# Analisis de algoritmos
# TP0 Repaso

Ovaillos, Cristopher Jason
Occhi, Tomás Ignacio
Medel Severini, Joaquín
Fernandez Loenard
Perez Penas, Facundo

Agosto 2024

## 1 Entrada/Salida con Java

- El siguiente codigo fue utilizado en los distintos programas, para llamar a metodos que seran de utilidad.

```java
   package ejemplos_InOut;
/*
 * @author nacho
 */
public class Aleatorio {
    public static double doubleAleatorio(int min, int max) {
        return Math.random() * (max - min) + min;
    }

    public static int intAleatorio(int min, int max) {
        return (int) doubleAleatorio(min, max);
    }

    public static char charAleatorio() {
        char a;
        if (Math.random() >= 0.5) {
            // letras mayusculas
            a = (char) intAleatorio(97, 123);
        } else {
            // letras minusculas
            a = (char) intAleatorio(65, 91);
        }
        return a;
    }

    public static char charAleatorioAux() {
        char a;
        double rand = Math.random();

        if (rand < 1.0 / 3.0) {
            // números
            a = (char) intAleatorio(48, 58); // ASCII 48-57 for '0'-'9'
        } else if (rand < 2.0 / 3.0) {
            // letras minusculas
```

```
            a = (char) intAleatorio(97, 123); // ASCII 97-122 for 'a'-'z'
        } else {
            // letras mayusculas
            a = (char) intAleatorio(65, 91); // ASCII 65-90 for 'A'-'Z'
        }

        return a;
    }

    public static String stringAleatorio(int tam) {
        String res = "";
        for (int i = 0; i < tam; i++) {
            res = res + charAleatorio();
        }
        return res;
    }

    public static String stringAleatorioConNumeros(int tam) {
        String res = "";
        for (int i = 0; i < tam; i++) {
            res = res + charAleatorioAux();
        }
        return res;
    }

    public static String nombreAleatorio() {
        // ejemplo para definir un valor aleatorio entre varias posibilidades
        // en este caso nombres
        String arr[] = { "Juan", "Carlos", "Pedro", "Tito", "Cacho" };
        return arr[intAleatorio(0, arr.length - 1)];
    }

}
```

- 1. Utilizando un archivo de entrada que contenga un texto con varias palabras y varias l´ ıneas de texto y uno o m´as espacios en blanco entre ellas, generar sinEspacios.txt con el mismo texto pero eliminando todos los espacios en blanco.

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

        String nombreArchivoEntrada = "src/ejemplos_InOut/Ejercicios/entrada.txt";
        String nombreArchivoSalida = "src/ejemplos_InOut/Ejercicios/sinEspacios.txt";

        String linea = null;

        try {

            FileReader lectorArchivo = new FileReader(nombreArchivoEntrada);

            FileWriter escritorArchivo = new FileWriter(nombreArchivoSalida);

            BufferedReader bufferLectura = new BufferedReader(lectorArchivo);
            BufferedWriter bufferEscritura = new BufferedWriter(escritorArchivo);

         //readLine() es un metodo de la clase BufferedReader que lee una linea completa de
         ↪  texto desde un archivo o una fuente de datos.
            while ((linea = bufferLectura.readLine()) != null) {
```

```
                String lineaSinEspacio = linea.replaceAll(" ", "");
                bufferEscritura.write(lineaSinEspacio);
                bufferEscritura.newLine();
            }


