1. Prime Number

```
public class PrimeNo {
   // most efficient way to check prime number
   public static boolean isPrime(int n){
      if (n <= 1) return false; // 0 and 1 are not prime numbers
      if (n <= 3) return true; // 2 and 3 are prime numbers
      if (n % 2 == 0 || n % 3 == 0) return false; // eliminate multiples of 2 and 3

      for (int i = 5; i * i <= n; i += 6) { // check for factors from 5 to sqrt(n)
         if (n % i == 0 || n % (i + 2) == 0) {
            return false;
         }
      }
      return true;
   }
}
```

2. Prime no range

```
public static boolean Prime(int a) {
      if (a <= 1) return false;
      if (a == 2 || a == 3) return true;
      if (a % 2 == 0 || a % 3 == 0) return false;

      for(int i = 5; i*i < a; i+=6){
         if(a % i == 0 || a % (i+2) == 0){
            return false;
         }
      }
      return true;
   }
   public static void PrimeRange(int a, int b){
      System.out.println("Prime numbers: ");

      for(int i = a; i <= b; i++){
         if(Prime(i)) {
```

```
            System.out.println(i);
        }
    }
}
```

3. Sum Even Individuals

```
public static void SumEven(int n){
    int sum = 0;
    for( ; n>0 ; n/=10){
        int rem = n%10;

        if(n%2 ==0){
            sum = sum + rem;
        }
    }
    System.out.println("Sum of the even digits: " + sum);
}
```

4. Leap year

```
public static void leapyr(int n){
    if (n%4 == 0 && n%100 != 0 || n%400 == 0){
        System.out.println(n + " is a leap year.");
    } else {
        System.out.println(n + " is not a leap year.");
    }
}
```

5. count vowels in a string

```
public class vowels {
    public static void main(String[] args) {
        String str = "VISHNU yelde";
        System.out.println("No. of vowels in the string: " + countVowels(str));

    }

    // the most optimal way to count vowels in a string is as follows:
    public static int countVowels(String str) {
```

```
        int count = 0;
        for (int i = 0; i < str.length(); i++) {
            char ch = Character.toLowerCase(str.charAt(i));
            if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u') {
                count++;
            }
        }
        return count;
    }
}
```

6. MIN MAX ARRAY

```
public static void main(String[] args) {
        int[] arr = {12, 5, 7, 98, 2, 45, 32};
        findMinMax(arr);
    }
    // most efficient way to find min and max in a single pass
    public static void findMinMax(int[] arr) {
        if (arr == null || arr.length == 0) {
            System.out.println("Array must not be null or empty");
            return;
        }
        int min = arr[0];
        int max = arr[0];
        for (int i = 1; i < arr.length; i++) {
            if (arr[i] < min) min = arr[i];
            if (arr[i] > max) max = arr[i];
        }
        System.out.println("Minimum: " + min);
        System.out.println("Maximum: " + max);
    }
```

7. Fibonacchi series

```
public static void fibonacci(int n){
        int a = 0, b = 1;
        for (int i = 0; i < n; i++ ){
```

```
            System.out.print(a + " ");
            int c = a + b;
            a = b;
            b = c;
        }
    }
    public static void main(String[] args) {
        int n = 10;
        System.out.println("Fibonacci series up to " + n + " terms:");
        fibonacci(n);
    }
```

8. count the number of digits in an integer

```
public class NoOfDigits {
    public static void main(String[] args) {
        int number = 123456789;
        System.out.println("Number of digits in " + number + ": " + countDigits(number));
    }
    // most optimal way to count the number of digits in an integer
    public static int countDigits(int number) {
        if (number == 0) return 1; // Special case for 0
        int count = 0;
        while (number != 0) {
            number /= 10; // Remove the last digit
            count++;
        }
        return count;
    }
}
```

9. Reverse a number

```
public class ReverseNo {
    public static void main(String[] args) {
        int n = 654321;
        System.out.println("Reversed number of " + n + " is: " + reverseNumber(n));
    }
```

```
    public static int reverseNumber(int n){
        int reversed = 0;
        while (n!=0) {
            int digit = n % 10;                         // Get the last digit of n
            reversed = reversed * 10 + digit;           // Shift the current reversed number and add the last digit
            n = n/10;                                   // Remove the last digit from n
        }
        return reversed;
    }
}
```

10. Square Root

```
public class SquareRoot {
    public static void main(String[] args) {
        double n = 49;
        System.out.println("Square root of " + n + " is: " + squareRoot(n));
    }
    // time complexity O(log n) most efficient way to find square root
    public static double squareRoot(double n){
        double temp;
        double sqrt = n / 2.0; // Initial guess for the square root
        do {
            temp = sqrt;
            sqrt = (temp + (n / temp)) / 2.0; // Average of guess and n divided by guess
        }
        while ((temp - sqrt != 0.0)); // Continue until the guess stabilizes
        return sqrt;
    }
}
```