TRABAJO PRACTICO Nº 1 - PROGRAMACION 3

A- Codifique un programa de consola en Java que permita realizar las siguientes acciones:

Generar un número aleatorio entre 0 y 100, para ello use la siguiente función:

```
int x = new Double(Math.random() * 100).intValue();
```

Una vez generado el número codifique la lógica necesaria para encontrar el número aleatorio ayudando al usuario informando al mismo si el número ingresado es mayor o menor al número aleatorio buscado. Una vez encontrado el número muestre la cantidad de intentos necesarios para lograrlo.

Ejemplo:

Número aleatorio generado: 63
Ingrese un número entre 0 y 100.
Numero Ingresado: 50
Respuesta: Es muy bajo
Ingrese un número entre 0 y 100.
Numero Ingresado: 75
Respuesta: Es muy alto
Ingrese un número entre 0 y 100.
Numero Ingresado: 60
Respuesta: Es muy bajo
Ingrese un número entre 0 y 100.
Numero Ingresado: 65
Respuesta: Es muy alto
Ingrese un número entre 0 y 100.
Numero Ingresado: 63
Respuesta: Correcto, numero encontrado, cantidad de intentos **5**

A.2- Codifique un programa de consola o Swing en Java:

Programar un algoritmo recursivo que permita realizar la división por 2 de un número hasta que el mismo sea menor a 1. Muestre el resultante de cada recursión.

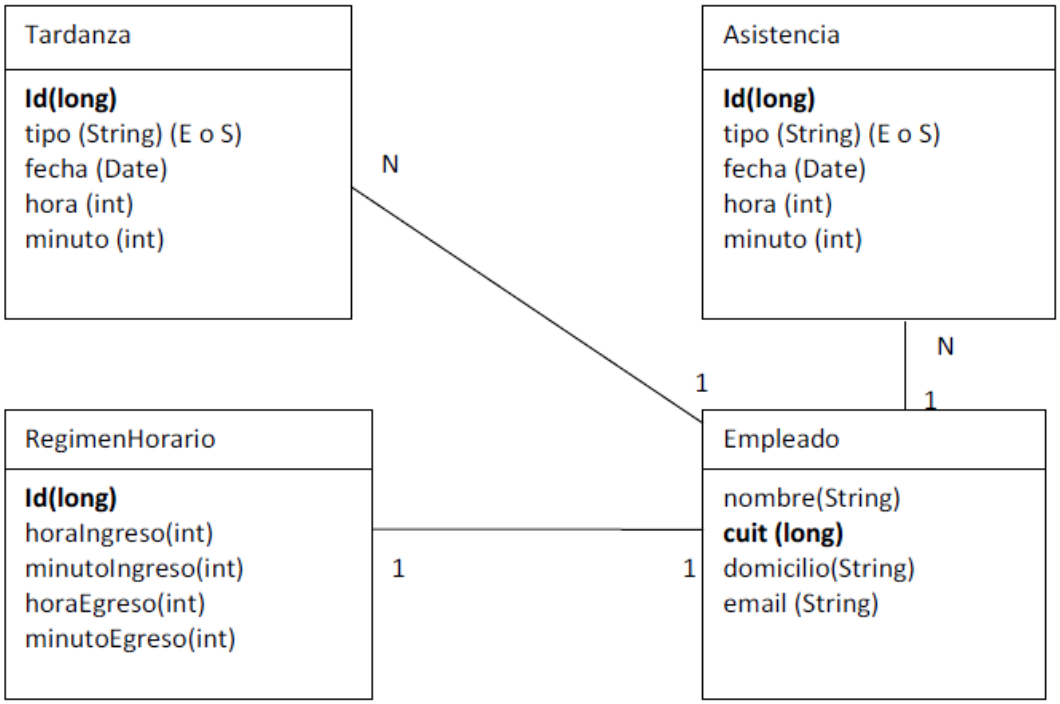
Ejemplo:
Entrada: *123*
1 recursión => *123* / 2 = 61,5
2 recursión => *61,5 / 2 = 30,75*
3 recursión => 30,75 / 2 = 15,375
4 recursión => *15,375 / 2 = 7,6875*
5 recursión => *7,6875 / 2 = 3,84375*
6 recursión => *3,84375 / 2 = 1,921875*
7 recursión => *1,921875 / 2 = 0,9609375*