            bufferLectura.close();
            bufferEscritura.close();
        }
        catch (FileNotFoundException ex) {
            System.err.println(ex.getMessage() + "\nSignifica que el archivo del "
                    + "que queriamos leer no existe.");
        }
        catch (IOException ex) {
            System.err.println("Error leyendo o escribiendo en algun archivo.");
        }
    }
```

- 2 con el mismo archivo de entrada, generar el archivo lineasImpares.txt con solo las lineas impares del texto.

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

        String nombreArchivoEntrada = "src/ejemplos_InOut/Ejercicios/entrada.txt";
        String nombreArchivoSalida = "src/ejemplos_InOut/Ejercicios/lineasImpares.txt";

        String linea = null;
        int line=1;
        try {
            FileReader lectorArchivo = new FileReader(nombreArchivoEntrada);
            FileWriter escritorArchivo = new FileWriter(nombreArchivoSalida);
            BufferedReader bufferLectura = new BufferedReader(lectorArchivo);
            BufferedWriter bufferEscritura = new BufferedWriter(escritorArchivo);
         //readLine() es un metodo de la clase BufferedReader que lee una linea completa de
         ↪  texto desde un archivo o una fuente de datos.
            while ((linea = bufferLectura.readLine()) != null) {

                if ((line % 2) !=0) {
                    bufferEscritura.write(linea);
                    bufferEscritura.newLine();
                }
                line++;

            }


            bufferLectura.close();
            bufferEscritura.close();
        }
        catch (FileNotFoundException ex) {
            System.err.println(ex.getMessage() + "\nSignifica que el archivo del "
                    + "que queriamos leer no existe.");
        }
        catch (IOException ex) {
            System.err.println("Error leyendo o escribiendo en algun archivo.");
        }
```

```
    }
```

- 3. Generar un archivo de texto que contenga 100 numeros reales (double o float) generados aleatoriamente con valores entre -100 y 100.

```
public class Ejercicio_3 {
static final int CANTNUMEROS = 100;
static final int MIN_VALOR = -100;
static final int MAX_VALOR = 100;
    static final String NOMBRE_ARCHIVO = "src/ejemplos_InOut/Ejercicios/numeros.txt";

private static void generarArchivo() {
        try {
                BufferedWriter buff = new BufferedWriter(new
                ↪  FileWriter(NOMBRE_ARCHIVO));

                for (int i = 0; i < CANTNUMEROS; i++) {
                        double num = Aleatorio.doubleAleatorio(MIN_VALOR, MAX_VALOR);
                        buff.write(num + "\n");
                        System.out.println(num);
                }
                buff.close();
        } catch (FileNotFoundException ex) {
        System.err.println(ex.getMessage() + "\nSignifica que el archivo del que
        ↪  queriamos leer no existe.");
    } catch (IOException ex) {
        System.err.println("Error leyendo o escribiendo en algun archivo.");
    }
}

private static double[] leerArchivo() {
        double[] arreglo = new double[CANTNUMEROS];
        try {
                BufferedReader buff = new BufferedReader(new
                ↪  FileReader(NOMBRE_ARCHIVO));
                for (int i = 0; i < CANTNUMEROS; i++) {
                        // Convertir cada línea leída en un double
                        arreglo[i] = Double.parseDouble(buff.readLine());
                }
                buff.close();
        } catch (FileNotFoundException ex) {
        System.err.println(ex.getMessage() + "\nSignifica que el archivo del que
        ↪  queriamos leer no existe.");
    } catch (IOException ex) {
        System.err.println("Error leyendo o escribiendo en algun archivo.");
    }
        return arreglo;
}

private static void mostrarLosNMasGrandesDelArchivo(int n, double[] arreglo) {
    Arrays.sort(arreglo); // Reemplazar esta llamada por sus propios metodos de
    ↪  ordenamiento.
    int tope = Math.max(0, arreglo.length - n);
    System.out.println("Los " + n + " números más grandes son:");
    for (int i = arreglo.length - 1; i >= tope; i--) {
            System.out.println(arreglo[i]);
    }
```

```
    }

    public static void main(String[] args) {
            // Generar el archivo con números aleatorios
            generarArchivo(); // Una vez que ya tenemos un archivo generado, no hace falta
            ↪  generar uno nuevo.

            // Leer los números del archivo generado
            double[] arregloGenerado = leerArchivo();

            // Tomar la hora del sistema en nanosegundos
            long inicio = System.nanoTime();

            // Mostrar los N números más grandes del arreglo
            mostrarLosNMasGrandesDelArchivo(CANTNUMEROS, arregloGenerado);

            long fin = System.nanoTime();
            System.out.println("Se tardo: " + (fin - inicio) + " nanosegundos en obtener los
            ↪  numeros mas grandes");
    }
```

