C.F.G.S. D.A.W. - 20-02-2023 - CONTROL 2º PARCIAL - PROGRAMACIÓN

Se pretende realizar un programa en Java para gestionar los proyectos que se usarán en el módulo de Programación de 1º DAW. Para ello, se han diseñado las siguientes clases, en las que se muestran sólo sus atributos:

`public class Proyecto`
`{`
` private String nombre;`
` private Clase tClases[];`
`}`
	`public class Clase`
	`{`
	` private String nombre;`
	` private Definicion tAtribs[];`
	` private Metodo tMetodos[];`
	`}`
		`public class Definicion`
		`{`
		` private String tipo;`
		` private String nombre;`
		`}`

`public class Metodo`
`{`
` private String tipoDevuelto;`
` private String nombre;`
` private Definicion tParametros[];`
` private int complejidad;`
`}`
	`public class Gestion`
	`{`
	` public static Proyecto face;`
	` public static Proyecto guerras;`
	`}`	

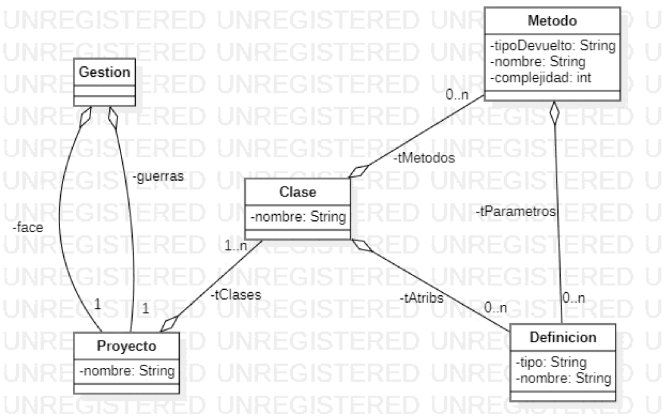
Es decir, un objeto de la clase **`Definicion`** representa lo que sabemos en Java: p.ej., **`int num`**

Un objeto de la clase **`Metodo`** representaría una función en Java, donde además, guardamos la complejidad ciclomática. p.ej: **`double calcMedia(float num1, float num2)`**, donde **`float num1`** y **`float num2`** serían dos objetos de la clase **`Definicion`**

Y un objeto de la clase **`Clase`** sería un nombre, varios atributos (objetos de la clase **`Definicion`**) y varias funciones o métodos (varios objetos de la clase **`Metodo`**)

Por último, un objeto de la clase **`Proyecto`** tendría un nombre, y un conjunto de objetos de la clase **`Clase`**

Por lo que el diagrama de clases UML es el siguiente:



Se pide realizar los siguientes métodos:

a) *(2 Puntos)* Realizar el método **`public boolean igualesParametros(Metodo otro)`** en la clase **`Metodo`** que recibe un objeto tipo **`Metodo`** como parámetro y devuelve un valor booleano indicando si el objeto al que se aplica tiene los mismos parámetros que el objeto de la clase **`Metodo`** pasado como parámetro (en orden, tipo y número). Atención: **`func1(int n, String m)`** tiene los mismos parámetros que **`func2(int numeroElementos, String nombre)`.**

b) *(1,5 Puntos)* Realizar el método **`public boolean equals(Object otro)`** en la clase **`Metodo`** que devuelve un **`boolean`** indicando si el objeto al que se aplica es igual que el objeto pasado como parámetro. El criterio es el de Java: dos métodos se consideran iguales si se llaman igual y tienen los mismos parámetros, en orden, tipo y número. Usar para este método el realizado en el punto anterior. No se aceptará el **`equals`** generado por Eclipse.

c) *(2,5 Puntos)* Una clase en Java no puede tener métodos duplicados. Realizar el método **`public boolean tieneMetodosDuplicados()`** en la clase **`Clase`** que devuelve un boolean indicando si el objeto al que se aplica es una clase errónea, por tener métodos iguales. Usar para ello los métodos hechos anteriormente. (en una clase es un error tener, por ejemplo los dos métodos **`void func(String a, float b)`** e **`int func(String nombreEmpleado, float valorMin)`**

d) *(1 Puntos)* Realizar en la clase **`Clase`** el método **`public boolean esMasComplejaQue(int complejidadMin)`** que devuelve un booleano indicando si el objeto al que se aplica tiene algún método con una complejidad ciclomática mayor que el valor **`complejidadMin`** pasado como parámetro.

e) *(3 Puntos)* Realizar el método **`public Metodo[] getMetodosQueUsan(String tipo)`** en la clase **`Clase`** que devuelva una tabla de objetos de la clase **`Metodo`** con aquellos métodos que tienen algún parámetro del tipo pasado como parámetro. El tamaño de la tabla debe estar ajustado al número de elementos que contenga.

