Colecciones y Estructuras Estáticas

Guía de Ejercicios

Ejercicio 1.

Implementá la clase ***Grupo***, la cual representa al grupo de Trabajos Prácticos y sus integrantes. Para eso, la clase tendrá un atributo *nombre* (String) para guardar el nombre del grupo y otro atributo llamado *integrantes* (colección de Strings), donde cada elemento será el nombre de un integrante de tu grupo de trabajos prácticos. Debe cumplir con el siguiente diseño (explotar/implementar los métodos que se mencionan a continuación):

- *public Grupo(String nombre)*
  Es el constructor. Debe inicializar correctamente los atributos de la clase.
- *public String getNombre()*
  Devuelve el nombre del grupo.
- *private void setNombre(String nombre)*
  Setea el nombre del grupo a partir del parámetro recibido.
- *public int getCantidadIntegrantes()*
  Devuelve la cantidad actual de integrantes del grupo.
- *public void agregarIntegrante(String nombreIntegrante)*
  Agrega al grupo el nombre de un nuevo integrante, siempre y cuando éste no haya sido cargado con anterioridad. Cuando ya exista, no debe volver a agregarlo.
- *private int obtenerPosicionIntegrante(String nombreIntegrante)*
  Devuelve la posición en el grupo del integrante cuyo nombre se ha recibido por parámetro. En caso de no encontrarlo devolverá -1.
- *public String obtenerIntegrante(int posicion)*
  Recibe un valor entero que representa la posición del integrante en la lista (entero igual o mayor que 1) y devuelve el integrante guardado en (*posicion* - 1) dentro de la colección de integrantes. Si el integrante no existe se debe devolver *null*.
- *public String buscarIntegrante(String nombre)*
  Busca al integrante del grupo cuyo nombre coincida con el parámetro de entrada. En caso de que haya coincidencia, lo devuelve; caso contrario, el método devolverá *null*.
- *public String removerIntegrante(String nombreIntegrante)*
  Remueve de la colección de integrantes a aquel cuyo nombre coincide con el nombre recibido como parámetro. Si lo encontró, luego de removerlo debe devolverlo con *return*, y si no debe devolver *null*.
- *private void mostrarIntegrantes()*
  Muestra por consola la cantidad de integrantes y los nombres de cada uno de ellos.
- *public void mostrar()*
  Muestra por consola el nombre del grupo, la cantidad de integrantes y el nombre de cada uno de ellos.
- *public void vaciar()*
  Elimina todos los elementos de la lista de integrantes.

Desde el *main()*:

- Creá una instancia de Grupo asignándole el número, la letra o el nombre que le hayan puesto.

- Agregale los miembros reales de tu grupo de trabajos prácticos. Agregale uno más, ficticio. Cada vez que agregues un integrante el programa debe mostrar el mensaje "[nombreIntegrante] fue asignado al grupo [nombreGrupo]", donde [nombreIntegrante] y [nombreGrupo] son, respectivamente, los nombres del nuevo integrante y del grupo.
- Escribí un método del programa principal que compruebe si un integrante determinado fue o no agregado al grupo. Probalo tanto con uno que exista como con uno que no.
- Mostrá todos los datos del grupo.
- Remové al integrante ficticio para que el grupo quede con los integrantes reales. Luego de removerlo de la lista, mostrá los datos del integrante removido.
- Volvé a remover el mismo integrante ficticio.
- Mostrá de nuevo los datos del grupo, actualizados.
- Vaciá el grupo y volvé a mostrarlo.

Ejercicio 2.

Para los Juegos Olímpicos de la Juventud nos pidieron un programa con el cual podamos calcular quién fue el ganador de una carrera. Para eso ingresaremos un atleta y los segundos que ha empleado (con decimales) para recorrer la distancia de una especialidad determinada. debe tener en cuenta que puede haber más de un ganador (sus tiempos empleados fueron los mismos).

Ejercicio 3.

Modificá el ejercicio anterior para que en vez de indicar solamente quién fue el ganador obtenga cuál fue la terna ganadora. Debe tener en cuenta que puede haber atletas que "empaten" en cualquiera de las tres posiciones del podio.

Ejercicio 4.

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 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*.

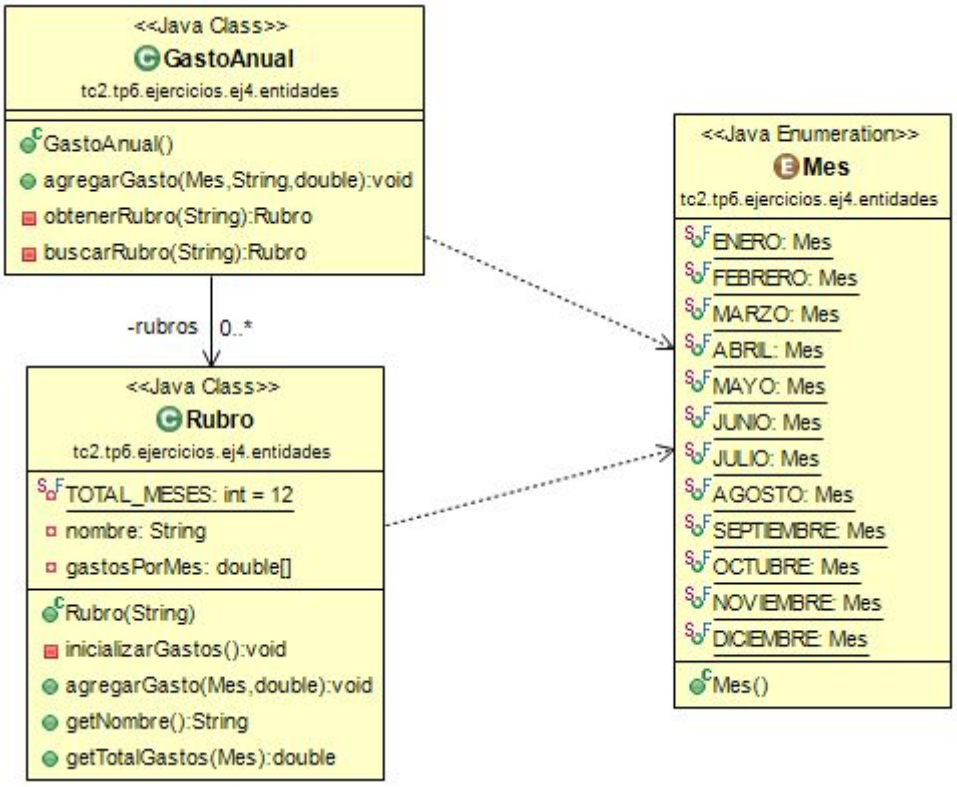
Ejercicio 5.

¿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*** (interno pero público) para enumerar a los doce meses del año, y modificá lo que creas conveniente.

Ejercicio 6.

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).