Max Sum of an hourglass:

```
import java.io.*;
class Main {
static int row = 5;
static int col = 5;
static int findMaxSum(int [][]mat)
{
        if (row < 3 || col < 3){
        System.out.println("Not possible to give");
      System.exit(0);
            }
        int max_sum = Integer.MIN_VALUE;
        for (int i = 0; i < row - 2; i++)
        {
        for (int j = 0; j < col - 2; j++)
        {
      int sum = (mat[i][j] + mat[i][j + 1] +
        mat[i][j + 2]) + (mat[i + 1][j + 1]) +
      (mat[i + 2][j] + mat[i + 2][j + 1] +
      mat[i + 2][j + 2]);
      max_sum = Math.max(max_sum, sum);
        }}
return max_sum;
}
static public void main (String[] args)
        {
        int [][]mat =
      {{1, 2, 3, 0, 0},
                {0, 0, 0, 0, 0},
                {2, 1, 4, 0, 0},
                {0, 0, 0, 0, 0},
```

```
                        {1, 1, 0, 1, 0}};
            int res = findMaxSum(mat);
            System.out.println("Maximum sum of hour glass = "+ res);
            }
}
```

Majority element in an array:

```
import java.util.*;
public class Main {
   public static int majorityElement(int []v) {
      int n = v.length;
      int cnt = 0;
      int el = 0;
      for (int i = 0; i < n; i++) {
         if (cnt == 0) {
            cnt = 1;
            el = v[i];
         } else if (el == v[i])
            cnt++;
         else
            cnt--;
      }
int cnt1 = 0;
      for (int i = 0; i < n; i++) {
         if (v[i] == el)
             cnt1++;
      }
if (cnt1 > (n / 2))
   return el;
      return -1;
   }
```

```
public static void main(String[] args) {
Scanner s = new Scanner(System.in);
System.out.println("Enter size of the array");
        int n = s.nextInt();
        int[] arr = new int[n];
System.out.println("Enter elements of the array");
        for (int i = 0; i < n; i++){
        arr[i] = s.nextInt();
        }
int ans = majorityElement(arr);
        System.out.println("The majority element is: " + ans);
    }
}
```

Block Swap algorithm:

```
public class BlockSwapAlgorithm {
    public static void rotateArray(int[] arr, int d) {
        int n = arr.length;
        d = d % n; // Adjust rotation if d is greater than array length
        if (d == 0)
            return; // No rotation needed
        swap(arr, 0, d - 1); // Swap elements in the first block
        swap(arr, d, n - 1); // Swap elements in the second block
        swap(arr, 0, n - 1); // Swap the entire array
    }
    // Helper function to swap elements in the array
    public static void swap(int[] arr, int start, int end) {
          while (start < end) {
           int temp = arr[start];
           arr[start] = arr[end];
           arr[end] = temp;
           start++;
```

```
            end--;
        }}
// Example usage
    public static void main(String[] args) {
        int[] arr = {1, 2, 3, 4, 5};
        int rotations = 2;
        rotateArray(arr, rotations);
        System.out.println("Rotated array:");
        for (int i : arr) {
            System.out.print(i + " ");
        }
    }
}
```

Leaders in an array:

Brute force:

```
import java.util.*;
class Main{
public static ArrayList<Integer>                printLeadersBruteForce(int[] arr, int n){
  ArrayList<Integer> ans= new ArrayList<>();
    for (int i = 0; i < n; i++) {
    boolean leader = true;
    for (int j = i + 1; j < n; j++)
      if (arr[j] > arr[i]) {
          leader = false;
        break;
      }
       if (leader)
    ans.add(arr[i]);
  }
```

```
  return ans;
}
public static void main(String args[])
{
  // Array Initialization.
  int n = 6;
  int arr[]=  {2,4,6,3,1,2};

  ArrayList<Integer> ans= printLeadersBruteForce(arr,n);

  for (int i = 0; i < ans.size(); i++) {
    System.out.print(ans.get(i)+" ");
  }
}
}
```

Efficient traversal:

```
import java.util.*;
import java.lang.*;
public class Main
{
public static void main(String[] args) {
Scanner s = new Scanner(System.in);
System.out.println("Enter size of the array");
        int n = s.nextInt();
        int[] arr = new int[n];
System.out.println("Enter elements of the array");
        for (int i = 0; i < n; i++){
        arr[i] = s.nextInt();
        }
findLeaders(arr, n);}
static void findLeaders(int arr[], int size){
```

```
  int rightMaximum=arr[arr.length-1];
System.out.print(rightMaximum+" ");
    for (int i = size-2; i>=0; i--) {
      if(arr[i] > rightMaximum){
        rightMaximum=arr[i];
        System.out.print(rightMaximum+" ");
      }
    }
  }
}
```

