Block Swap Algorithm

```
import java.util.*;

class MaxSubarray{


 public static void swap(int arr[], int a, int b, int r){
        for(int i = 0 ; i < r ; i++){
            int temp = arr[a + i];
            arr[a + i] = arr[b + i];
            arr[b + i] = temp;
        }

    }


public static void leftRotate(int arr[], int r){
        int n = arr.length;


 if(r == 0 || r == n) return;

        int i = r;
        int j = n - r;

 while (i != j){
            if(i < j){
                swap(arr, r-i, r+j-i, i);
                j = j - i;
            }
            else{
                swap(arr, r-i, r, j);
                i = i - j;
            }
        }


        swap(arr, r-i, r, i);
    }

    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();

        System.out.println("Enter the number of rotations");
        int no_of_rotations = s.nextInt();

        leftRotate(arr, no_of_rotations);

        System.out.println("Array Elements after rotating : ");
        for(int i = 0 ; i < n ; i++){
            System.out.print(arr[i] + " ");
        }
    }
}
```

Sample Input:

```
5

12345

2
```

Output:

```
Array Elements after rotating:
3 4 5 1 2
```

Maximum Product Subarray

```
class MaxProduct
{
    static int maxprod(int [] nums)
    {
        int length = nums.length;
        int left=0,right=0,result=nums[0];
        for(int i=0; i<length;i++)
        {
            left=(left==0 ? 1:left) * nums[i];
            right=(right==0 ? 1:right) * nums[length-1-i];
            int max = Math.max(left,right);
            result=Math.max(result,max);
        }
        return result;
    }
```

```
 public static void main(String args[])
 {
    int arr[] = {-1, -3, -10, 0, 60};
    System.out.println("Maximum Sub array product is "+ maxprod(arr));
 }
}
```

Sample Input:

6, -3, -10, 0, 2

Sample Output:

180

Maximum Sum of an Hourglass

```
import java.util.*;
class hourglass
{
  public static void main(String[]args)
  {
    Scanner scan = new Scanner(System.in);

    System.out.print("Enter the number of rows: ");
    int rows = scan.nextInt();

    System.out.print("Enter the number of columns: ");
    int columns = scan.nextInt();

    int[][]matrix = new int[rows][columns];
    System.out.println("Enter the elements of the Matrix: ");

    for(int i = 0; i < rows; i++)
    {
      for(int j = 0; j < columns; j++)
      {
        matrix[i][j]=scan.nextInt();
      }
    }
    int sum = 0,max = 0;
    for(int i = 0; i < rows - 2; i++)
    {
      for(int j = 0; j < columns - 2; j++)
      {
        sum = (matrix[i][j] + matrix[i][j + 1] + matrix[i][j + 2]) + (matrix[i + 1][j +
1]) + (matrix[i + 2][j] + matrix[i + 2][j + 1] + matrix[i + 2][j + 2]);

        if(sum > max)
        {
          max = sum;
        }
      }
    }
    System.out.println("The maximum sum in the hourglass is: "+max);
  }
}
```

Sample Input

5

5

1 2 4 5 6

7 5 2 3 6

0 0 0 0 0

7 5 1 3 5

0 0 0 0 0

Sample Output:

27

Maximum Equilibrium

```
import java.io.*;

public class MaxEquilibrium {

    static int findMaxSum(int []arr, int n)
    {

        int []preSum = new int[n];
        int []suffSum = new int[n];

        int ans = Integer.MIN_VALUE;

        preSum[0] = arr[0];
        for (int i = 1; i < n; i++)
            preSum[i] = preSum[i - 1] + arr[i];


        suffSum[n - 1] = arr[n - 1];
        if (preSum[n - 1] == suffSum[n - 1])
            ans = Math.max(ans, preSum[n - 1]);

        for (int i = n - 2; i >= 0; i--)
        {
            suffSum[i] = suffSum[i + 1] + arr[i];

            if (suffSum[i] == preSum[i])
                ans = Math.max(ans, preSum[i]);
        }

        return ans;
    }


    static public void main (String[] args)
    {
        int []arr = { -2, 5, 3, 1, 2, 6, -4, 2 };
        int n = arr.length;

        System.out.println( findMaxSum(arr, n));
    }
}
```

Sample Input

-2, 5, 3, 1, 2, 6, -4, 2

Sample Output:

7

Leaders Array

```
class LeadersInArray
{

    void printLeaders(int arr[], int size)
    {
        for (int i = 0; i < size; i++)
        {
            int j;
            for (j = i + 1; j < size; j++)
            {
                if (arr[i] <=arr[j])
                    break;
            }
            if (j == size)
                System.out.print(arr[i] + " ");
        }
    }

   public static void main(String[] args)
    {
        LeadersInArray lead = new LeadersInArray();
        int arr[] = new int[]{16, 17, 4, 3, 5, 2};
        int n = arr.length;
        lead.printLeaders(arr, n);
    }
}
```

Sample Input

16 17 4 3 5 2

Sample Output

2 5 17

Majority Element

```
import java.util.*;
public class Main
{
static void maj(int arr[], int n)
{
    int c = 0;
    int index = -1;
    for(int i = 0; i < n; i++)
    {
        int count = 0;
        for(int j = 0; j < n; j++)
        {
            if(arr[i] == arr[j])
            count++;
        }
        if(count > c)
        {
            c = count;
            index = i;
        }
    }
    if (c > n/2)
    System.out.println (arr[index]);
    else
    System.out.println ("No Majority Element");
}

    public static void main (String[] args) {
        Scanner s= new Scanner(System.in);
        System.out.println("Enter length of the array");
        int l=s.nextInt();
        System.out.println("Enter the elements of the array");
        int[] arr= new int[l];
        for(int i=0; i<l; i++)
            arr[i]=s.nextInt();
        System.out.println("Majority Element is: ");
        maj(arr, l);
    }
}
```

Sample input

```
5
1 3 4 1 1
```

Sample Output:

```
1
```