- 4. Generar un archivo de texto con cadenas aleatorias de 10 caracteres alfanumericos (0-9, a-z, A-Z).

```
    static final int CANT_CADENAS = 10000; // Cantidad de cadenas a generar
    static final int LONGITUD_CADENA = 10; // Longitud de cada cadena alfanumérica
    static final String NOMBRE_ARCHIVO = "src/ejemplos_InOut/Ejercicios/cadenas.txt"; //
    ↪  Archivo de salida

    public static void main(String[] args) {

        try {
            BufferedWriter bufferEscritura = new BufferedWriter(new
            ↪  FileWriter(NOMBRE_ARCHIVO));

            for (int i = 0; i < CANT_CADENAS; i++) {
                bufferEscritura.write(Aleatorio.stringAleatorioConNumeros(LONGITUD_CADENA) +
                ↪  "\n");
            }

            bufferEscritura.close();
        } catch (IOException ex) {
            System.err.println("Error escribiendo en el archivo: " + ex.getMessage());
        }
    }
```

- 5. Generar un archivo con numeros aleatorios de los numeros del 1 al 1000 sin que los elementos se repitan.

```
package ejemplos_InOut.Ejercicios;

import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.util.HashSet;
import java.util.Scanner;
```

```java
import java.util.Set;
import ejemplos_InOut.Aleatorio;//importo un archivo, con los metodos que utilizaremos

public class Ejercicio_5 {
    static final int CANT_NUMEROS = 50; // Cantidad de cadenas a generar

    static final String NOMBRE_ARCHIVO = "src/ejemplos_InOut/Ejercicios/ejercicio5.txt"; //
    ↪ Archivo de salida

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int rangoInferior = 1;
        int rangoSuperior = 1000;
        Set<Integer> set = new HashSet<>();
        while (set.size()<=CANT_NUMEROS) {
            int random= Aleatorio.intAleatorio(rangoInferior, rangoSuperior);
            set.add(random);
        }
        try {
            BufferedWriter bufferEscritura = new BufferedWriter(new
            ↪ FileWriter(NOMBRE_ARCHIVO));
            bufferEscritura.write(set.toString());
            bufferEscritura.close();
        } catch (IOException ex) {
            System.err.println("Error escribiendo en el archivo: " + ex.getMessage());
        }
    }
}
```

## 2 Repaso de Algoritmia

- **1. Realiza detenidamente una traza al siguiente programa y muestra cuál sería la salida por pantalla:**

```
ALGORITMO ej1
    VARIABLES
    suma, i, j: ENTERO
    PARA i ← 1 HASTA 4 HACER
        PARA j ← 3 HASTA 0 PASO -1 HACER
            suma ← i * 10 + j
            escribir(suma)
        FIN PARA
    FIN PARA
FIN ALGORITMO
```

| suma | i | j | PANTALLA |
|---|---|---|---|
| ~~13~~ | ~~1~~ | ~~3~~ | ~~13~~ |
| ~~12~~ | ~~1~~ | ~~2~~ | ~~12~~ |
| ~~11~~ | ~~1~~ | ~~1~~ | ~~11~~ |
| ~~10~~ | ~~1~~ | ~~0~~ | ~~10~~ |
| ~~23~~ | ~~2~~ | ~~3~~ | ~~23~~ |
| ~~22~~ | ~~2~~ | ~~2~~ | ~~22~~ |
| ~~21~~ | ~~2~~ | ~~1~~ | ~~21~~ |
| ~~20~~ | ~~2~~ | ~~0~~ | ~~20~~ |
| ~~33~~ | ~~3~~ | ~~3~~ | ~~33~~ |
| ~~32~~ | ~~3~~ | ~~2~~ | ~~32~~ |
| ~~31~~ | ~~3~~ | ~~1~~ | ~~31~~ |
| ~~30~~ | ~~3~~ | ~~0~~ | ~~30~~ |
| ~~43~~ | ~~4~~ | ~~3~~ | ~~43~~ |
| ~~42~~ | ~~4~~ | ~~2~~ | ~~42~~ |
| ~~41~~ | ~~4~~ | ~~1~~ | ~~41~~ |
| ~~40~~ | ~~4~~ | ~~0~~ | ~~40~~ |
| 40 | 4 | 0 | 40 |

