Trabajo Práctico 5 - Estructuras Estáticas

Ejercicio 1.

Creá la clase **Anio**, que tendrá un arreglo de Strings que contenga los nombres de cada uno de los doce meses del año, y otro que contenga la cantidad de días de cada uno (ignorá los años bisiestos) con los siguientes métodos:

- `public String getNombreDelMes(int numeroMes)`
  Recibe el número de mes (entre 1 y 12) y devuelve el nombre del mes en cuestión.
- `public int diasTranscurridos(int numeroMes)`
  Recibe el número de mes y devuelve la cantidad de días transcurridos en el año antes de comenzar el mes en cuestión.

En el programa principal mostrá cuántos días transcurrieron antes del comienzo del año y qué día del año es el día de cumpleaños de cada integrante del grupo.

Para discutir en clase o a través de los foros del ***Aula Virtual***: Si el método *diasTranscurridos(..)* es usado una y otra vez, ¿hay alguna manera que evitar que el cálculo de los días transcurridos se haga permanentemente? Si la hay, modificá la clase para mejorar su *performance*.

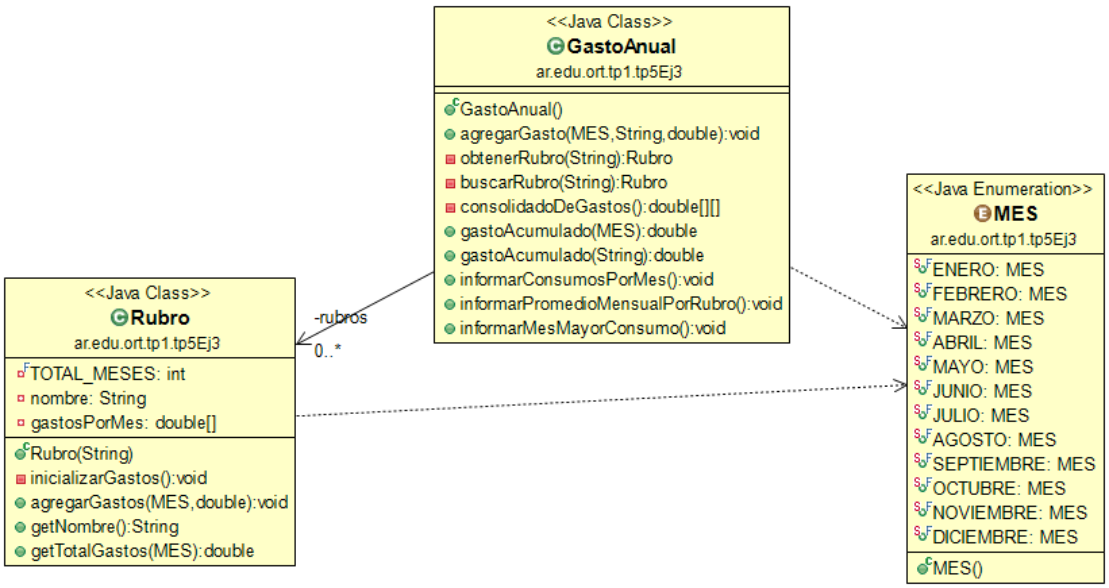
Ejercicio 2.

¿Cambiaría algo si en vez de usar un *Array de Strings* para los nombres de los meses usáramos un enumerado?

Creá una nueva versión de la clase Anio llamada **AnioV2** que use el ***enum*** **Mes** (definido dentro de la clase pero público) para enumerar los doce meses del año. Luego modificá lo que corresponda para que todo siga funcionando.

Ejercicio 3.

La clase **GastosAnuales** permite guardar los datos de los gastos comunes del año separados por rubro y mes. Para eso tiene una colección de rubros, donde cada **Rubro** tiene un nombre y guarda el importe de los gastos de cada uno de los meses del año. El diseño de clases (incompleto) es el que se muestra en el siguiente gráfico:



Implementá los métodos de **Rubro**:

- **`public Rubro(String nombre)`**
  Es el constructor. Recibe el nombre o descripción del rubro. Debe inicializar el arreglo de importes.
- **`private void inicializarGastos()`**
  Inicializa el arreglo de importes.
- **`public void agregarGasto(Mes mes, double importe)`**
  Acumula el importe en la posición correspondiente al mes indicado.
- **`public getNombre()`**
  Devuelve el nombre del Rubro.
- **`public double getTotalGastos(Mes mes)`**
  Devuelve el importe acumulado de gastos para el mes indicado.

Implementá los métodos de **GastoAnual**:

- **`public GastoAnual()`**
  Es el constructor, e inicializa la colección de Rubros.
- **`public void agregarGasto(Mes mes, String nombreRubro, double importe)`**
  Agrega el importe gastado al rubro que corresponda y en el mes indicado. Si el rubro no se encuentra registrado en la colección se lo agregará, y cuando ya exista se acumulará en este el valor del gasto. Pero debe controlarse que el importe ingresado sea mayor que cero.
- **`private Rubro obtenerRubro(String nombreRubro)`**
  Obtiene y devuelve el Rubro a partir de su nombre. Cuando éste no exista deberá crearlo.
- **`private Rubro buscarRubro(String nombreRubro)`**
  Busca y devuelve un Rubro a partir de su nombre. Cuando no lo encuentre deberá volver null.

