
Universidad Diego Portales

Escuela de Informática & Telecomunicaciones


ESTRUCTURAS DE DATOS & ANÁLISIS DE ALGORITMOS

# Laboratorio 1: Introducción a Java y algoritmos sobre arreglos


*Autores:*
*Cinthya Fuentealba Bravo*
*Ignacia Reyes Ojeda*

*Profesor:*
*Cristian Llull Torres*


27 de Agosto de 2025

# Índice

# 1. Introducción

Durante el proceso de estudio, existen diferentes tipos de evaluaciones, y mantener un formato ordenado y claro es primordial para los docentes, ya que esto facilita el ingreso de los resultados de las evaluaciones realizadas, calculo de promedios, entre otros cálculos que se necesitan en el rubro académico.

En este primer laboratorio, la Escuela de Informática y Telecomunicaciones de la Universidad Diego Portales solicita un código informático para procesar una gran cantidad de calificaciones y de esta forma manejar esta gran cantidad de información de manera eficiente. La escuela podría pagar por un software especializado en el requerimiento planteado, pero se prefiere ser proactivo optar por vender el producto creado en este laboratorio para que sea utilizado.

En la escuela existen una gran cantidad de estudiantes y también una gran variedad de evaluaciones, es por ese motivo que se opta por crear un sistema que pueda calcular estadísticas complejas de manera optimizada. El sistema planteado es la creación de una matriz que pueda manejar las evaluaciones, y también el uso de arreglos para un mejor funcionamiento.

Una matriz esta formada por filas y columnas, en este problema las filas representan a un estudiante y las columnas representaran una evaluación.

Este informe requiere del uso de un programa informático creado en IntelliJ IDEA, este se caracteriza por ser un editor, compilador y depurador. El código informático realizado esta realizado en lenguaje Java.

Por ende, dentro del informe se detallan la realización del código que busca dar solución al problema propuesto, también sobre los componentes de este código y su funcionamiento. En el repositorio de GitHub se encuentra el código Java y el archivo LaTeX para su análisis.

https://github.com/Cinthya982012/Laboratorio$_{1E}DA$

# 2. Implementación

Para comenzar a dar solución al problema planteado, y teniendo en cuenta las consideraciones que se deben tener en la simulación, se inicia la creación del código informático, para ello lo primero realizado en este laboratorio es la creación de la siguiente clase:

## 2.1. Clase Analizador de Notas

Esta clase representa el análisis de los diferentes cálculos que se utiliza por los docentes que se compone por los siguientes atributos:

- `double [][]notas`: corresponde a una matriz bidimensional, la cual permite el almacenamiento de las calificaciones, en donde [i][j], representan a lois estudiantes y a las evaluaciones, respectivamente.
- `String []evaluaciones`: arreglo que contiene el nombre de las evaluaciones y el tamaño de este debe coincidir con la cantidad de notas.
- `int []rut`:arreglo con los RUT de los estudiantes, cada uno de ellos debe ser único.
- `int CantEstudiantes`: es el número total de estudiantes del curso(cantidad de filas).
- `int CantEvaluaciones`: es el número total de evaluaciones del curso (cantidad de columnas).

A continuación se puede observar los atributos implementados:

```
1  //CLASE ANALIZADOR DE NOTAS
2  class AnalizadorDeNotas {
3
4      //=============================================
5      //ATRIBUTOS CLASE ANALIZADOR DE NOTAS
6      private double[][] notas; //Matriz bidimensional de las
           calificaciones[i][j](i Estudiantes /j Evaluaciones)
7      private String[] evaluaciones; //Arreglo con los nombres de
           las evaluaciones
8      private int[] rut; //Arreglo con los RUT de los estudiantes
9      private int cantEstudiantes; //Numero total de estudiantes
           (filas)
10     private int cantEvaluaciones;//Numero total de evaluaciones
           (columnas)
11
12     //=============================================
```

Esta clase esta compuesta por dos constructores principalmente, estos son:

- `public AnalizadorDeNotas(int cantEstudiantes, int cantEvaluaciones)`: corresponde a un constructor básico que recibe el tamaño de la matriz, es de-

cir, recibe la cntidad de estudiantes y evaluaciones que definen el tamaño de la matriz y los inicializa con datos aleatorios.

