transaction = t;

	}
	public String getName() {
		return name;
	}
	public String getSurname() {
		return surname;
	}
	public int getYear() {
		return year;
	}
	public String getCity() {
		return city;
	}
	public int getTransaction() {
		return transaction;
	}


}
```