PRACTICO N° 8

En todas las soluciones considere la importancia de diseñar algoritmos correctos, eficientes y legibles seleccionando estructuras de control adecuadas. En todas las implementaciones puede agregar métodos para favorecer la modularización.

EJERCICIO 1. Un profesor desea automatizar el mantenimiento de la planilla de notas de sus alumnos.
Durante el trimestre el profesor considera tres notas para cada alumno, la primera corresponde a una prueba escrita, la segunda a la exposición oral y la tercera a los trabajos prácticos entregados. Todas las notas están dentro del rango 0 a 10. A partir de la planilla el profesor desea saber:

a. El promedio de cada alumno.
b. La nota máxima por evaluación, escrita, oral y tp.
c. El promedio general, que se obtiene de promediar el promedio de todos los alumnos.
d. El promedio general por evaluación, escrita, oral y tp.
e. El promedio de un alumno en particular.

El profesor ha codificado la nómina de alumnos de modo que a cada uno de ellos le asignó un número comenzando con 1.

NotasAlumnos
<<atributos de instancia>> examen: entero [][]
<<constructor>> NotasAlumnos(cantAlu : entero) <<comandos>> establecerNotaEscrita(p, n : entero) establecerNotaOral(p, n : entero) establecerNotaTP(p, n : entero) <<consultas>> cantAlumnos(): entero obtenerNotaEscrita(p : entero) : entero obtenerNotaOral(p : entero) : entero obtenerNotaTP(p : entero) : entero promedioAlumno(p : entero) : real promedioEscritas( ) : real promedioOrales( ) : real promedioTP( ) : real promedioGeneral( ) : real maximaNotaEscrita( ) : entero maximaNotaOral( ) : entero maximaNotaTP( ) : entero

- Implemente una solución para este problema considerando el diagrama de clase y agregando todos los métodos que considere adecuados.
- Defina una clase tester para la clase NotasAlumnos, que lea las notas de un archivo secuencial y luego verifique los servicios provistos por la clase.
- Agregue a la clase NotasAlumnos una consulta ***frecuencia()*** que retorne objeto de la clase *NumerosEnteros* (del práctico 7) de 11 números enteros. Cada componente i del arreglo indica cuántos alumnos obtuvieron la nota i (0=< i <=10) en la exposición oral.

EJERCICIO 2. Implemente la clase GrillaEnteros modelada en el siguiente diagrama

GrillaEnteros
<<atributos de instancia>> S : entero[][]
<<constructores>> GrillaEnteros (n,m : entero) <<comandos>> intercambiar(f1,f2:entero) reemplazar(v1,v2:entero) reemplazarPrimera (v1,v2 : entero) <<Consultas>> cantFilas():entero cantColumnas():entero obtenerFilas():entero obtenerColumnas():entero obtener(f,c:entero):entero contarFilas(v : entero):entero

El constructor ***GrillaEnteros (n,m : entero)*** crea un arreglo de *n* por *m* componentes y lo inicializa con números generados al azar.
intercambiar(f1,f2:entero) intercambia los elementos de las filas *f1* y *f2*.
reemplazar(v1,v2:entero) reemplazar todas las apariciones del elemento *v1* por el elemento *v2*.
reemplazarPrimera (v1,v2 : entero) reemplaza en cada fila la primera aparición del elemento *v1* por el elemento *v2*.
contarFilas(v : entero):entero Cuenta en cuántas filas aparece el elemento *v*.

a. Implemente la clase GrillaEnteros
b. Establezca casos de prueba adecuados para verificar cada servicio.
c. Escriba una clase tester que permita verificar los servicios con los casos de prueba almacenados en un archivo secuencial. La clase debe incluir un método interno mostrarGrilla()

Ejercicio 3. Analice la legibilidad, eficiencia y correctitud de la siguiente implementación del método tieneUnaRaya. El método debe retornar verdadero si y solo si, la matriz m, cuyos valores han sido establecidos previamente, contiene una fila i, 0<=i<nf, tal que $m_{i,j} = m_{i,j-1}$ para 1<=j<nc

```
class Matriz {
      //Atributos de instacia
      int [][] m;
      int nf,nc;

      //Constructor
      public Matriz (int f,int c){
         m = new int [f][c];
         nf = f ; nc = c;
      }

      //Comando
      public void establecer( int e,int f,int c){
         //Requiere 0<=f<nf y 0<=c<nc
         m[f][c] = e;
      }
```

```
      //Consultas
      public boolean tieneUnaRaya (){
         //Retorna true si en una fila de m todos los elementos son iguales entre sí
         int i = 0; int j = 1; boolean tieneRaya = false;
         while (i <nf && !tieneRaya){
               if (m[i][j-1] != m[i][j]) {
                  i ++; j=1;}
               else
                 j++;
                 if (j == nc)  tieneRaya = true;
            }
         return tieneRaya;
      }
}
```

Ejercicio 4. Considere el siguiente diagrama e implemente la clase

Texto
tex [] [] char
Texto (Arch:String, fil,col:entero)
comprimir ()

Un objeto de tipo Texto permite representar una secuencia de líneas de texto. Cada línea contiene palabras separadas por uno o más espacios y signos. Las palabras no se cortan de una línea a otra. Todas las líneas tienen la misma cantidad de caracteres. El constructor inicializa el texto con los valores almacenados en el archivo Arch. Un ejemplo de texto sería:

T	**I**	**P**	**O**		**D**	**E**				**D**	**A**	**T**	**O**								
U	**N**		**T**	**I**	**P**	**O**		**D**	**E**		**D**	**A**	**T**	**O**		**E**	**S**		**U**	**N**	**A**
C	**O**	**L**	**E**	**C**	**C**	**I**	**O**	**N**		**D**	**E**										
O	**P**	**E**	**R**	**A**	**C**	**I**	**O**	**N**	**E**	**S**		**D**	**E**	**F**	**I**	**N**	**I**	**D**	**A**	**S**	
P	**A**	**R**	**A**		**U**	**N**		**C**	**O**	**N**	**J**	**U**	**N**	**T**	**O**						
													D	**E**							
			D	**A**	**T**	**O**	**S**	**.**		**S**	**I**										
				L	**A**		**R**	**E**	**P**	**R**	**E**	**S**	**E**	**N**	**T**	**A**	**C**	**I**	**O**	**N**	
E	**S**	**T**	**A**				**E**	**N**	**C**	**A**	**P**	**S**	**U**	**L**	**A**	**D**	**A**				
E	**L**			**T**	**I**	**P**	**O**		**E**	**S**		**A**	**B**	**S**	**T**	**R**	**A**	**C**	**T**	**O**	**.**

El método ***comprimir()*** : cuando es posible, reemplaza las secuencias de uno o más espacios en blancos por uno solo.
Las primeras cuatro líneas del ejemplo anterior quedarían:

T	**I**	**P**	**O**		**D**	**E**		**D**	**A**	**T**	**O**		**U**	**N**		**T**	**I**	**P**	**O**		**D**	**E**
D	**A**	**T**	**O**		**E**	**S**		**U**	**N**	**A**		**C**	**O**	**L**	**E**	**C**	**C**	**I**	**O**	**N**		
D	**E**		**O**	**P**	**E**	**R**	**A**	**C**	**I**	**O**	**N**	**E**	**S**									
D	**E**	**F**	**I**	**N**	**I**	**D**	**A**	**S**		**P**	**A**	**R**	**A**		**U**	**N**						

Analice la correctitud, legibilidad y eficiencia de su solución.