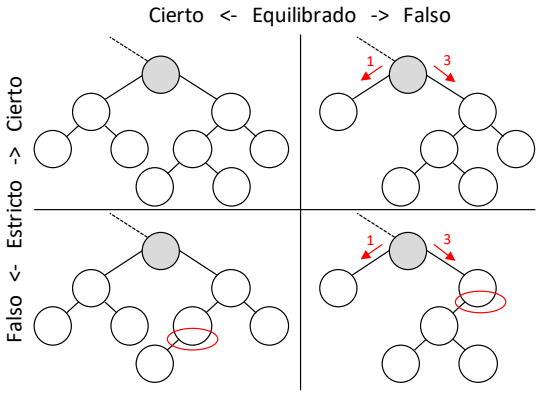
Técnicas Actuales de Programación 2019/2020
Examen Final

Responda al examen con un proyecto en Netbeans (use Java with Ant -> Java Application) que lleve su nombre. Cada ejercicio deberá resolverlo en un package diferente.

1- Métodos en base a un Interfaz (1/5)



Tenemos un interfaz que denominamos BinTreeNode, que será implementado por clases que permitan construir árboles binarios, y que especifica la existencia dos métodos para obtener los dos nodos hijos de uno dado (obviamente, si alguno de los hijos no existe, la correspondiente función devolverá null)…

```
public interface BinTreeNode {
    BinTreeNode rightChild();
    BinTreeNode leftChild();
}
```

Escriba una clase de utilidad **BinTreeUtil**, con los tres métodos estáticos siguientes:

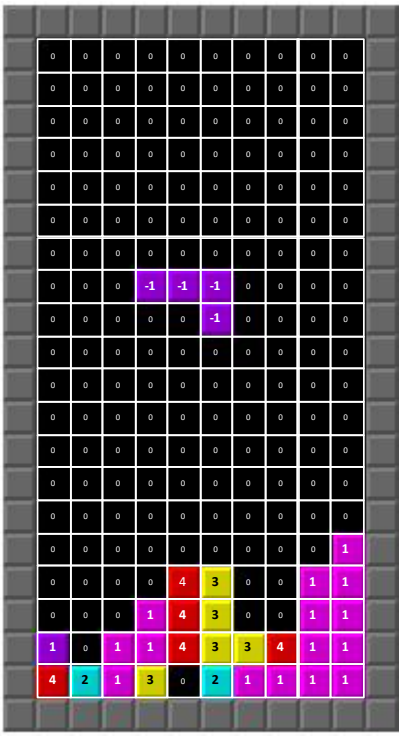
`public static int depth(BinTreeNode node) {…}`
que proporciona la profundidad del (sub)árbol que parte del nodo dado (incluido). Es 1 + la profundidad del árbol hijo más profundo (un nodo sin hijos tiene profundidad 1).

`public static boolean isStrict(BinTreeNode node){…}`
que indica si el (sub)árbol que parte del nodo dado (incluido) es "estricto", es decir, si ningún nodo tiene un solo hijo (o lo que es lo mismo: todos tienen dos o ningún hijo)

`public static boolean isBalanced(BinTreeNode node){…}`
que indica si el (sub)árbol que parte del nodo dado (incluido) es "balanceado", es decir, si para todo nodo la diferencia de "profundidad" de sus dos hijos no difiere en más de 1 (o lo que es lo mismo: es 0 o 1)

Estos métodos son extremadamente sencillos si tiene en cuenta que la respuesta para un nodo se construye en función de información obtenida para sus hijos (en ocasiones con llamadas recursivas, es decir, llamando a la misma función que se está definiendo, aplicada a un nodo hijo)

2- Clase para una App. (2/5)



Escriba una clase que represente al tablero de un juego de Tetris (un videojuego que se hizo "viral" a finales del pasado siglo). Si no conoce el juego atienda a las explicaciones previas al tiempo de examen, y en todo caso tiene una explicación al final del enunciado (puede probar el juego aquí: https://www.freetetris.org/game.php)

Esta clase (que será de tipo "model") estará acompañada (al menos) por otras dos que NO SE PIDE escribir:

- una de tipo "view" (un GUI) que mostrará una representación gráfica del juego.
- una clase de tipo "controller" será el motor del juego y llamará a los métodos que se le piden más abajo

Fijaremos la representación interna:

El tablero será un array bidimensional de enteros, donde el valor 0 representará una casilla vacía, y los valores 1..7 representarán la ocupación de la casilla por piezas de diferentes colores/topologías. Las casillas de la pieza "móvil" se representarán con su código de color, pero con valor negativo, de modo que podrán distinguirse de las demás y ser manejadas por el motor del juego en respuesta a las acciones del jugador y del paso del tiempo.