- **2. ¿Qué imprime el siguiente programa?**

```
1        class Ejercicio {
2        public static void main (String [] args){
3        char [] matriz ← {'e','u','o','i','a'};
4        metodo(matriz);
5        for (int i ← 0; i<matriz.length; i++){
6            System.out.print(matriz[i]);
7        }
8    }
9    public static void metodo (char [] vocales){
10       char aux;
11
12       for (int i ← 1; i<vocales.length; i++){
13           if (vocales[i-1]>vocales[i]){
14               aux ← vocales[i-1];
15               vocales[i-1] ← vocales[i];
16               vocales[i] ← aux;
17           }
18       }
19   }
20 }
```

**Respuesta:**

El codigo realiza una ordenacion parcial de las vocales en el array 'matriz'. La función 'metodo' intercambia los caracteres si el anterior es mayor que el actual, de modo que solo realiza un pase de comparación y ordenación. El resultado es que las vocales se reordenan parcialmente, pero no se obtiene una ordenación completa. La salida del programa ejecutado es: **eoiau**

- **3. Realizar un programa que nos pida un número n y nos diga cuantos números primos existen entre 1 y n.**

```java
package ejemplos_InOut.Ejercicios.Repaso_Algoritmia;
import java.util.Scanner;

public class Ej_3 {

    // metodo para verificar si un num es primo
    public static boolean esPrimo(int numero) {
        boolean esPrimo = true;

        if (numero <= 1) {
            esPrimo = false;
        } else {
            int i = 2;
            //no es necesario verificar todos los posibles divisores [2-numero]
            //basta con que sea de  [2- raiz cuadrada de (numero)]
            int cantDivisores = (int) Math.sqrt(numero);
            while (i <= cantDivisores && esPrimo) {
                if (numero % i == 0) {
                    esPrimo = false;  // si es divisor, entonces no es primo
                }
                i++;
            }
        }

        return esPrimo;
    }


    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Ingresa un número n: ");
        int n = scanner.nextInt();

        int contadorPrimos = 0;

        // Contamos los números primos entre 1 y n
        for (int i = 1; i <= n; i++) {
            if (esPrimo(i)) {
                contadorPrimos++;
            }
        }

        System.out.println("Hay " + contadorPrimos + " numeros primos entre 1 y " + n +
        ↪  ".");

        scanner.close();
    }
}
```

- **4. Realizar un juego para adivinar un numero: Generar un número entero en forma aleatoria dentro de un rango, y luego, ir pidiendo numeros indicando mayor o menor segun sea mayor o menor con respecto a n. El proceso termina cuando el usuario acierta.**

```java
package ejemplos_InOut.Ejercicios.Repaso_Algoritmia;
import java.util.Scanner;
import ejemplos_InOut.Aleatorio;//importo un archivo, con los metodos que utilizaremos
public class Ej_4 {

    public static void main(String[] args) {
        //creo un objeto Scanner para leer la entrada del usuario
        Scanner scanner = new Scanner(System.in);
        //Defino mi rango
        int rangoInferior = 1;
        int rangoSuperior = 100;
        //Con ayuda de Aleatorio.java podemos buscar el numero que sera secreto
        int numSecreto = Aleatorio.intAleatorio(rangoInferior, rangoSuperior);
        //var para almacenar las adivinansas
        int adivinanza = 0;
        System.out.println("Numero entre " + rangoInferior + " y " + rangoSuperior + "
        ↪  generado. ¡Intenta adivinarlo!");
        //System.out.println(numSecreto);
        // Bucle hasta que lo adivine
        while (adivinanza != numSecreto) {
            System.out.print("Introduce un numero: ");
            adivinanza = scanner.nextInt();

            //comparamos el numero ingresado con el numero secreto
            if (adivinanza < numSecreto) {
                System.out.println("El num secreto es mayor.");
            } else if (adivinanza > numSecreto) {
                System.out.println("El num secreto es menor.");
            } else {
                System.out.println("Felicidades!!!!!!! adivinaste el numero.");
            }
        }

        scanner.close();
    }
}
```

- **5. Suponiendo $n \leq 1000000$ y un usuario que siga de forma óptima la lógica del juego y que quiera dar la menor cantidad de pasos hasta adivinar el número: ¿Cuál es el máximo número de intentos que puede necesitar el jugador hasta encontrar un número dentro del intervalo?**

