String numero = Console.ReadLine();

Int j;

If(Int.tryParse(numero, out j){}

out=es una forma de pasar un puntero en la funcion

***Clase 2***

*Programacion Orientada a Objetos*:

-Propone resolver problemas de la resalidad a travez de identificar objetos y relaciones de colaboracion entre ellos

-El objeto y el mensaje son sus eelementos fundamentales

***Piilares***:

*Abstracción*: cuando programemos nosotros podriamos realizar una solucion sin importar de no saber como una funcion hace algo pero si que hace.

*Encapsulamiento*:poder separar las funciones del main o de donde se ejecute el codigo

Herencia: poder reutilizar codigo que contegan las mismas carateristicas(en objetos)

Poliformismo: tener mismos nombres pero reprensentan otros objetos

***Clases***:

Se las llama en main asi:

Ej: Galletita galletitaUno = new Galletita()

*Modificadores*:

Abstract: indica que la clase no podrá instanciarse.

Internal: Accesible en todo el proyecto(Assembly)

Public: Accesible desde cuaquier proyecto.

Private: Accesor por defecto

Sealed: Indica que la clase no podrá heredar

Modificadores atributos:

Private: los mienbros de la mism clase

Protected: los mienbros de la misma clase y clases derivadas o hijas

Internal protected: lso miembros del mismo proyecto o clases derivadas.

Public: Cualquier miembro. Accesibilidad abierta.

***Metodos***:

*Abstract*: Sólo la firma del método, sinimplementar.

*Extern*: Firma del método (para métodosexternos).

*internal* (\*): Accesible desde el mismo proyecto.

*Override*: Reemplaza la implementación delmismo método declarado como virtual en una clase padre.

*public* (\*): Accesible desde cualquier proyecto.

*private* (\*): Sólo accesible desde la clase.

*protected* (\*): Sólo accesible desde la clase oderivadas.

*Static*: Indica que es un método de clase.

*Virtual*: Permite definir métodos, con suimplementación, que podrán ser

sobrescritos en clases derivadas.

***Namespace***:

-Es una agrupación lógica de clases y otros elementos.

-Toda clase esta dentro de un NameSpace.

-Proporcionan un marco de trabajo jerárquico sobre el cuál se construye y organiza todo el código.

-Su función principal es la organización del código para reducir los conflictos entre nombres.

-Esto hace posible utilizar en un mismo programa componentes de distinta procedencia.

***Directivas***:

C# posee dos directivas de NameSpace:

-Using: Permite la especificación de una llamada a un método sin el uso obligatorio de un nombre completamente cualificado.

-Alias: ej: using SC = System.Console;

Anotaciones:

***static***: es una funcion que se puede usar sin tener creado el objeto de la clase

out: setea el dato

ref: pasa la referencia del dato

'nombre'.Next('max','min');