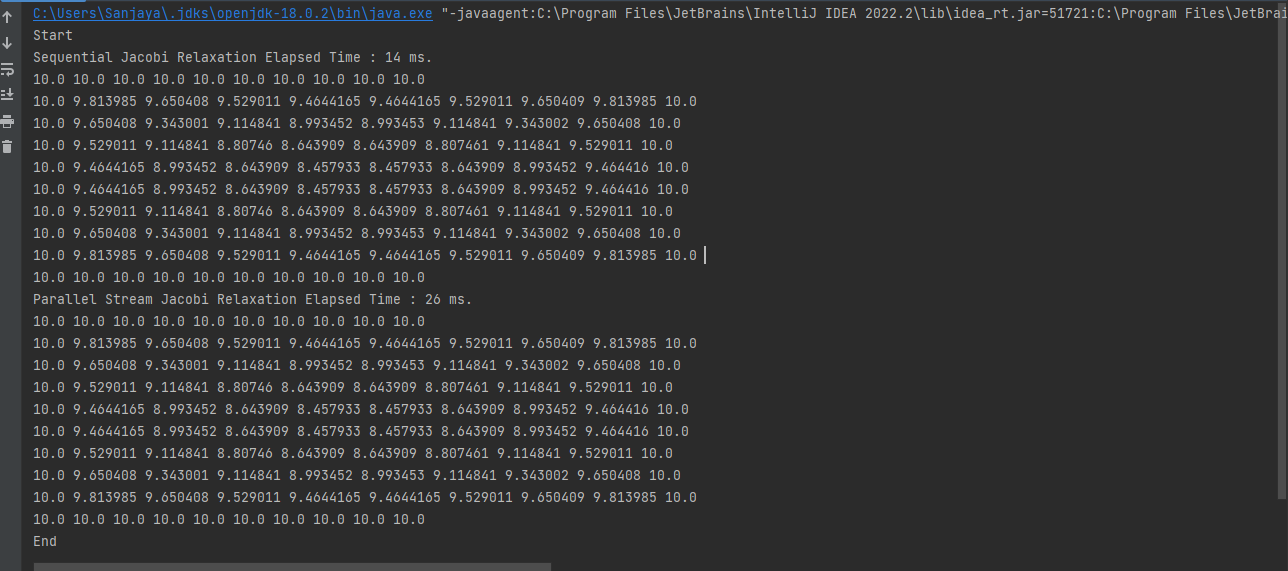
**Code:**

package jacobi;  
  
import java.util.Date;  
import java.util.stream.IntStream;  
  
public class Main {  
  
 private static float *tolerance* = 0.1f;  
 private static int *n* = 10;  
 private static float *A*[][];  
  
 public static long *sequentialTime*;  
 public static long *parallelTime*;  
  
 static boolean *done* = false;  
  
 public static void main(String[] args) {  
 *A* = new float[*n*][*n*];  
  
 System.*out*.println("Start");  
  
 *sequential*();  
  
 *parallelStream*();  
  
 System.*out*.println("End\n");  
 }  
  
 private static void loadData() {  
 for (int i = 1; i < *n*-1; i++)  
 for (int j = 1; j < *n*-1; j++)  
 *A*[i][j] = 0;  
 for (int i = 0; i < *n*; i++) {  
 *A*[i][0] = 10; *A*[i][*n*-1] = 10;  
 *A*[0][i] = 10; *A*[*n*-1][i] = 10;  
 }  
  
 }  
  
 private static void sequential() {  
 *loadData*();  
 Date start = new Date();  
 do {  
  
 float rtn1[][] = IntStream.*range*(0,*n*)  
 .mapToObj(i -> {  
 float arr[] = new float[*n*];  
 for (int j = 0; j < *n*; j++) {  
 if (i == 0 || i == *n*-1 || j==0|| j==*n*-1) {  
 arr[j] = *A*[i][j];  
 } else {  
 arr[j] = (*A*[i - 1][j] + *A*[i + 1][j] + *A*[i][j - 1] + *A*[i][j + 1]) / 4.0f;  
 }  
 }  
 return arr;  
 }).toArray(float[][]::new);  
  
 *done* = true;  
 *A* = IntStream.*range*(0,*n*)  
 .mapToObj(i -> {  
 float arr[] = new float[*n*];  
 for (int j = 0; j < *n* ; j++) {  
 if (Math.*abs*(*A*[i][j] - rtn1[i][j]) > *tolerance*)  
 *done* = false;  
 arr[j] = rtn1[i][j];  
 }  
 return arr;  
 }).toArray(float[][]::new);  
  
