universidad del país vasco euskal herriko unibertsitatea
departamento de electricidad y electrónica
elektrika eta elektronika saila

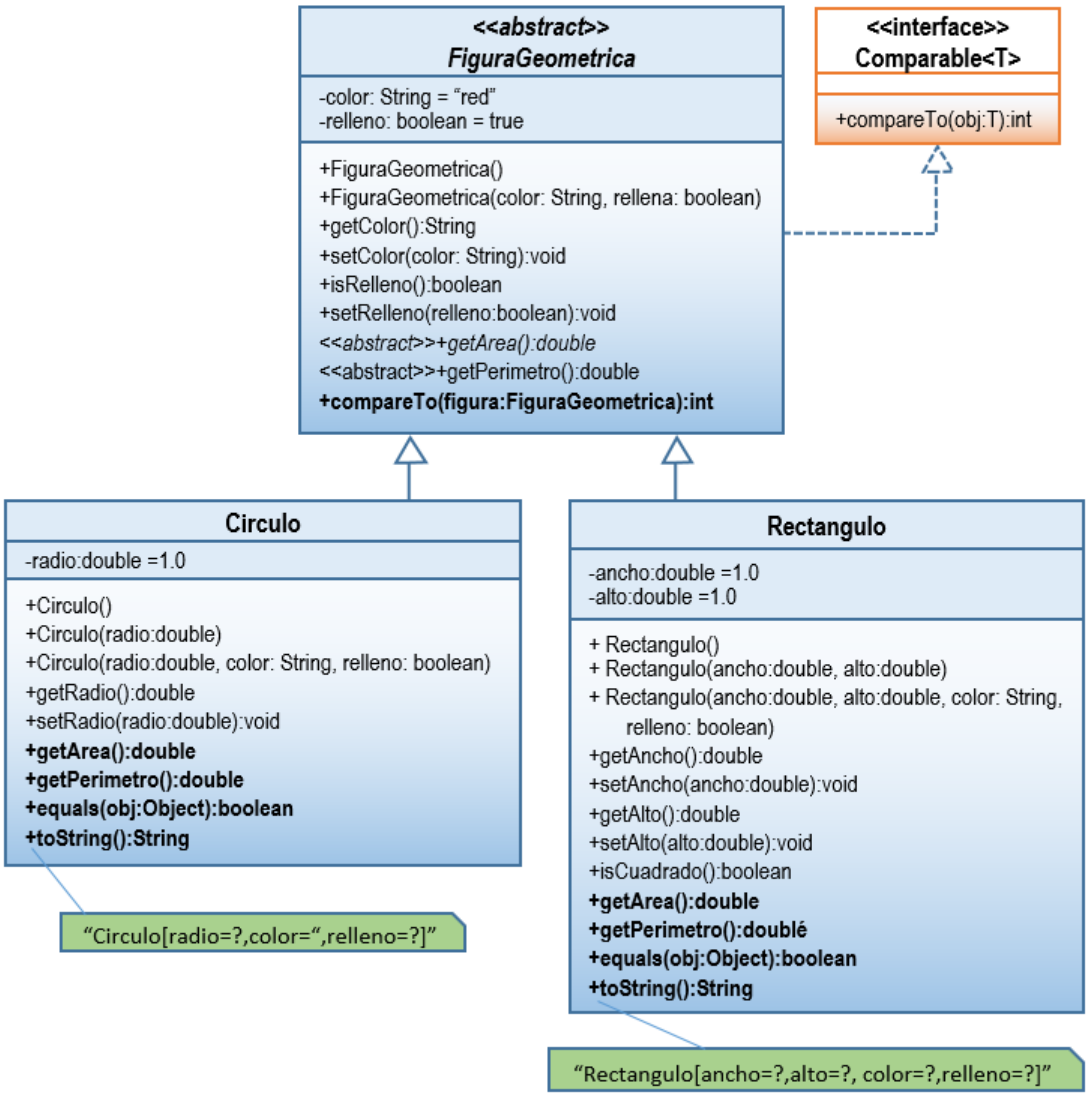
Técnicas Actuales de Programación 2020/2021

Examen Final (21.01.2021)

Responda al examen con un proyecto que lleve su nombre (si usa Netbeans elija Java with Ant -> Java Application). Cada ejercicio deberá resolverlo en un package diferente (al poder ser elija su espacio de nombres dentro de edu.upvehu.tap).