Estructura del Metodo:

```
public void calculaMitadNumero (double numero){

}
```

B- Dado el siguiente modelo relacional:

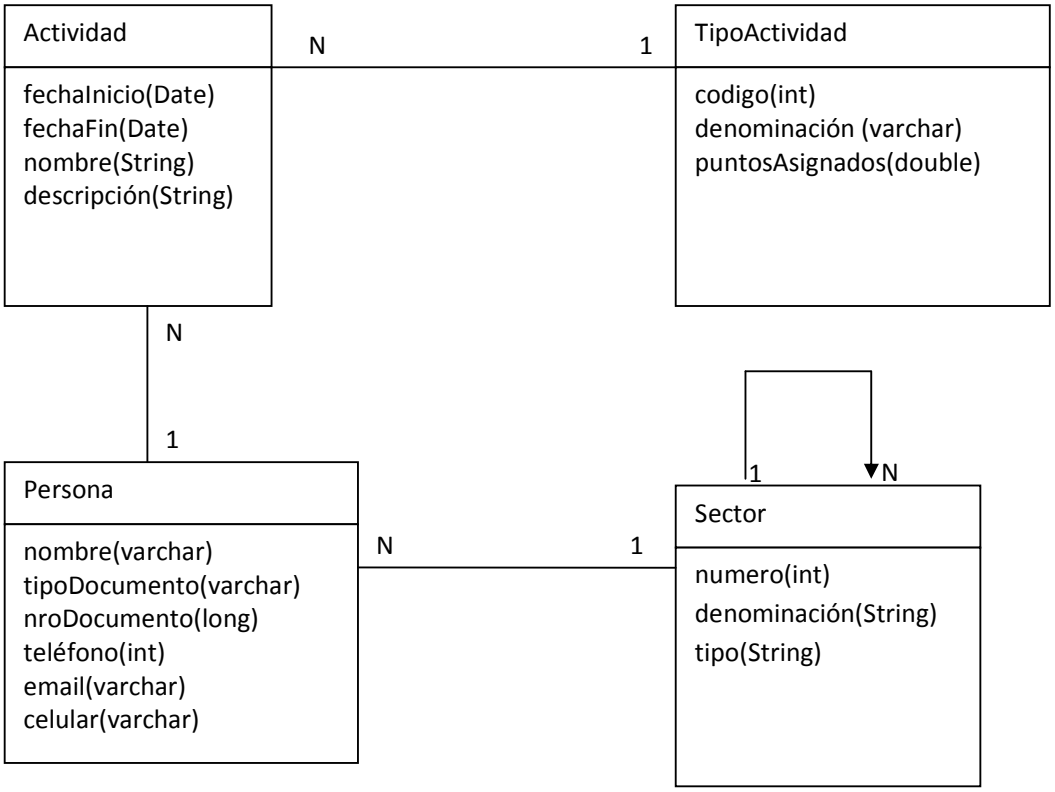


B.1- Represente el modelo de clases de entidad anterior mediante código **JAVA**, todas las asociaciones deben ser bidireccionales.

B.2- Codifique en la clase **Empleado** un método denominado **public List<Asistencia> getAsistenciaXMesXAnio(int mes, int anio){}** que retorne solo las asistencias del empleado correspondientes al mes y año indicados

B.3- Codifique en la clase **Empleado** un método llamado **public List<Tardanza> getDiasConTardanza(int mes, int anio){}** que retorne los días con tardanza para hacerlo verifique el horario de la asistencia contra el horario asignado en el régimen horario, si la asistencia supera en 15 minutos el horario establecido en el régimen horario, cree una instancia de Tardanza y copie la información de la Asistencia a la Tardanza, almacene la tardanza en un Array del mismo tipo, finalmente retorne el Array de tardanzas. Reutilice el método **getAsistenciaXMesXAnio** () del punto anterior.
Dado el siguiente modelo de Clases:

C- Dado el siguiente modelo relacional:



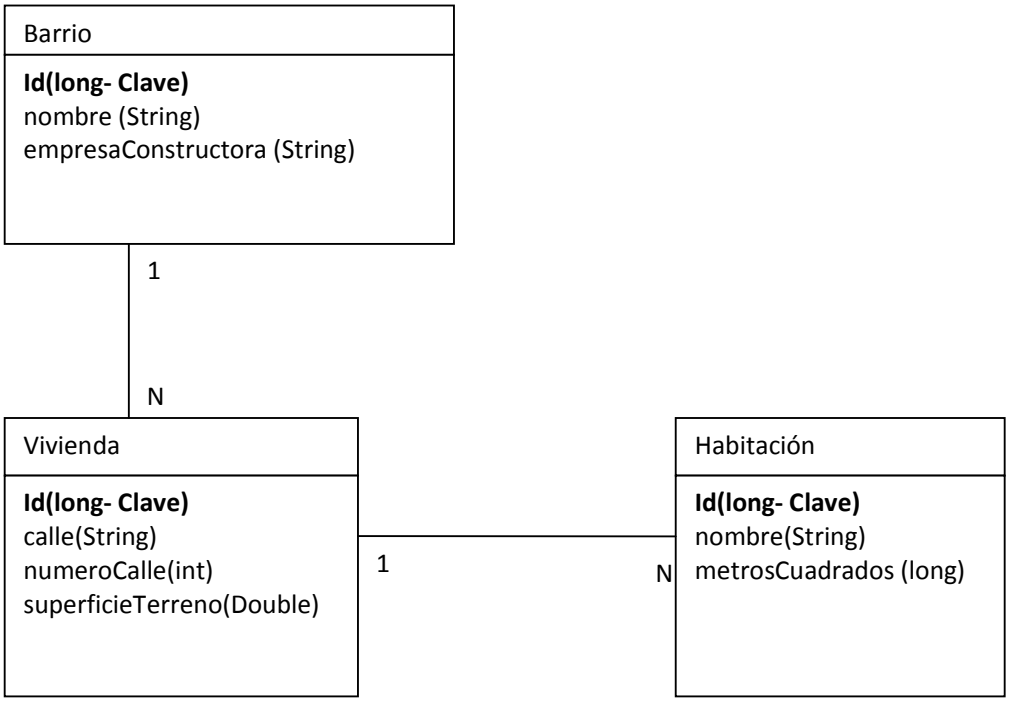
C.1- Codifique en la clase Persona un método denominado **"public double totalPuntosAsignados(long nroDocumento)"** que retorne el total de puntos asignados de las actividades realizadas por una persona.

C.2- Codifique en la clase Persona un método denominado "**public double totalPuntosAsignados (long nroDocumento, int código)"** que retorne el total de puntos asignados de un único tipo de actividad realizada por una persona.

C.3- Codifique en la clase Persona un método denominado "**public doublé totalPuntosAsignados (long nroDocumento, int código, int anio)"** que retorne el total de puntos asignados de un único tipo de actividad realizada por una persona para un periodo asignado.

C.4- Codifique en la clase Sector un método RECURSIVO denominado "**public List<Sector> obtenerTotalSubsectores()"** que retorne la totalidad de subsectores de un sector.

D- Dado el siguiente modelo relacional:



D.1- Represente el modelo de clases de entidad anterior mediante código **JAVA**, todas las asociaciones deben ser bidireccionales. Mapee mediante anotaciones de **JPA (10%)**

D.2- Codifique en la clase **Barrio** un método llamado **public double getSuperficieTotalTerreno(){}** que retorne el total de metros de terreno del barrio teniendo en cuenta la totalidad de viviendas asociadas al mismo. **(10%)**

D.3- Codifique en la clase **Vivienda** un método denominado **public double getMetrosCuadradosCubiertos(){}** que retorne el total de metros cuadrados de la vivienda teniendo en cuenta la cantidad de habitaciones asociadas. Al finalizar el cálculo valide que el valor obtenido no sea mayor que la superficie del terreno, si ocurre esa situación emita una excepción con el mensaje "La superficie cubierta no puede ser mayor a la superficie del terreno" **(15%)**

D.4- Codifique en la clase **Barrio** un método llamado **public double getSuperficieTotalCubierta(){}** que retorne los metros cuadrados cubiertos del barrio sumando la totalidad de metros cuadrados cubiertos de las viviendas que componen al barrio. Reutilice el método **getMetrosCuadradosCubiertos** () del punto anterior. **(10%)**