  La mejor estrategia seria ir dividiendo el intervalo por la mitad, para asi quedarse con solo una de las mitades(descartando la otra segun n mayor o menor). De esta manera el numero maximo de intentos sera: En cada iteracion se reduce a la mitad por lo que su orden sera logaritmico

$$numIntentos = log_2(1,000,000) \approx 19.93$$

  Redondeando.... numIntentos=20

- **6. Liste y describa claramente los algoritmos para la resolución del problema de búsqueda que conoce. (Tip: recordar distintas implementaciones de interfaz TablaDeBusqueda).**

- En ABB(Arbol Binario de Busqueda) los elementos se ordenan de tal manera que el arbol sea un arbol balanceado donde el hijo izquierdo y derecho de cada nodo son menores y mayores respectivamente. En el algoritmo de busqueda de ABB en el peor de los casos el numero de intentos hasta encontrar el nodo es de h(altura del arbol). La altura de un arbol ABB esta dada por log(x), x=cantNodos

- En Grafos tenemos busqueda en profundidad y busqueda en anchura. La búsqueda en profundidad explora tan lejos como sea posible a lo largo de cada rama antes de retroceder, mientras que la búsqueda en anchura explora todos los vecinos en un nivel antes de pasar al siguiente.

- Hash es una estructura de datos que almacena pares de clave-valor. La clave es única y funciona como entrada para la funcion hash. El resultado de la funcion es un entero que indica la posicion de la tabla donde se almacenara el valor. En general su orden es O(1) dependiendo de su implementacion .

- **7. Al ordenar una lista de números enteros aplicando el algoritmo quicksort, si como pivote se elige el primer elemento de la lista, ¿qué pasaría si se selecciona otro pivote?**
  - Primer o último pivote: Simples pero pueden llevar al peor caso de O($n^2$) en listas ya ordenadas.
  - Pivote aleatorio: Rendimiento promedio, minimiza la probabilidad del peor caso.
  - Pivote central: Generalmente proporciona particiones más equilibradas, mejorando el rendimiento promedio y reduciendo la probabilidad del peor caso.
- **8. Resolver el siguiente problema escolar. Dadas las notas de los alumnos de un colegio en el primer curso de bachillerato, en las diferentes asignaturas (5 por comodidad) se desea:**
  a. Calcular la media de cada alumno.
  b. Calcular la media de cada asignatura.
  c. Calcular la media total de la clase.
  d. Ordenar el listado de los alumnos por orden decreciente de notas medias individuales, que incluyen todas las materias.

  **Nota:** Utilizar dos algoritmos de ordenación diferentes para resolver el problema, justificando la elección. Generar los casos de prueba antes de hacer el ejercicio. **Nota:** Los casos de pruebas, estan presente en el repositorio. **El link ubicado al final del trabajo**.

  Clase Alumno:

```java
package ejemplos_InOut.Ejercicios.Repaso_Algoritmia;

public class Alumno {
    private int nombre;
    private float[] notas;

    public Alumno(int nombre, float[] notas) {
        this.nombre = nombre;
        this.notas = notas;
    }

    public int getNombre() {
        return nombre;
    }

