1) Print even and odd numbers in increasing order using two threads in Java

Given an integer N, the task is to write Java Program to print the first N natural numbers in increasing order using two threads.
Prerequisite: Multithreading
Examples:
Input: N = 10
Output: 1 2 3 4 5 6 7 8 9 10
Input: N = 18
Output: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
Recommended: Please try your approach on {IDE} first, before moving on to the solution.
Approach: The idea is to create two threads and print even numbers with one thread and odd numbers with another thread. Below are the steps:
Create two threads T1 and T2 using the below syntax, where T1 and T2 are used to print odd and even numbers respectively.

```
Thread T1 = new Thread(new Runnable() {
        public void run() { mt.printEvenNumber(); }
});
Thread T2 = new Thread(new Runnable() {
        public void run() { mt.printOddNumber(); }
});
```

where, printOddNumber() is used to print all the odd numbers till N, printEvenNumber() is used to print all the even numbers till N.
Maintain a global counter variable and start both threads using the below function:

```
T1.start(); T2.start();
```

If the counter is even in the Thread T1, then wait for the thread T2 to print that even number. Otherwise, print that odd number, increment the counter and notify to the Thread T2 using the function notify().
If the counter is odd in the Thread T2, then wait for the thread T1 to print that odd number. Otherwise, print that even number, increment the counter and notify the Thread T1 using the function notify().
Below is the implementation of the above approach:
Java

```
// Java program for the above approach

public class GFG {

   // Starting counter
   int counter = 1;

   static int N;

   // Function to print odd numbers
   public void printOddNumber()
   {
      synchronized (this)
      {
         // Print number till the N
```

```
        while (counter < N) {

            // If count is even then print
            while (counter % 2 == 0) {

                // Exception handle
                try {
                    wait();
                }
                catch (
                    InterruptedException e) {
                    e.printStackTrace();
                }
            }

            // Print the number
            System.out.print(counter + " ");

            // Increment counter
            counter++;

            // Notify to second thread
            notify();
        }
    }
}

// Function to print even numbers
public void printEvenNumber()
{
    synchronized (this)
    {
        // Print number till the N
        while (counter < N) {

            // If count is odd then print
            while (counter % 2 == 1) {

                // Exception handle
                try {
                    wait();
                }
                catch (
                    InterruptedException e) {
                    e.printStackTrace();
                }
            }
```

```
            // Print the number
            System.out.print(
               counter + " ");

            // Increment counter
            counter++;

            // Notify to 2nd thread
            notify();
         }
      }
   }

   // Driver Code
   public static void main(String[] args)
   {
      // Given Number N
      N = 10;

      // Create an object of class
      GFG mt = new GFG();

      // Create thread t1
      Thread t1 = new Thread(new Runnable() {
         public void run()
         {
            mt.printEvenNumber();
         }
      });

      // Create thread t2
      Thread t2 = new Thread(new Runnable() {
         public void run()
         {
            mt.printOddNumber();
         }
      });

      // Start both threads
      t1.start();
      t2.start();
   }
}
```

Output:

1 2 3 4 5 6 7 8 9 10

2) Java Program to Rotate all odd numbers right and all even numbers left in an Array of 1 to N

Given a permutation arrays A[] consisting of N numbers in range [1, N], the task is to left rotate all the even numbers and right rotate all the odd numbers of the permutation and print the updated permutation.
Note: N is always even.
Examples:
Input: A = {1, 2, 3, 4, 5, 6, 7, 8}
Output: {7, 4, 1, 6, 3, 8, 5, 2}
Explanation:
Even element = {2, 4, 6, 8}
Odd element = {1, 3, 5, 7}
Left rotate of even number = {4, 6, 8, 2}
Right rotate of odd number = {7, 1, 3, 5}
Combining Both odd and even number alternatively.
Input: A = {1, 2, 3, 4, 5, 6}
Output: {5, 4, 1, 6, 3, 2}

Approach:
It is clear that the odd elements are always on even index and even elements are always laying on odd index.
To do left rotation of even number we choose only odd indices.
To do right rotation of odd number we choose only even indices.
Print the updated array.
Below is the implementation of the above approach:
Java

```
// Java program to implement
// the above approach

import java.io.*;
import java.util.*;
import java.lang.*;

class GFG {

// function to left rotate
static void left_rotate(int[] arr)
{
int last = arr[1];
for (int i = 3;
i < arr.length;
i = i + 2) {
arr[i - 2] = arr[i];
}
arr[arr.length - 1] = last;
}

// function to right rotate
static void right_rotate(int[] arr)
{
int start = arr[arr.length - 2];
for (int i = arr.length - 4;
i >= 0;
i = i - 2) {
arr[i + 2] = arr[i];
}
arr[0] = start;
```

```
}

// Function to rotate the array
public static void rotate(int arr[])
{
left_rotate(arr);
right_rotate(arr);
for (int i = 0; i < arr.length; i++) {
System.out.print(arr[i] + " ");
}
}

// Driver code
public static void main(String[] args)
{
int arr[] = { 1, 2, 3, 4, 5, 6 };

rotate(arr);
}
}
```