Constructor:

- Se inicializará como un tablero vacío con las dimensiones dadas como parámetros.

Métodos:

- **Un método** permitirá ocupar una casilla con un color como parte de una pieza que será considerada "móvil" (es decir, recibe fila, columna y color, y codifica el valor negativo del color en la casilla indicada)
- **Otro método** permitirá saber si no hay ninguna pieza móvil en el tablero.
- La pieza móvil podrá ir hacia abajo mediante **un método** siempre y cuando no haya hecho tope contra el fondo o las piezas que en él se encuentren. En caso de que no pueda moverse la pieza se dará por fijada y sus valores negativos pasarán a ser positivos.
- También podrá ir a derecha o izquierda si no está ya en el extremo mediante **dos métodos** al efecto.
- Para eliminar filas horizontales completas, **un método** devolverá el primer número de fila que se encuentre completa, comenzando por el fondo (-1 si no hay ninguna). Posteriormente **otro método** se encargará de eliminar la línea bajando todos los elementos fijos que se encuentren por encima de ella.
- (serían necesarias unas rutinas de giro de la pieza móvil que no se pide escribir)
- Como un jugador avezado puede durar horas sin ser derrotado, ha de ser posible guardar el estado del juego y recuperarlo en otra ocasión, por lo que necesitaremos **dos métodos** que lo vuelquen y recuperen en/de unos Streams (a elegir… como texto, como datos, como objetos,… pero en todo caso ambos métodos deben complementarse)

En qué consiste el juego:

Se trata de un espacio rectangular que podemos considerar un tablero con un determinado número de casillas en horizontal y otro en vertical. Dentro de este tablero, en un instante dado, hay un número de piezas geométricas que se muestran, cada una de ellas, como en un patrón concreto de casillas ocupadas (con diversos colores). Las piezas van apareciendo, una a una, en la parte superior del tablero, y caen hacia el fondo, donde quedan depositadas. De este modo, en todo instante tenemos una pieza cayendo, y un conjunto de piezas que ya se encuentran descansando en el fondo. El jugador debe mover a derecha o izquierda y rotar la pieza que cae (mediante el teclado) de modo que consiga el mejor encaje en el fondo, ya que toda fila completa en el mismo se volatiliza, compactándose el resto, y el objetivo del juego es durar el mayor tiempo posible sin que el tablero se llene de piezas. Esto se traduce en puntos simplemente por conteo de las filas eliminadas.

3- Arquitectura de clases (2/5)

Supongamos que queremos crear una aplicación como WhatsApp. Esta aplicación debe tener al menos las siguientes estructuras:

- **User** – Un participante (el propietario o un contacto) que puede enviar mensajes. Contendrá al menos la siguiente información:
  - Nombre: un identificador. No puede haber dos usuarios con el mismo nombre.
  - Estado: (en el trabajo, enojado, feliz ...)
  - número de teléfono
- **Group** – indica un conjunto de contactos. Contendrá al menos la siguiente información:
  - Nombre: un identificador. No puede haber dos grupos con el mismo nombre o el nombre de un participante.
  - Colección de contactos participantes (un subconjunto de ellos serán administradores, lo que significa que se les permitirá realizar cambios).
- **Chat** – Una conversación con mensajes que se da entre dos contactos o dentro de un grupo. Ambos tipos de conversación comparten características a su vez tienen diferencias que los distinguen, por lo que razonablemente podemos considerar la distinción UserChat y GroupChat. Contendrá al menos la siguiente información:
  - Colección secuencial de mensajes
- **Message** – texto enviado por un usuario (propietario o contacto). Contendrá al menos la siguiente información:
  - La identidad del remitente.
  - El momento de creación.
  - Contenido del mensaje

Cree las clases / interfaces que sirvan para representar las estructuras anteriores y cree una clase **WhatsApp**, con campos, constructores y métodos que estime necesarios (sólo declaraciones, no es preciso definir el contenido de los métodos).