1- Jerarquía de clases (puntuación 1/3)

Escriba el código correspondiente a las clases de la figura (se explicará en el aula).
El interfaz ya existe en la biblioteca estándar. Se implementa en la clase FiguraGeométrica y por tanto se podrá comparar toda clase de figuras geométricas, lo que se hará **en base a su área**. (Estudiar en la documentación de "Comparable<T>" cómo ha de comportarse el método "compareTo(.)")
Incluya un mínimo de documentación para ser procesada por Javadoc.



Si una vez resueltos todos los ejercicios, tiene tiempo, compruebe que todo funciona correctamente escribiendo otra clase con un "main(.)" que ponga a prueba las características que merecen comprobación.

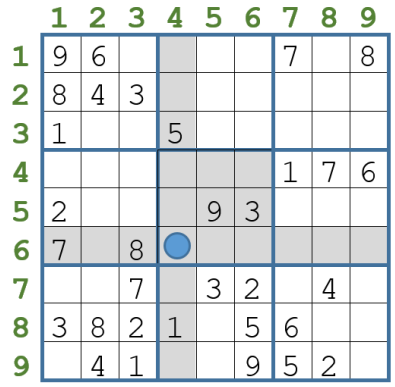
2- Resolución de un problema numérico (Métodos) (puntuación 1/3)

Escribir el código necesario para descubrir todos los números "autoexplicativos" entre 1 y 1E6.
Un número es autoexplicativo si está formado por 10 cifras o menos, y cada cifra en la posición "i" indica las veces que la cifra "i" está contenida en el número.

Ejemplo: 2020 es autoexplicativo
posición 0 = 2 y tiene dos ceros
Posición 1 = 0 y tiene cero unos
posición 2 = 2 y tiene dos doses
posición 3 = 0 y tiene cero treses

Estructure bien la solución.

3- Una clase (puntuación 1/3)



	1	2	3	4	5	6	7	8	9
1	9	6					7		8
2	8	4	3						
3	1			5					
4							1	7	6
5	2				9	3			
6	7		8						
7			7		3	2		4	
8	3	8	2	1		5	6		
9		4	1			9	5	2	

En el juego llamado **Sudoku**, se define una matriz de 9x9 celdas que debemos rellenar con números del rango [1-9] en base a una sencilla regla: Un número puede aparecer una única vez en cualquier fila, columna o sección de 3x3 de las 9 que se observan en la figura.

Como parte del desarrollo de una aplicación relacionada con este juego, deberá crear una clase **`Sudoku`** que cumpla con los siguientes requerimientos:

- Los objetos de la clase se inicializarán a partir del nombre de un fichero. Este fichero contendrá texto en el que cada línea válida constará de tres enteros en el rango [1-9]. Estos enteros indicarán respectivamente: número de fila, número de columna, y valor de la celda. Aquellas celdas que no sean inicializadas quedarán vacías.
- Han de existir dos métodos, **`get(i,j)`** y **`set(i,j,x)`**, que permitan consultar y establecer el valor de la celda i,j (sin exigir que el número respete las normas del juego).
- Un método **`size()`** devolverá el número de celdas conteniendo números (entre 0 y 81).
- Un método **`rectangleIndexes(i,j)`** devolverá una lista de arrays de dos enteros con los índices de las celdas que comparten rectángulo con la celda i,j. La lista devuelta no deberá contener los índices i,j.
- Un método **isValid(i,j,x)** servirá para comprobar si es posible colocar el valor x en la celda i,j (que es un número válido y que no incumple las normas del juego).
- Un método **`available(i,j)`** devolverá la lista de posibles valores para la celda (i,j) teniendo en cuenta las restricciones de su fila/columna/rectángulo.

Todo aquello que no se especifica en el enunciado queda a su criterio (representación del tablero, de las celdas vacías, mecanismo para controlar las situaciones de error,…). En cuanto a la herencia de Object, reescriba lo que considere necesario y ponga un comentario justificando por qué no considera necesario reescribir lo que no reescriba)

Si ve que le sobra bastante tiempo puede ponerse a definir el fichero con los datos de la figura e intentar escribir el código que resuelve el sudoku con ayuda de la clase que ha implementado