```

```
    public float[] getNotas() {
        return notas;
    }
}
```

```
package ejemplos_InOut.Ejercicios.Repaso_Algoritmia;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.Scanner;

public class Ej_8 {
    static final int CANTALUMNOS = 6;
    static final int CANT_MATERIAS = 5;
    //static final String NOMBRE_ARCHIVO_0=
    ↪  "src/ejemplos_InOut/Ejercicios/Repaso_algoritmia/notas.txt";
    static final String NOMBRE_ARCHIVO_0 =
    ↪  "src/ejemplos_InOut/Ejercicios/Repaso_algoritmia/nota_2.txt";
    //static final String NOMBRE_ARCHIVO_0 =
    ↪  "src/ejemplos_InOut/Ejercicios/Repaso_algoritmia/nota_3.txt";
    static final Alumno[] alumnos = new Alumno[CANTALUMNOS];

    private static void medioAlumno() {
        for (int i = 0; i < alumnos.length; i++) {
            float[] notas = alumnos[i].getNotas();
            float promedio = calculoPromedio(notas);
            System.out.println("El promedio del alumno " + alumnos[i].getNombre() + " es: "
            ↪  + promedio);
        }
    }

        private static float calculoPromedio(float[] notas){
                float retornar=0;
                for (int i = 0; i < notas.length; i++) {
                        retornar+=notas[i];
                }
                retornar=retornar/CANT_MATERIAS;
                return retornar;
        }

        private static void insertionSort(){
                for (int i = 1; i < alumnos.length; i++) {
                        //auxiliar   alumno
                        Alumno aux = alumnos[i];
                        int j = i - 1;

                        while (j >= 0 && calculoPromedio(alumnos[j].getNotas()) <
                        ↪  calculoPromedio(aux.getNotas())) {
                                alumnos[j + 1] = alumnos[j];
                                j = j - 1;
                        }
                        alumnos[j + 1] = aux;
                }
        }
```

```
47
48      private static void bubbleSort() {
49          int n = alumnos.length;
50          for (int i = 0; i < n - 1; i++) {
51              for (int j = 0; j < n - i - 1; j++) {
52                  if (calculoPromedio(alumnos[j].getNotas()) < calculoPromedio(alumnos[j +
                    ↪  1].getNotas())) {
53                      Alumno temp = alumnos[j];
54                      alumnos[j] = alumnos[j + 1];
55                      alumnos[j + 1] = temp;
56                  }
57              }
58          }
59      }
60
61      private static void mediaAsignaturas() {
62          for (int j = 0; j < CANT_MATERIAS; j++) {
63              float sumaNotas = 0;
64              for (int i = 0; i < alumnos.length; i++) {
65                  sumaNotas += alumnos[i].getNotas()[j];
66              }
67              float promedio = sumaNotas / CANTALUMNOS;
68              System.out.println("La media de la asignatura " + (j + 1) + " es: " + promedio);
69          }
70      }
71          private static void mediaTotalClase() {
72          float sumaTotalNotas = 0;
73          int totalNotas = CANTALUMNOS * CANT_MATERIAS;
74
75          for (int i = 0; i < alumnos.length; i++) {
76              float[] notas = alumnos[i].getNotas();
77              for (int j = 0; j < notas.length; j++) {
78                  sumaTotalNotas += notas[j];
79              }
80          }
81
82          float promedioTotal = sumaTotalNotas / totalNotas;
83          System.out.println("La media total de la clase es: " + promedioTotal);
84      }
85
86      private static void leerArchivo_1() {
87          try {
88              BufferedReader buff = new BufferedReader(new FileReader(NOMBRE_ARCHIVO_0));
89              Scanner s = new Scanner(buff);
90              for (int i = 0; i < CANTALUMNOS; i++) {
91                  float[] nota = new float[CANT_MATERIAS];
92                  for (int j = 0; j < CANT_MATERIAS; j++) {
93                      nota[j] = s.nextFloat();
94                  }
95                  alumnos[i] = new Alumno(i + 1, nota);
96              }
97              buff.close();
98          } catch (FileNotFoundException ex) {
99              System.err.println(ex.getMessage() + "\nSignifica que el archivo del "
100                     + "que queriamos leer no existe.");
101         } catch (IOException ex) {
102             System.err.println("Error leyendo o escribiendo en algun archivo.");
103         }
```

```
    }



    public static void main(String[] args) {
        leerArchivo_1();
        medioAlumno();
                mediaAsignaturas();
                mediaTotalClase();
                //insertionSort();
                //bubbleSort();
                for (int i = 0; i < alumnos.length; i++) {
                        System.out.print("Id Alumno"+alumnos[i].getNombre()+"-        ");
                }
    }
}
```

- **9. Se leen dos listas de números enteros, $A$ y $B$ de 100 y 60 elementos, respectivamente. Se desea resolver mediante procedimientos las siguientes tareas:**

  a. Ordenar aplicando un método de ordenación distinto a cada una de las listas $A$ y $B$.

  b. Crear una lista $C$ a partir de la mezcla de las listas $A$ y $B$ ya ordenadas.

  c. Mostrar la lista $C$.

