Lesson:

Problems on Array-3

PW SKILLS

Pre-Requisites

- Arrays in Java

List of Concepts Involved

- Problems based on a two pointers approach.

Pattern: Two Pointers
In this approach, we declare two pointers and perform operations on the array. The two pointers are usually declared one from the start and the other one from the end and we keep incrementing and decrementing them according to the problem.

Problem 1: Sort an Array consisting of only 0s and 1s.

Input
[0,1,1,0,1,1,0]

Output
[0,0,0,1,1,1,1]

Code:

```
import java.io.*;
import java.util.*;
public class Target {
  public static void sortZeroesAndOne(int[] a) {
    int n = a.length;
    int i = 0;
    int j = n - 1;
    while (i < j) {
      if (a[i] == 1 && a[j] == 0) {
        a[i] = 0;
        a[j] = 1;
        i++;
        j--;
      }
      if (a[i] == 0) i++;
      if (a[j] == 1) j--;
    }
  }
    public static void main(String[] args){
      int[] a={0,1,1,0,1,1,0};
      sortZeroesAndOne(a);
      for (int element: a) {
        System.out.print(element + " ");
      }
    }
}
```

Explanation: Take two pointers one from start and another from end, and if the starting pointer's element has value 1 and ending pointer's value is 0, then we need to swap them as we want to sort the array. Check this at every iteration and change values if it satisfies the condition else if it is fine then just increment the starting pointer and decrement the ending pointer as shown.

Problem 2: Given an array of integers 'a', move all the even integers at the beginning of the array followed by all the odd integers. The relative order of odd or even integers does not matter. Return any array that satisfies the condition.

Input :
[1,2,3,4,5]

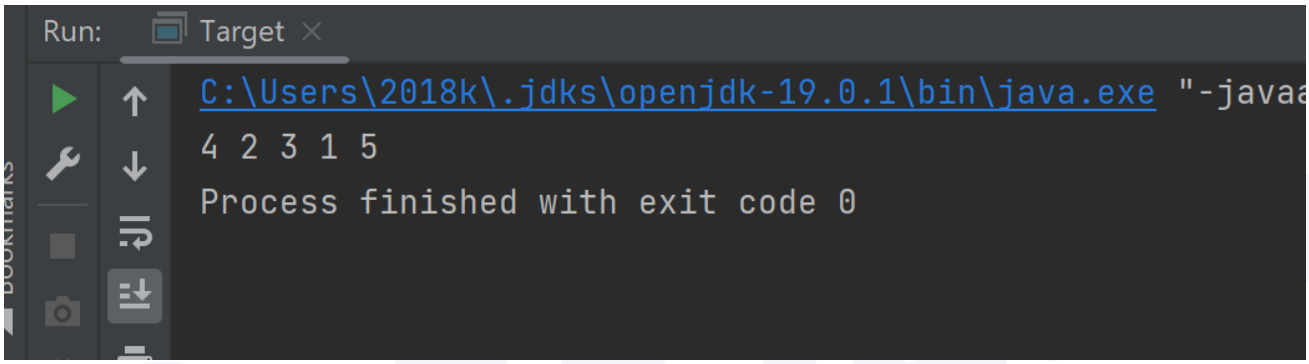
Output :
[4,2,3,1,5]

Explanation:
We need to keep all the even parity values at the beginning followed by the odd values so whenever we encounter the case of odd value at the start pointer and even value at the end pointer we swap them. Also we keep on incrementing the pointers if the values are already at the correct position.

Code:

```
import java.io.*;
import java.util.*;
public class Target {
  public static int[] sortArrayByParity(int[] a) {
    int i = 0, j = a.length-1;
    while(i < j) {
      if(a[i] % 2 == 1 && a[j] % 2 == 0) {
      // swapping the values of a[i] and a[[j]
        int temp=a[i];
        a[i]=a[j];
        a[j]=temp;
        i++;
        j--;
    }
```

```
      if(a[i] % 2 == 0) i++;
      if(a[j] % 2 == 1) j--;
    }
    return a;
  }
  public static void main(String[] args){
      int[] a={1,2,3,4,5};
      int[] ans=sortArrayByParity(a);
      for (int element: ans) {
        System.out.print(element + " ");
      }
    }
}
```



Explanation: Take two pointers one from start and another from end, and if the starting pointer's element has odd value and ending pointer's value is even, then we need to swap them as we want to sort the array. Check this at every iteration and change values if it satisfies the condition else if it is fine then just increment the starting pointer and decrement the ending pointer as shown.

Problem 3: Given an integer array 'a' sorted in non-decreasing order, return an array of the squares of each number sorted in non-decreasing order.

Input :
[-10,-3,2,5,6]

Output :
[4,9,25,36,100]

Explanation:
Note that the square of a negative number is always positive so we can just compare the absolute values. Given that the array is sorted, we can just take the higher absolute value from the first or last of the array and move our pointers accordingly. In the end we need to reverse our answer as we are taking the higher values first so it will be in decreasing order and we need to return our answer in non-decreasing order.

Code:

```
import java.io.*;
import java.util.*;
public class Target {
  public int[] sortedSquares(int[] a) {
    int n = a.length, i = 0, j = n-1, k = 0;
    int[] ans = new int[n];
    while(i <= j) {
      if(Math.abs(a[i]) < Math.abs(a[j])) {
        a[j] *= a[j];
        ans[k++] = (a[j--]);
      } else {
        a[i] *= a[i];
        ans[k++] = (a[i++]);
      }
    }
    reverse(ans, 0, ans.length - 1);
    return ans;
  }
  private void reverse(int[] a, int i, int j){
    int temp = 0;
    while(i < j){
      temp = a[i];
      a[i] = a[j];
      a[j] = temp;
      i++;
      j--;
    }
  }
  public static void main(String[] args){
      int[] a={-10,-3,2,5,6};
      Target obj1 = new Target();
      int[] ans=obj1.sortedSquares(a);
      for (int element: ans) {
        System.out.print(element + " ");
      }
    }
}
```

Upcoming Class Teasers:

- Problems based on Prefix Sums