- **`private double[][] consolidadoDeGastos()`**
  Genera un arreglo bidimensional consolidando en una sola estructura todos los gastos del año. La matriz debe medir 12 (la cantidad de meses del año) por la cantidad de Rubros existentes, y cada celda debe contener el importe acumulado para el rubro en ese mes.
- **`public double gastoAcumulado(Mes mes)`**
  Devuelve el importe del gasto acumulado en el mes indicado.
- **`public double gastoAcumulado(String nombreRubro)`**
  Devuelve el importe del gasto acumulado en el rubro indicado. Si el rubro no existe deberá devolver -1.
- **`public void informarConsumosPorMes()`**
  Muestra los consumos por mes (discriminado por cada rubro de gasto y acumulado).
- **`public void informarPromedioMensualPorRubro()`**
  Muestra los consumos promedio por mes en cada rubro.
- **`public void informarMesMayorConsumo()`**
  Calcula y muestra nombre e importe acumulado del mes con mayor consumo total (puede ser uno o más de uno).

Ejercicio 4.

Descargá el proyecto **TP1-TP4.zip** del Aula Virtual, el cual se encuentra incompleto. Importalo en Eclipse y completá las clases y el programa para cumplir con la siguiente consigna:

> **ACLARACIÓN**: tratar de reutilizar la mayor cantidad de código posible (es decir, no copiar y pegar en cada método de la clase **CircuitoATP**).

Se desea llevar la estadística del circuito anual de tenis. En el mismo participan 5 jugadores del país: "Pella", "Del Potro", "Schwartzman", "Mayer" y "Delbonis".

El circuito consta de 5 torneos, a saber: "Australia", "USOpen", "RolandGarros", "Wimbledon", "Shangai".

Existe una matriz de valores enteros de 5x5 que contiene en qué puesto finalizó cada tenista en cada torneo, donde cada fila es un tenista y cada columna un torneo. Por ejemplo:

		Torneos				
		Australia	**USOpen**	**RolandGarros**	**Wimbledon**	**Shangai**
Jugadores	**Pella**	1	3	4	1	3
	Del Potro	3	2	3	4	1
	Schwartzman	2	1	5	5	2
	Mayer	4	5	1	2	5
	Delbonis	5	4	2	3	4

Deberás completar los siguientes métodos de la clase **CircuitoATP**:

- **`public ArrayList<Jugador> procesarInfo()`**
  Tal que procese la matriz, cree las instancias de jugadores, procese los resultados de cada uno y retorne la lista de jugadores.
- **`private int buscoJugador(String jugador)`**
  el cual recibe el nombre de un jugador y devuelve la posición del mismo dentro del array de jugadores. En caso de que no exista, devolver -1.
- **`private int buscoTorneo(String torneo)`**
  tal que reciba el nombre de un torneo y devuelva la posición correspondiente del mismo en el array de torneos. En caso de que no exista, devolver -1.
- **`public String procesarTorneosJugador(String Jugador)`**;
  tal que retorne, para el jugador enviado por parámetro, el resultado de todos los torneos en un string con el nombre del torneo y su puesto en el mismo. Por ejemplo:

```
Del Potro: Australia:3 USOpen:2 RolandGarros:3 Wimbledon:4 Shangai:1
```

- **`public String obtenerResultadoJugador(String jugador, String torneo)`**
  tal que devuelva el puesto en que finalizó un jugador (enviado por parámetro) en un torneo (enviado por parámetro) y lo devuelva en un string. Ejemplo:

```
Resultado de Schwartzman en RolandGarros:5
```

- **`public int procesarPeorPosTorneoJugador(String jugador)`**
  tal que devuelva la peor posición en un torneo del jugador enviado por parámetro. Ejemplo:

```
Peor Resultado de Pella en el año: 4
```

En el main debe:

1. Crear una instancia de la clase **CircuitoATP**
2. Invocar el método ***procesarInfo()***
3. Imprimir el resultado final del campeonato, recorriendo la lista obtenida en el punto 2.
4. Imprimir los resultados de Delbonis, invocando al método correspondiente de la clase **CircuitoATP**
5. Imprimir el resultado de Schwartzman en RolandGarros, invocando al método correspondiente de la clase **CircuitoATP**
6. Imprimir el peor resultado de Pella en todo el año.

```
La salida debe ser:
Resultado final campeonato
Jugador=Pella, puntos=125, mejor_resultado=1, peor_resultado=4]
Jugador=Del Potro, puntos=100, mejor_resultado=1, peor_resultado=4]
Jugador=Schwartzman, puntos=100, mejor_resultado=1, peor_resultado=5]
Jugador=Mayer, puntos=80, mejor_resultado=1, peor_resultado=5]
Jugador=Delbonis, puntos=45, mejor_resultado=2, peor_resultado=5]
Delbonis: Australia:5 USOpen:4 RolandGarros:2 Wimbledon:3 Shangai:4
Resultado de Schwartzman en RolandGarros:5
Peor Resultado de Pella en el año: 4
```