- `public AnalizadorDeNotas(int cantEstudiantes, int cantEvaluaciones, String [] evaluaciones)`: corresponde a un constructor avanzado, ya que además de lo anterior incluso recibe un arreglo con los nombre de las evaluaciones para asociarlos de manera explícita.

```
1  //CLASE ANALIZADOR DE NOTAS
2  class AnalizadorDeNotas {
3      //================================================
4      //CONSTRUCTOR 1
5      public AnalizadorDeNotas(int cantEstudiantes, int
           cantEvaluaciones) {
6          this.cantEstudiantes = cantEstudiantes;
7          this.cantEvaluaciones = cantEvaluaciones;
8
9          this.notas = new
               double[cantEstudiantes][cantEvaluaciones]; //Creaci n
               de la matriz (tama o estudiantes X evaluaciones)
10         this.rut = new int[cantEstudiantes]; // Creaci n de
               arreglo con rut
11         this.evaluaciones = new String[cantEstudiantes];
               //Creaci n de arreglo con los nombres de las
               evaluaciones
12
13         /**Genera RUT unicos y simples*/
14         for (int i = 0; i < cantEstudiantes; i++) {
15             rut[i] = 1000000000 + i;
16         }
17
18         /**Genera nombres de las evaluaciones*/
19         for (int j = 0; j < cantEvaluaciones; j++) {
20             this.evaluaciones[j] = "Evaluaci n" + (j + 1);
21         }
22
23         /**Rellena de manera aleatoria la matriz con notas entre 1
               y 7*/
24         java.util.Random rand = new java.util.Random();
25         for (int i = 0; i < cantEstudiantes; i++) {
26             for (int j = 0; j < cantEvaluaciones; j++) {
27                 this.notas[i][j] = 1.0 + rand.nextDouble() * 6.0;
28             }
29         }
30     }
31
32     //================================================
```

```
//=================================================
    //CONSTRUCTOR 2
    public AnalizadorDeNotas(int cantEstudiantes, int
        cantEvaluaciones, String[] evaluaciones) {
        this.cantEstudiantes = cantEstudiantes;
        this.cantEvaluaciones = cantEvaluaciones;

        /**Se copian los nombres de las evaluaciones*/
        this.evaluaciones = new String[cantEvaluaciones];
        for (int j = 0; j < cantEvaluaciones; j++) {
            this.evaluaciones[j] = evaluaciones[j];
        }

        /**Se copian los RUT de los estudiantes*/
        this.rut = new int[cantEstudiantes];
        for (int i = 0; i < cantEstudiantes; i++) {
            this.rut[i] = rut[i];
        }

        /**Se copia la matiz de las calificaciones*/
        this.notas = new double[cantEstudiantes][cantEvaluaciones];
        for (int i = 0; i < cantEstudiantes; i++) {
            for (int j = 0; j < cantEvaluaciones; j++) {
                this.notas[i][j] = notas[i][j];
            }
        }
    }
    //=============================================================================
```

También se realizan los *setters* y *getters* correspondientes de los atributos, los cuales permiten acceder y modificar elementos fuera de la clase. Esto se puede visualizar en la siguiente representación:

```
 //=============================================
    //SETTERS Y GETTERS
    public double[][] getNotas() {return notas;}
    public String[] getEvaluaciones() { return evaluaciones;}
    public int[] getRut() {return rut;}
    public int getCantEstudiantes() {return cantEstudiantes;}
    public int getCantEvaluaciones() {return cantEvaluaciones;}


    public void setNotas(double[][] notas) {this.notas = notas;}
    public void setEvaluaciones(String[] evaluaciones)
        {this.evaluaciones = evaluaciones;}
    public void setRut(int[] rut) {this.rut = rut;}
    public void setCantEstudiantes(int cantEstudiantes)
        {this.cantEstudiantes = cantEstudiantes;}
    public void setCantEvaluaciones(int cantEvaluaciones) {
        this.cantEvaluaciones = cantEvaluaciones;}

//=============================================
```

Además de los *atributos*, *constructores*, *setters* y *getters* creados, esta clase posee métodos, estos son los siguientes:

- `public double calcularPromedioEstudiante(int rut)`: realiza el calculo del promedio de un estudiante en especifico, según el RUT que posee.

```
//=================================================
//METODO QUE CALCULA EL PROMEDIO DEL ESTUDIANTE I
     public double calcularPromedioEstudiante(int rut) {
        /**Buscar el rut que corresponda al rut que se esta
           buscando*/
        int index = -1;
        for (int i = 0; i < cantEstudiantes; i++) {
            if (this.rut[i] == rut) {
                index = i;
                break;
            }
        }
        /**Si no se encuentra el RUT buscado se devuelve -1*/
        if (index == -1) {
            System.out.println("El estudiante con RUT:" + rut
               + " no existe.");
            return -1;
        }

        /**Sumar todas las notas del estudiante buscado por su
           RUT*/
        double suma = 0;
        for (int j = 0; j < cantEvaluaciones; j++) {
            suma += this.notas[index][j];
        }
        /**Se calcula el promedio del estudiante buscado por
           su RUT y se retorna dicho valor*/
        return suma / cantEvaluaciones;
   }
//=================================================
```

