ABSTRACTAS

Una clase sabemos que es abstracta ya que no creamos oobjetos de esa clase, no se puede o porque en el enunciado nos piden métodos que no se implementa ejemplo de método abstracto:

public abstract double calcularPuntuacion();

INTERFACES

Una **clase abstracta** puede heredar de una sola **clase (abstracta** o no) mientras que una **interfaz** puede extender varias **interfaces** de una misma vez. Una **clase abstracta** puede tener métodos que sean **abstractos** o que no lo sean, mientras que las **interfaces** sólo y exclusivamente pueden definir métodos **abstractos**.

En las interfaces solo hay exclusivamente métodos (por defecto) abstractos y atributos constantes(public static final).

Los métodos no hace falta definirlos abstractos, lo son predeterminadamente.

Al hacer una interfaz marcar como **interface** para que sea una interfaz. En las clases “hijas” usamos el ‘**implements**’ en vez del ‘**extends**’ lo que hacemos con los métodos los implementamos, osea hacemos un override del método abstracto con los datos de esa clase

TRATAMIENTO DE ERRORES Y EXCEPCIONES

Utilizaremos los **try{}** y **catch(exception e){}.**

Cuando pongamos un catch siempre pondremos las excepciones más específicos en los primeros catch y luego los más banales.

Dentro de los catch tiene que haber contenido(programación).

PROPAGACIÓN DE EXCEPCIONES

El main debe ser el encargado de tratar las excepciones.

**runTimeException** , si no tratamos este tipo error en un método auxiliar (por ejemplo) se propaga sin que pongamos un throws.

Los errores distintos a **runTimeException** al propagarlos debemos poner en la cabecera del metodo auxiliar un “throws (tipoDeExcepcion) {” y asi propagamos el tipo de excepcion,

COMO PASAR DE CADENA de TEXTO A NUMERO ENTERO

