```java
// To find all the primitive roots of the group G = <Zp*, *>

import java.util.*;

public class prac4 {
    public static void main(String args[])
    {
        List<Integer> elements = new ArrayList<Integer>();
        List<Integer> roots = new ArrayList<Integer>();
        Scanner scan = new Scanner(System.in);

        System.out.print("\nEnter n:: ");
        int n = scan.nextInt();
        int gcd = 0, temp = 1, element = 0;

        // checking prime or not
        boolean prime = false;
        while(prime == false)
        {
            if(n%6==1 || n%6==5 || n==2 || n==3)
            {
                prime = true;
            }
            else
            {
                System.out.print("\nNumber is not prime. Enter again:: ");
                n = scan.nextInt();
            }
        }
        while(temp!=n)
        {
            for(int i = 1; i <= n; i++)
            {
                if(temp%i==0 && n%i==0)
                {
                    gcd = i;
                    //storing number if gcd is 1
                    if(gcd == 1)
                    {
                        element = temp;
            }
        }
    }
    //adding number to the array
    if(gcd==1)
    {
        elements.add(element);
    }
    temp = temp + 1;
}

//printing Zn*
System.out.printf("\nZ%d* = {",n);
for(int i=0; i< elements.size(); i++)
{
    System.out.print(elements.get(i) + ", ");
}
System.out.print("\b\b}\n");

//store and print order of group
int orderOfGroup = elements.size();
    //unicode for phi is \u03D5
System.out.println("\n\u03D5("+n+") = "+ orderOfGroup);

//creating array
int[][] table = new int[orderOfGroup+1][orderOfGroup+1];
for(int i=1; i<=orderOfGroup;i++)
{
    table[0][i] = i;
    table[i][0] = elements.get(i-1);
}

//creating table
for(int i = 1; i <= orderOfGroup; i++)
{
    for(int j = 1; j <= orderOfGroup; j++)
    {
        int base = table[i][0];
        int exponent = table[0][j];
        int result = 1;
        // finding power(base,exp)
        // Math.power(a,b) returns double, hence not used.
        for(int k = 1; k <= exponent; k++)
        {
            result = result * base;
        }

        table[i][j] = result % n;
    }
}

//printing generated table
String[][] display_table = new String[orderOfGroup+1][orderOfGroup+1];
display_table[0][0] = " ";

for(int i=1; i<=orderOfGroup;i++)
{
    display_table[0][i] = "\ti = "+table[0][i];
    display_table[i][0] = "a = "+table[i][0];
}

for(int i = 1; i <= orderOfGroup; i++)
{
    for(int j = 1; j <= orderOfGroup; j++)
    {
        display_table[i][j] = "\t  "+Integer.toString(table[i][j]);
    }
}

System.out.print("\nCyclic group generated:: \n");
for(int i = 0; i <= orderOfGroup; i++)
{
    for(int j = 0; j <= orderOfGroup; j++)
    {
        System.out.print(display_table[i][j] + " ");
    }
    System.out.print("\n");
}
```

```
        //finding primitive roots
        int root = 0, order = 0;
        for(int i = 1; i <= orderOfGroup; i++)
        {
            for(int j = 1; j <= orderOfGroup; j++)
            {
                if(table[i][j] == 1)
                {
                    root = i;
                    order = j;
                    j = orderOfGroup + 1;
                }
            }
            if(order == orderOfGroup)
            {
                roots.add(root);
            }
        }

        //printing roots
        System.out.printf("\nPrimitive Roots:: ");
        for(int i=0; i< roots.size(); i++)
        {
            System.out.print(roots.get(i)+" ");
        }
        System.out.printf("\n\n");

        scan.close();
    }
}
```