  **Nota**: En este ejercicio entendimos que el metodo de ordenamiento debe ser distintos a los utilizados en el ejercicio N°8.

```
        package ejemplos_InOut.Ejercicios.Repaso_Algoritmia;


import ejemplos_InOut.Aleatorio;//importo un archivo, con los metodos que utilizaremos

public class Ej_9 {
    static final int TAMANIO_A = 100;
    static final int TAMANIO_B = 60;

    private static void mostrarArray(int[] a){
        for (int i = 0; i < a.length; i++) {
            System.out.print(a[i]+" ");
        }
        System.out.println();
    }
    public static void main(String[] args) {
        int[] listaA = new int[TAMANIO_A];
        int[] listaB = new int[TAMANIO_B];
        for (int i = 0; i < TAMANIO_A; i++) {
            listaA[i] = Aleatorio.intAleatorio(1, 1000);
        }
        for (int i = 0; i < TAMANIO_B; i++) {
            listaB[i] = Aleatorio.intAleatorio(1, 1000);
        }

        mergeSort(listaA, 0, listaA.length - 1);
        mostrarArray(listaA);
        System.out.println();
```

```
        mergeSort(listaB, 0, listaB.length - 1);
        mostrarArray(listaB);
        System.out.println();
        //creo la lista C a partir de la mezcla de listaA y listaB
        int[] listaC = fusionLista(listaA, listaB);
        System.out.println("Lista C (mezcla de A y B ordenadas):");
        mostrarArray(listaC);
        System.out.println();


    }


  private static void mergeSort(int[] array, int izq, int der) {
    if (izq < der) {
        int mid = (izq + der) / 2;
        mergeSort(array, izq, mid);
        mergeSort(array, mid + 1, der);
        merge(array, izq, mid, der);
    }
}

//meetodo para fusionar dos mitades ordenadas
private static void merge(int[] array, int izq, int mid, int der) {
    int n1 = mid - izq + 1;
    int n2 = der - mid;
    //creo arrays tempo para las dos mitades
    int[] arrayIzq = new int[n1];
    int[] arrayDer = new int[n2];
    //copio datos a  los arrays temporales
    for (int i = 0; i < n1; i++) {
        arrayIzq[i] = array[izq + i];
    }
    for (int j = 0; j < n2; j++) {
        arrayDer[j] = array[mid + 1 + j];
    }
    // fusiono los arrays temp de vuelta al array originall
    int i = 0, j = 0;
    int k = izq;
    while (i < n1 && j < n2) {
        if (arrayIzq[i] <= arrayDer[j]) {
            array[k++] = arrayIzq[i++];
        } else {
            array[k++] = arrayDer[j++];
        }
    }

    // Copiar los elementos restantes de arrayIzq, si hay alguno
    while (i < n1) {
        array[k++] = arrayIzq[i++];
    }

    // Copiar los elementos restantes de arrayDer, si hay alguno
    while (j < n2) {
        array[k++] = arrayDer[j++];
    }
}

```

```java
// metodo para fusionar dos listas ordenadas
private static int[] fusionLista(int[] listaA, int[] listaB) {
    int[] listaC = new int[listaA.length + listaB.length];
    int i = 0, j = 0, k = 0;

    while (i < listaA.length && j < listaB.length) {
        if (listaA[i] <= listaB[j]) {
            listaC[k++] = listaA[i++];
        } else {
            listaC[k++] = listaB[j++];
        }
    }

    // Copio los elementos restantes de listaA
    while (i < listaA.length) {
        listaC[k++] = listaA[i++];
    }

    // Copio los elementos restantes de listaB
    while (j < listaB.length) {
        listaC[k++] = listaB[j++];
    }

    return listaC;
}

}
```

# 3 Repositorio en GitHub

El código fuente del proyecto está disponible en el siguiente repositorio de GitHub:
`https://github.com/Cristopher-Ovaillos/Analisis_Algoritmos.git`

# 4 Instrucciones

Para clonar el repositorio, usa el siguiente comando en tu terminal:

```
git clone https://github.com/Cristopher-Ovaillos/Analisis_Algoritmos.git
```