Karatsuba (GFG):

```
/// Java Program to Implement Karatsuba Algorithm

// Importing Random class from java.util packahge
import java.util.Random;

// MAin class
class GFG {

        // Main driver method
        public static long mult(long x, long y) {

                // Checking only if input is within range
                if (x < 10 && y < 10) {

                        // Multiplying the inputs entered
                        return x * y;
                }
```

```
// Declaring variables in order to
// Find length of both integer
// numbers x and y
int noOneLength = numLength(x);
int noTwoLength = numLength(y);

// Finding maximum length from both numbers
// using math library max function
int maxNumLength
        = Math.max(noOneLength, noTwoLength);

// Rounding up the divided Max length
Integer halfMaxNumLength
        = (maxNumLength / 2) + (maxNumLength % 2);

// Multiplier
long maxNumLengthTen
        = (long)Math.pow(10, halfMaxNumLength);

// Compute the expressions
long a = x / maxNumLengthTen;
long b = x % maxNumLengthTen;
long c = y / maxNumLengthTen;
long d = y % maxNumLengthTen;


// Compute all mutilpying variables
// needed to get the multiplication
long z0 = mult(a, c);
long z1 = mult(a + b, c + d);
long z2 = mult(b, d);
```

```
        long ans = (z0 * (long)Math.pow(10, halfMaxNumLength * 2) +
                        ((z1 - z0 - z2) * (long)Math.pow(10, halfMaxNumLength) + z2));

        return ans;

}

// Method 1
// To calculate length of the number
public static int numLength(long n)
{
        int noLen = 0;
        while (n > 0) {
                noLen++;
                n /= 10;
        }

        // Returning length of number n
        return noLen;
}

// Method 2
// Main driver function
public static void main(String[] args)
{
        // Showcasing karatsuba multiplication

// Case 1: Big integer lengths
        long expectedProduct = 1234 * 5678;
        long actualProduct = mult(1234, 5678);
```

```
        // Printing the expected and corresponding actual product
        System.out.println("Expected 1 : " + expectedProduct);
        System.out.println("Actual 1 : " + actualProduct + "\n\n");

        assert(expectedProduct == actualProduct);


        expectedProduct = 102 * 313;
        actualProduct = mult(102, 313);

        System.out.println("Expected 2 : " + expectedProduct);
        System.out.println("Actual 2 : " + actualProduct + "\n\n");

assert(expectedProduct == actualProduct);

        expectedProduct = 1345 * 63456;
        actualProduct = mult(1345, 63456);

        System.out.println("Expected 3 : " + expectedProduct);
        System.out.println("Actual 3 : " + actualProduct + "\n\n");

assert(expectedProduct == actualProduct);

        Integer x = null;
        Integer y = null;
        Integer MAX_VALUE = 10000;

        // Boe creating an object of random class
        // inside main() method
        Random r = new Random();
```

```
                for (int i = 0; i < MAX_VALUE; i++) {
                        x = (int) r.nextInt(MAX_VALUE);
                        y = (int) r.nextInt(MAX_VALUE);

                        expectedProduct = x * y;

                        if (i == 9999) {

                        // Prove assertions catch the bad stuff.
                                expectedProduct = 1;
                        }
                        actualProduct = mult(x, y);

                        // Again printing the expected and
                        // corresponding actual product
                        System.out.println("Expected: " + expectedProduct);
                        System.out.println("Actual: " + actualProduct + "\n\n");

                        assert(expectedProduct == actualProduct);
                }
        }
}
```

Booth algorithm:

```
import java.util.Scanner;
public class Booth
{
   public static Scanner s = new Scanner(System.in);
   /** Function to multiply **/
   public int multiply(int n1, int n2)
```

```
    {
        int[] m = binary(n1);
        int[] m1 = binary(-n1);
        int[] r = binary(n2);
        int[] A = new int[9];
        int[] S = new int[9];
        int[] P = new int[9];
 for (int i = 0; i < 4; i++)
        {   A[i] = m[i];
            S[i] = m1[i];
            P[i + 4] = r[i];
        }
display(A, 'A');
        display(S, 'S');
        display(P, 'P');
        System.out.println();
         for (int i = 0; i < 4; i++){
            if (P[7] == 0 && P[8] == 0);
                // do nothing
           else if (P[7] == 1 && P[8] == 0)
                add(P, S);
           else if (P[7] == 0 && P[8] == 1)
                add(P, A);
           else if (P[7] == 1 && P[8] == 1);
                // do nothing
 rightShift(P);
            display(P, 'P');
        }
        return getDecimal(P);
    }
/** Function to get Decimal equivalent of P **/
```

```
    public int getDecimal(int[] B){
        int p = 0;
        int t = 1;
      for (int i = 7; i >= 0; i--, t *= 2)
            p += (B[i] * t);
        if (p > 64)
            p = -(256 - p);
        return p;
    }
  /** Function to right shift array **/
    public void rightShift(int[] A)
    {
        for (int i = 8; i >= 1; i--)
            A[i] = A[i - 1];
    }
/** Function to add two binary arrays **/
    public void add(int[] A, int[] B){
        int carry = 0;
        for (int i = 8; i >= 0; i--){
            int temp = A[i] + B[i] + carry;
            A[i] = temp % 2;
            carry = temp / 2;
        }
    }
/** Function to get binary of a number **/
    public int[] binary(int n){
        int[] bin = new int[4];
        int ctr = 3;
        int num = n;
  /** for negative numbers 2 complment **/
        if (n < 0)
```

```
            num = 16 + n;
        while (num != 0){
            bin[ctr--] = num % 2;
            num /= 2;
        }
        return bin;
    }
/** Function to print array **/
    public void display(int[] P, char ch){
        System.out.print("\n"+ ch +" : ");
for (int i = 0; i < P.length; i++){
            if (i == 4)
                System.out.print(" ");
            if (i == 8)
                System.out.print(" ");
            System.out.print(P[i]);
        } }
 /** Main function **/
    public static void main (String[] args){
     Scanner scan = new Scanner(System.in);
     Booth b = new Booth();
     System.out.println("Enter two integer numbers -");
        int n1 = scan.nextInt();
        int n2 = scan.nextInt();
        int result = b.multiply(n1, n2);
        System.out.println("\n\nResult : "+ n1 +" * "+ n2 +" = "+ result);
    } }
```