SOME IMPORTANT CODES SOLVED WITH JAVA:

1. Program to check whether the given number is Armstrong number

SOLUTION: A number is called Armstrong number if the following equation holds true for that number:

```
xy..z = xⁿ + yⁿ+.....+ zⁿ
```

where n denotes the number of digits in the number

For example this is a 3 digit Armstrong number

```
370 = 3³ + 7³ + o³
          = 27 + 343 + 0
          = 370

//code to check Armstrong number

public class JavaExample {

    public static void main(String[] args) {

        int num = 370, number, temp, total = 0;

        number = num;
        while (number != 0)
        {
            temp = number % 10;
            total = total + temp*temp*temp;
            number /= 10;
        }

        if(total == num)
            System.out.println(num + " is an Armstrong number");
        else
            System.out.println(num + " is not an Armstrong number");
    }
}
```

Output:

```
370 is an Armstrong number
```

2. **Program to check Leap Year**

SOLUTION: Here we will write a java program to check whether the input year is a leap year or not. Before we see the program, lets see how to determine whether a year is a leap year mathematically:
To determine whether a year is a leap year, follow these steps:
1. If the year is evenly divisible by 4, go to step 2. Otherwise, go to step 5.
2. If the year is evenly divisible by 100, go to step 3. Otherwise, go to step 4.
3. If the year is evenly divisible by 400, go to step 4. Otherwise, go to step 5.
4. The year is a leap year (it has 366 days).
5. The year is not a leap year (it has 365 days).

//code to check leap year

```java
import java.util.Scanner;
public class Demo {

    public static void main(String[] args) {

        int year;
        Scanner scan = new Scanner(System.in);
        System.out.println("Enter any Year:");
        year = scan.nextInt();
        scan.close();
        boolean isLeap = false;

        if(year % 4 == 0)
        {
            if( year % 100 == 0)
            {
                if ( year % 400 == 0)
                    isLeap = true;
                else
                    isLeap = false;
            }
            else
                isLeap = true;
        }
        else {
            isLeap = false;
        }

        if(isLeap==true)
            System.out.println(year + " is a Leap Year.");
        else
            System.out.println(year + " is not a Leap Year.");
    }
}
```

Output:

```
Enter any Year:
2001
```

```
2001 is not a Leap Year.
```

3. Program to Calculate Area of Rectangle

SOLUTION:

//code to calculate area of rectangle

```
import java.util.Scanner;
class AreaOfRectangle {
   public static void main (String[] args)
   {
	   Scanner scanner = new Scanner(System.in);
	   System.out.println("Enter the length of Rectangle:");
	   double length = scanner.nextDouble();
	   System.out.println("Enter the width of Rectangle:");
	   double width = scanner.nextDouble();
	   //Area = length*width;
	   double area = length*width;
	   System.out.println("Area of Rectangle is:"+area);
   }
}
```

Output:

```
Enter the length of Rectangle:
2
Enter the width of Rectangle:
8
Area of Rectangle is:16.0
```

4. Java Program to Find Factorial using For

SOLUTION:

Factorial of a number n is denoted as n! and the value of n! is: 1 * 2 * 3 * … (n-1) * n

//code to find factorial

```
public class JavaExample {

    public static void main(String[] args) {

        //We will find the factorial of this number
        int number = 5;
        long fact = 1;
        for(int i = 1; i <= number; i++)
        {
            fact = fact * i;
        }
        System.out.println("Factorial of "+number+" is: "+fact);
    }
}
```

Output:

```
Factorial of 5 is: 120
```

5. Java program to reverse a number using while

SOLUTION:

//code to reverse a number string

```
import java.util.Scanner;
class ReverseNumberWhile
{
   public static void main(String args[])
   {
      int num=0;
      int reversenum =0;
      System.out.println("Input your number and press enter: ");
      //This statement will capture the user input
      Scanner in = new Scanner(System.in);
      //Captured input would be stored in number num
      num = in.nextInt();
      //While Loop: Logic to find out the reverse number
      while( num != 0 )
      {
          reversenum = reversenum * 10;
          reversenum = reversenum + num%10;
          num = num/10;
      }

      System.out.println("Reverse of input number is: "+reversenum);
   }
}
```

Output:

```
Input your number and press enter:
145689
Reverse of input number is: 986541
```

6. Program to display prime numbers from 1 to n

SOLUTION:

//code to display prime numbers

```
import java.util.Scanner;
class PrimeNumbers2
{
   public static void main (String[] args)
   {
      Scanner scanner = new Scanner(System.in);
      int i =0;
      int num =0;
      //Empty String
      String  primeNumbers = "";
      System.out.println("Enter the value of n:");
      int n = scanner.nextInt();
      scanner.close();
      for (i = 1; i <= n; i++)
      {
         int counter=0;
         for(num =i; num>=1; num--)
         {
            if(i%num==0)
            {
                counter = counter + 1;
            }
         }
         if (counter ==2)
         {
            //Appended the Prime number to the String
            primeNumbers = primeNumbers + i + " ";
         }
      }
      System.out.println("Prime numbers from 1 to n are :");
      System.out.println(primeNumbers);
   }
}
```

Output:

```
Enter the value of n:
150
Prime numbers from 1 to n are :
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89
97 101 103 107 109 113 127 131 137 139 149
```

7. Java Program to Display Fibonacci Series using loops

SOLUTION:

//code for Fibonacci series

```
import java.util.Scanner;
public class JavaExample {

    public static void main(String[] args) {

        int count, num1 = 0, num2 = 1;
        System.out.println("How may numbers you want in the sequence:");
        Scanner scanner = new Scanner(System.in);
        count = scanner.nextInt();
        scanner.close();
        System.out.print("Fibonacci Series of "+count+" numbers:");

        int i=1;
        while(i<=count)
        {
            System.out.print(num1+" ");
            int sumOfPrevTwo = num1 + num2;
            num1 = num2;
            num2 = sumOfPrevTwo;
            i++;
        }
    }
}
```