Se deben realizar todas las funciones auxiliares (si son necesarias) que se usen. **Los métodos no deben pedir datos al usuario, ni mostrar ningún resultado. No se deben añadir atributos a las clases. Los resultados se muestran con las instrucciones que ya hay en `Gestion.java`, y que hacen uso de los métodos que se piden al alumno.**

Valoraciones:
- Las clases (los ficheros fuente .java a entregar) deben compilar sin errores.
- No se deben producir excepciones.
- Sólo se puede presuponer alguna condición si no contradice el enunciado.
- Se valorará el código correcto, indentado y comentado; la reutilización de código, la descomposición en funciones en los casos adecuados, la eficiencia y claridad de los algoritmos y la inexistencia de código o variables superfluas.

Ejemplo de salida de datos con los datos que se proporcionan en **`Gestion.java`**, que son los siguientes:

```
Proyecto: ProyectoGuerras
=========================
public class Gestion
  //ATRIBUTOS
  //METODOS
public class Guerra
  //ATRIBUTOS
  private String denominacion;
  private Bando bandoA;
  private Bando bandoB;
  private Batalla tablaBatallas[];
  private int nBatallas;
  //METODOS
  public  Guerra (String denominacion, Bando bandoA, Bando bandoB) - compejidad: 2;
  public void anadeBatalla (Batalla batalla) - compejidad: 3;
  public String toString () - compejidad: 1;
  public int getNPaises () - compejidad: 5;
  public int compareTo (Guerra otra) - compejidad: 4;
public class Bando
  //ATRIBUTOS
  //METODOS
public class Batalla
  //ATRIBUTOS
  //METODOS
public class Pais
  //ATRIBUTOS
  private String nombre;
  private Guerra tablaGuerras[];
  private int nGuerras;
  //METODOS
  public boolean haSidoAliadoDe (Pais otro) - compejidad: 2;
  public Pais[] posiblesEnemigos () - compejidad: 3;
  public int getNBatallas () - compejidad: 1;

Proyecto: ExamenFacebook
========================
public class GestionFacebook
  //ATRIBUTOS
  //METODOS
  public void main (String args[]) - compejidad: 2;
  public void hacerAmigos (Miembro a, Miembro b) - compejidad: 6;
  public boolean sonTodosAmigos (ListaMiembros lista) - compejidad: 4;
  public float índiceDeSimilitud (Miembro a, Miembro b) - compejidad: 3;

public class Miembro
  //ATRIBUTOS
  private String email;
  private String nombre;
  private ListaMiembros amigos;
  //METODOS
  public boolean tieneComoAmigoA (Miembro otro) - compejidad: 4;
  public ListaMiembros personasQueQuizaConozca () - compejidad: 4;
  public ListaMiembros amigosEnComun (Miembro b) - compejidad: 4;
public class ListaMiembros
  //ATRIBUTOS
  private Miembro tabla[];
  //METODOS
  public void añadeMiembroSinRepetir (Miembro m) - compejidad: 2;
  public boolean contieneMiembro (Miembro m) - compejidad: 6;
  public void eliminaMiembro (Miembro m) - compejidad: 4;
```

La salida debería ser:

```
Apartado a): (debe salir false):false
Apartado a): (debe salir true):true
Apartado b): (debe salir false):false
Apartado b): (debe salir true):true
Apartado c): (debe salir false):false
Apartado c): (debe salir true):true
Apartado d): (debe salir true):true
Apartado d): (debe salir false):false
Apartado e): [void main (String args[]) - compejidad: 2]
Apartado e): [ Guerra (String denominacion, Bando bandoA, Bando bandoB) - compejidad: 2, void Guerra
(String x, Bando y, Bando z) - compejidad: 1]
Apartado e): []
```