  
 } while (!*done*);  
 Date end = new Date();  
 *sequentialTime* = end.getTime() - start.getTime();  
 System.*out*.printf("Sequential Jacobi Relaxation Elapsed Time : %d ms.\n", *sequentialTime*);  
 *printMethod*();  
 }  
  
 private static void printMethod() {  
 for(int i=0;i<*A*.length;i++) {  
 for (int j=0;j<*A*.length;j++) {  
 System.*out*.print(*A*[i][j] + " ");  
 }  
 System.*out*.println();  
 }  
 }  
  
 private static void parallelStream() {  
 *loadData*();  
 Date start = new Date();  
 do {  
  
 float rtn1[][] = IntStream.*range*(0,*n*)  
 .parallel()  
 .mapToObj(i -> {  
 float arr[] = new float[*n*];  
 for (int j = 0; j < *n*; j++) {  
 if (i == 0 || i == *n*-1 || j==0|| j==*n*-1) {  
 arr[j] = *A*[i][j];  
 } else {  
 arr[j] = (*A*[i - 1][j] + *A*[i + 1][j] + *A*[i][j - 1] + *A*[i][j + 1]) / 4.0f;  
 }  
 }  
 return arr;  
 }).toArray(float[][]::new);  
  
 *done* = true;  
 *A* = IntStream.*range*(0,*n*)  
 .parallel()  
 .mapToObj(i -> {  
 float arr[] = new float[*n*];  
 for (int j = 0; j < *n*; j++) {  
 if (Math.*abs*(*A*[i][j] - rtn1[i][j]) > *tolerance*)  
 *done* = false;  
 arr[j] = rtn1[i][j];  
 }  
 return arr;  
 }).toArray(float[][]::new);  
  
  
 } while (!*done*);  
 Date end = new Date();  
 *parallelTime* = end.getTime() - start.getTime();  
 System.*out*.printf("Parallel Stream Jacobi Relaxation Elapsed Time : %d ms.\n", *parallelTime*);  
 *printMethod*();  
 }  
}

**output:**

****

Start

Sequential Jacobi Relaxation Elapsed Time : 14 ms.

10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0

10.0 9.813985 9.650408 9.529011 9.4644165 9.4644165 9.529011 9.650409 9.813985 10.0

10.0 9.650408 9.343001 9.114841 8.993452 8.993453 9.114841 9.343002 9.650408 10.0

10.0 9.529011 9.114841 8.80746 8.643909 8.643909 8.807461 9.114841 9.529011 10.0

10.0 9.4644165 8.993452 8.643909 8.457933 8.457933 8.643909 8.993452 9.464416 10.0

10.0 9.4644165 8.993452 8.643909 8.457933 8.457933 8.643909 8.993452 9.464416 10.0

10.0 9.529011 9.114841 8.80746 8.643909 8.643909 8.807461 9.114841 9.529011 10.0

10.0 9.650408 9.343001 9.114841 8.993452 8.993453 9.114841 9.343002 9.650408 10.0

10.0 9.813985 9.650408 9.529011 9.4644165 9.4644165 9.529011 9.650409 9.813985 10.0

10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0

Parallel Stream Jacobi Relaxation Elapsed Time : 26 ms.

10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0

10.0 9.813985 9.650408 9.529011 9.4644165 9.4644165 9.529011 9.650409 9.813985 10.0

10.0 9.650408 9.343001 9.114841 8.993452 8.993453 9.114841 9.343002 9.650408 10.0

10.0 9.529011 9.114841 8.80746 8.643909 8.643909 8.807461 9.114841 9.529011 10.0

10.0 9.4644165 8.993452 8.643909 8.457933 8.457933 8.643909 8.993452 9.464416 10.0

10.0 9.4644165 8.993452 8.643909 8.457933 8.457933 8.643909 8.993452 9.464416 10.0

10.0 9.529011 9.114841 8.80746 8.643909 8.643909 8.807461 9.114841 9.529011 10.0

10.0 9.650408 9.343001 9.114841 8.993452 8.993453 9.114841 9.343002 9.650408 10.0

10.0 9.813985 9.650408 9.529011 9.4644165 9.4644165 9.529011 9.650409 9.813985 10.0

10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0

End