Output:
5 4 1 6 3 2
Time Complexity: O(N)
Auxiliary Space: O(1)

3) Program for multiple catch blocks

Example 1
Let's see a simple example of java multi-catch block.
MultipleCatchBlock1.java

```
public class MultipleCatch {

public static void main(String[] args) {

try{
int a[]=new int[5];
a[5]=30/0;
}
catch(ArithmeticException e)
{
System.out.println("Arithmetic Exception occurs");
}
catch(ArrayIndexOutOfBoundsException e)
{
System.out.println("ArrayIndexOutOfBounds Exception occurs");
}
catch(Exception e)
{
System.out.println("Parent Exception occurs");
}
System.out.println("rest of the code");
```

```
}
}
```
Output:
```
Arithmetic Exception occurs
rest of the code
```
Example 2
MultipleCatchBlock2.java
```
public class MultipleCatchBlock2 {

public static void main(String[] args) {

try{
int a[]=new int[5];

System.out.println(a[10]);
}
catch(ArithmeticException e)
{
System.out.println("Arithmetic Exception occurs");
}
catch(ArrayIndexOutOfBoundsException e)
{
System.out.println("ArrayIndexOutOfBounds Exception occurs");
}
catch(Exception e)
{
System.out.println("Parent Exception occurs");
}
System.out.println("rest of the code");
}
}
```

4) Java Program for practising various String methods given below

No.	Method	Description
1	char charAt(int index)	It returns char value for the particular index
2	int length()	It returns string length
3		
4		
5	String substring(int beginIndex)	It returns substring for given begin index.
6	String substring(int beginIndex, int endIndex)	It returns substring for given begin index and end index.
7	boolean contains(CharSequence s)	It returns true or false after matching the sequence of char value.
8		

10	boolean equals(Object another)	It checks the equality of string with the given object.
11	boolean isEmpty()	It checks if string is empty.
12	String concat(String str)	It concatenates the specified string.
13	String replace(char old, char new)	It replaces all occurrences of the specified char value.
14	String replace(CharSequence old, CharSequence new)	It replaces all occurrences of the specified CharSequence.
15		
16		
17		
18	String intern()	It returns an interned string.
21	int indexOf(String substring)	It returns the specified substring index.
22		
23	String toLowerCase()	It returns a string in lowercase.
24		
25	String toUpperCase()	It returns a string in uppercase.
26		
27	String trim()	It removes beginning and ending spaces of this string.

5) Program for Map vs Set Vs List

```
// A Java Program to implement the addition of elements in the List
import java.util.*;
public class Javalist {
   public static void main(String args[])
   {

      // here create a List
      List<String> al = new ArrayList<>();

      // Now add the elements in the List
      al.add("BMW");
      al.add("Hundai");
```

```
        al.add("Toyota");
        al.add("Swift");

        // Iterating the List
        // element using for-each loop
        for (String cars : al)
            System.out.println(cars);
    }
}
Output:
BMW
Hundai
Toyota
Swift
// A Java program to illustrate a Map.
 // add elements using Map
import java.util.*;
class JavaMap {
    public static void main(String args[])
    {

        // Creating object for a Map.
        Map<String, Integer> map
            = new HashMap< String, Integer>();

        // Adding Elements using Map.
        map.put( "Rajat", 101);
        map.put("Shyam", 102);
        map.put("Rahul", 103);
        map.put("Krishna", 104);
        // here, elements may traverse in any order
        for (Map.Entry m : map.entrySet()) {
            System.out.println(m.getKey() + " "
                               + m.getValue());
        }
    }
}
Output:
Rahul 103
Shyam 102
Krishna 104
Rajat 101

// A Java program to illustrate a Set.
// add Elements using Set.

import java.util.*;
public class JavaSet {

    public static void main(String[] args)
```

```
    {
        // Set demonstration via using HashSet
        Set<String> Set = new HashSet<String>();

        // Adding some Elements
        Set.add("Java");
        Set.add("Python");
        Set.add("DBMS");
        Set.add("Machine Learning");
        Set.add("Operating System");

        // Here Set follows unordered way.
        System.out.println(Set);
    }
}
```

Output:

[Java, Operating System, DBMS, Machine Learning, Python]