- `public double calcularPromedioEvaluacion(int index)`: realiza el calculo del promedio de una evaluación especifica.

```
//=================================================
//METODO QUE CALCULA EL PROMEDIO DE LA EVALUACION BUSCADA
   SEGUN SU NOMBRE
    public double calcularPromedioEvaluacion(int index) {
        /**Validar la evaluacion*/
        if (index < 0 || index >= cantEstudiantes) {
            System.out.println("La evaluacion:" + index + " es
               un indice invalido.");
            return -1;
        }

        /**Se recorren todos los estudiantes que rindieron la
           evaluacion*/
```

```
        double suma = 0;
        for (int i = 0; i < cantEstudiantes; i++) {
            suma += this.notas[i][index];
        }

        /**Se calcula el promedio de la evualuacion buscada y
           se retorna dicho valor*/
        return suma / cantEstudiantes;
    }
//=============================================================
```

- `public double calcularVarianzaEvaluacion(int index)`: realiza el calculo de la varianza de una evaluación especifica.

```
//=============================================================
//METODO QUE CALCULA LA VARIANZA DE UNA EVALUACION
    public double calcularVarianzaEvaluacion(int index) {
        /**Validar la evaluacion*/
        if (index < 0 || index >= cantEstudiantes) {
            System.out.println("La evaluacion:" + index + " es
               un indice invalido.");
            return -1;
        }

        /**Corresponde al promedio de la evaluacion*/
        double promedio = calcularPromedioEvaluacion(index);

        /**Suma de las notas del estuante menos el promedio,
           elevado a 2 (parte de a ecuacion de varianza)*/
        double suma = 0;
        for (int i = 0; i < cantEstudiantes; i++) {
            double resta = notas[i][index] - promedio;
            suma += resta * resta;
        }

        /**Se divide por la cantidad total de estudiantes y se
           retorna dicho valor*/
        return suma / cantEstudiantes;
    }
//=============================================================
```

- `public double [] calcularPromediosEstudiantes()`: realiza el calculo del promedio de las calificaciones de cada estudiante.

```
//=============================================
//METODO QUE CALCULA EL PROMEDIO DE NOTAS DE CADA ESTUDIANTE
    public double[] calcularPromediosEstudiantes() {
        double[] promedios = new double[cantEstudiantes];
        for (int i = 0; i < cantEstudiantes; i++) {
            double suma = 0;
            for (int j = 0; j < cantEvaluaciones; j++) {
                suma += this.notas[i][j];
            }
```

```
            promedios[i] = suma / cantEvaluaciones;
        }
        return promedios;
    }
```

- `public double [] calcularVarianzaEstudiantes()`: realiza el calculo de la varianza de las notas de cada estudiante.

```
//=================================================
//METODO QUE CALCULA LA VARIANZA DE NOTAS DE CADA ESTUDIANTE
    public double[] calcularVarianzaEstudiantes() {
        /**Arreglo de varianza de notas, uno por estudiante*/
        double[] varianzas = new double[cantEstudiantes];

        /**Para cada uno de los estudiantes*/
        for (int i = 0; i < cantEstudiantes; i++) {
            double suma = 0;

            /**Se suman todas las notas del estudiante i*/
            for (int j = 0; j < cantEvaluaciones; j++) {
                suma += this.notas[i][j];
            }

            /**Se calcula el promedio del estudiante i*/
            double promedio = suma / cantEvaluaciones;

            /**Suma de las diferencias del las notas del
               estudiante menos el promedio de la evaluaci n,
               elevado a 2 (parte de a ecuaci n de varianza)*/
            double sumadeRestas = 0;
            for (int j = 0; j < cantEvaluaciones; j++) {
                double resta = notas[i][j] - promedio;
                sumadeRestas += resta * resta;
            }

            /**Calculo de la varianza de las notas del
               estudiante i (suma de las diferencias dividido
               en la cantidad de estudiantes)*/
            varianzas[i] = sumadeRestas / cantEstudiantes;
        }

        /**Se retorna el valor de las varianzas*/
        return varianzas;
    }
//=================================================
```