Output:

```
How may numbers you want in the sequence:
6
Fibonacci Series of 6 numbers:0 1 1 2 3 5
```

8. Program to print magic square

SOLUTION:

A magic square of order n is an arrangement of n^2 numbers, usually distinct integers, in a square, such that the n numbers in all rows, all columns, and both diagonals sum to the same constant. A magic square contains the integers from 1 to n^2.
The constant sum in every row, column and diagonal is called the magic constant or magic sum, M. The magic constant of a normal magic square depends only on n and has the following value: M = n(n^2+1)/2

```
For normal magic squares of order n = 3, 4, 5, ...,
the magic constants are: 15, 34, 65, 111, 175, 260, ...
```

Three conditions hold:

1. The position of next number is calculated by decrementing row number of previous number by 1, and incrementing the column number of previous number by 1. At any time, if the calculated row position becomes -1, it will wrap around to n-1. Similarly, if the calculated column position becomes n, it will wrap around to 0.

2. If the magic square already contains a number at the calculated position, calculated column position will be decremented by 2, and calculated row position will be incremented by 1.

3. If the calculated row position is -1 & calculated column position is n, the new position would be: (0, n-2).

```
// code for odd number magic square

class GFG
{
    // Function to generate odd sized magic squares
    static void generateSquare(int n)
    {
        int[][] magicSquare = new int[n][n];

        // Initialize position for 1
        int i = n/2;
        int j = n-1;

        // One by one put all values in magic square
        for (int num=1; num <= n*n; )
        {
            if (i==-1 && j==n) //3rd condition
            {
                j = n-2;
                i = 0;
            }
            else
            {
                //1st condition helper if next number
                // goes to out of square's right side
                if (j == n)
                    j = 0;

                //1st condition helper if next number is
                // goes to out of square's upper side
                if (i < 0)
                    i=n-1;
            }

            //2nd condition
            if (magicSquare[i][j] != 0)
            {
                j -= 2;
                i++;
                continue;
            }
            else
                //set number
                magicSquare[i][j] = num++;

            //1st condition
            j++;  i--;
        }

        // print magic square
        System.out.println("The Magic Square for "+n+":");
        System.out.println("Sum of each row or column "+n*(n*n+1)/2+":");
        for(i=0; i<n; i++)
        {
            for(j=0; j<n; j++)
                System.out.print(magicSquare[i][j]+" ");
            System.out.println();
        }
    }

    // driver program
    public static void main (String[] args)
    {
        // Works only when n is odd
        int n = 7;
        generateSquare(n);
    }
}
```

Output:

```
The Magic Square for n=7:
Sum of each row or column 175:

 20  12   4  45  37  29  28
 11   3  44  36  35  27  19
  2  43  42  34  26  18  10
 49  41  33  25  17   9   1
 40  32  24  16   8   7  48
 31  23  15  14   6  47  39
 22  21  13   5  46  38  30
```

NOTE:FOR ODD NUMBERS ONLY

```
// code for even number Magic square
import java.io.*;

class GFG
{
    // Function for calculating Magic square
    static void doublyEven(int n)
    {
        int[][] arr = new int[n][n];
        int i, j;

        // filling matrix with its count value
        // starting from 1;
        for ( i = 0; i < n; i++)
            for ( j = 0; j < n; j++)
                arr[i][j] = (n*i) + j + 1;

        // change value of Array elements
        // at fix location as per rule
        // (n*n+1)-arr[i][j]
        // Top Left corner of Matrix
        // (order (n/4)*(n/4))
        for ( i = 0; i < n/4; i++)
            for ( j = 0; j < n/4; j++)
                arr[i][j] = (n*n + 1) - arr[i][j];

        // Top Right corner of Matrix
        // (order (n/4)*(n/4))
        for ( i = 0; i < n/4; i++)
            for ( j = 3 * (n/4); j < n; j++)
                arr[i][j] = (n*n + 1) - arr[i][j];

        // Bottom Left corner of Matrix
        // (order (n/4)*(n/4))
        for ( i = 3 * n/4; i < n; i++)
            for ( j = 0; j < n/4; j++)
                arr[i][j] = (n*n+1) - arr[i][j];

        // Bottom Right corner of Matrix
        // (order (n/4)*(n/4))
        for ( i = 3 * n/4; i < n; i++)
            for ( j = 3 * n/4; j < n; j++)
                arr[i][j] = (n*n + 1) - arr[i][j];

        // Centre of Matrix (order (n/2)*(n/2))
        for ( i = n/4; i < 3 * n/4; i++)
            for ( j = n/4; j < 3 * n/4; j++)
                arr[i][j] = (n*n + 1) - arr[i][j];

        // Printing the magic-square
        for (i = 0; i < n; i++)
        {
            for ( j = 0; j < n; j++)
                System.out.print(arr[i][j]+" ");
            System.out.println();
        }
    }

    // driver program
    public static void main (String[] args)
    {
        int n = 8;
        // Function call
        doublyEven(n);
    }
}
```

Output:

```
64   63   3    4    5    6    58   57
56   55   11   12   13   14   50   49
17   18   46   45   44   43   23   24
25   26   38   37   36   35   31   32
33   34   30   29   28   27   39   40
41   42   22   21   20   19   47   48
16   15   51   52   53   54   10   9
8    7    59   60   61   62   2   1
```

NOTE:FOR EVEN NUMBER INPUT