- `public double [] calcularPromedioEvaluaciones(String [] evaluaciones)`: realiza el calulo de los promedios de evaluaciones especificas, para cada estudiante. Debe retornar un arreglo del tamaño de los estudiantes.

```
//=================================================
//METODO QUE CALCULA EL PROMEDIO DE LAS EVALUACIONES
   ESPECIFICAS PARA CADA ESTUDIANTE
```

```
3       public double[] calcularPromediosEvaluaciones(String[]
            evaluaciones) {
4           /**Arreglo de promedios de notas, uno por estudiante*/
5           double[] promedios = new double[cantEstudiantes];
6
7           /**Encontrar el nombre de las evaluaciones que se
                necesitan*/
8           int[] nombreEvaluaciones = new
                int[evaluaciones.length];
9           for (int i = 0; i < cantEstudiantes; i++) {
10              for (int j = 0; j < cantEvaluaciones; j++) {
11                  for (int k = 0; k < evaluaciones.length; k++) {
12                      nombreEvaluaciones[k] = -1; // Inicializar
                            en -1
13
14                      if
                            (this.evaluaciones[j].equals(nombreEvaluaciones[k]))
                            {
15                          nombreEvaluaciones[k] = j;
16                          break;
17                      }
18                      if (nombreEvaluaciones[k] == -1) {
19                          System.out.println("Evaluaci n " +
                                evaluaciones[k] + " no existe.");
20                      }
21                      /**C*/
22
23                      double suma = 0;
24                      int contador = 0;
25                      for (int indice : nombreEvaluaciones) {
26                          if (indice != -1) { //Si, solo si la
                                evaluacion es escontrada
27                              suma += this.notas[i][indice];
28                              contador++;
29                          }
30                      }
31                      promedios[i] = (contador > 0) ? suma /
                            contador : 0.0;
32                  }
33
34              }
35          }
36          return promedios;
37      }
38  //==================================================
```

- `public String encontrarMaximo(int index)`: corresponde a un método que retorna la nota máxima, para la evaluación elegida y retorna el RUT de dicho estudiante.

```
1   //==================================================
2   //Metodo para encontrar la calificacion maxima para la
```

```
    evaluacion seleccionada
     public String encontrarMaximos(int index) {
         /**Validar el index*/
         if (index < 0 || index >= cantEstudiantes) {
             System.out.println("Evaluaci n" +index + "no
                encontrada");
             return null;
         }
//================================================
```

## 2.2. Inicialización de la matriz de calificaciones

Lo primero realizado en el laboratorio es generar la creación de una matriz bidimensional (`double[][]notas`). Esta se encarga de representar la nota del estudiante $i$ en la evaluación $j$.

Se usa la clase *Random* perteneciente a Java, la cual permite inicializar las calificaciones con valores entre 0 y 1.

Se realiza la creación de un arreglo `int[] rut` el cual se encarga de almacenar los RUT de cada uno de los estudiantes.

Se crea un arreglo `String[]evaluaciones` el cual su función es guardar los nombre de cada una de las evaluaciones.

## 2.3. Estrategias de almacenamiento

La estructura base de este laboratorio es la creación de una matriz que esta formada por filas ($i$) y columnas ($j$). Esta guarda los valores de estudiantes y evaluaciones, respectivamente.Luego, crea la matriz de notas, el arreglo de ruts y el de evaluaciones, después, geenra ruts ficticios empezando desde 1000000000. A continuación, genera y enumera los nombres de evaluaciones, por último, llena la matriz de notas aleatorias entre 1.0 y 7.0.

## 2.4. Estrategias de Almacenamiento

1. Se utilizó un arreglo bidimensional para representar la matriz de notas, ya que permite acceso directo por (i, j).
2. Los nombres de evaluaciones y rut se almacenan en arreglos de tipo String[ ] e int[ ]

# 3. Experimentación y resultados

## 3.1. Dificultades

Existió dificultad en el manejo correcto de índices entre `cantEstudiantes` y `cantEvaluaciones`, así como también confusiones o simplemente distraciones.

La implementación de la clase `AnalizadorDeNotas`, no entregó los resultados esperados principalmente por por el diseño y forma de la utilización de los diferentes métodos usados en `main`.

El mayor problema reflejado en la implementación es la creación incorrecta de generar RUT para cada uno de los estudiantes. Como el RUT es invalido, permite que el método `calcularPromedioEstudiante` tome el RUT como un parámetro erróneo y retorna -1, y mensajes de que el RUT no existe.

Otro de los problemas presentados es en el `Contructor 2` ya que debe recibir un arreglo lleno de parámetros y lo que hace es terminar copiando un arreglo vació lleno de ceros.

Deben existir otros errores y detalles que corregir en el código informático, pero los mencionados anteriormente fueron los más destacables.

# 4. Conclusión

El laboratorio realizado no obtuvo los resultados esperados, ya que existen errores en el código informático, los cuales permiten una buena ejecución de el pero malos resultados mostrados por pantalla.

En conclusión los cálculos descritos como promedios y varianzas, están correctamente planteados pero el sistema no funciona de una manera correcta.

Uno de los principales errores es la creación errónea de los RUT de los estudiantes, en el `Constructor 2`lo cual lleva a invalidar el RUT ingresado y marcarlo como inexistente y por lo tanto a resultados inconsistentes.

Para trabajos futuros se debe de realizar ajustes mas puntuales en el `Main`, validar correctamente los parámetros, corregir errores que no permitan resultados esperados, mejorar mensajes de salida, entre otros. Con ello en próximos trabajos se debería obtener resultados coherentes y permitir la resolución correcta del problema planteado.