***Polimorfismo***: El método heredado desde una superclase por dos o más subclases y cada subclase implementa un comportamiento distinto para él.

***Sobreescritura:*** se utiliza para modificar la implementación abstracta o virtual de un método, propiedad, indizador o evento heredado. El modificador que se utiliza para lograrlo es **override.**

***Sobrecarga***: cuando tenemos varios métodos con igual nombre y distinta firma.

**TEORIA DE OBJETOS**

Acciones (Métodos) vs reacciones (Eventos): en la acción “le solicito” al objeto que haga algo, mientras que en la reacción es cuando algo externo produce un cambio en él.

**Características del Modelo O.O:**

* Abstracción: Mecanismo que define las características esenciales de un objeto que lo distingue de los otros, rescatamos los aspectos relevantes. Separar una porción de la realidad perteneciente al dominio de un problema. Métricas:
  + Acoplamiento: que tan interrelacionada está una abstracción respecto a otras. (débil)
  + Cohesión: grado de conectividad y consistencia de los elementos dentro de una abstracción. (alto)
  + Suficiencia: la abstracción captura **suficientes** características y comportamientos para una interacción significativa y eficaz.
  + Compleción: la abstracción captura **todas** las características y comportamientos para una interacción significativa y eficaz.
  + Ser primitivo: Refiere a las operaciones de la abstracción. Una operación es primitiva cuando su implementación es representada por la forma más simple posible, sin perder la naturaleza de lo que hace.
* Encapsulamiento: “Oculta” los detalles de implementación del objeto, estableciendo barreras para distinguir el alcance de la abstracción. (no confundir con “Esconder”). Respetar el encapsulamiento es respetar el contacto entre objetos. Separa la interfaz de una abstracción con la implementación.
  + Interfaz: Elemento constituyente de un objeto que representa su visión externa. Contraposición del encapsulamiento (visión interna vs visión externa).
* Modularidad: Propiedad de un sistema que ha sido descompuesto en un conjunto de módulos cohesivos y débilmente acoplados con el objetivo de poder administrarlos de forma más eficiente.
* Jerarquía: Clasificación u ordenación de abstracciones. Tipos:
  + **Herencia** (“es-un”): el elemento más generalizado posee una jerarquía superior al más especializado. Se establece por la naturaleza de las cosas. (Águila “es un “ave).
  + **Agregación** (“todo-parte”): el todo posee una jerarquía superior a las partes. (Rueda parte de Auto)
  + Debido a la **Jerarquía de tipos** (“es-un”) todo objeto se corresponderá con una clase y las clases debido a la **Jerarquía estructural** (“todo-parte”) estarán ordenadas jerárquicamente.
* Tipos: Son la puesta en vigor de la clase de los objetos, de modo que los objetos de tipos distintos no pueden intercambiarse o, como mucho, pueden intercambiarse solo de formas muy restringidas. Un objeto podrá adoptar distintos tipos dependiendo de las definiciones realizadas en la clase que le dio origen. Una clase está formada de varios tipos, pero un tipo está formado de una clase. Ligaduras:
  + Estática: se fijan los tipos de todas las variables y expresiones en tiempo de compilación.
  + Dinámica: los tipos de variables y expresiones no se conocen hasta el momento de ejecución.
* Persistencia: Define que todo estado de un objeto va a perdurar en el tiempo y el espacio con independencia de quien lo creo. El ciclo de vida del objeto queda establecido desde que el objeto es creado hasta que es matado.
* Concurrencia: Varios objetos pueden ejecutarse simultáneamente. Tipos:
  + Concurrencia pesada: un proceso del S.O está ejecutando solo un elemento del sistema en una porción propia de memoria. Prioriza la seguridad e independencia de los procesos.
  + Concurrencia liviana: un proceso del S.O está ejecutando varios elementos del sistema, compartiendo una porción de memoria. Para el intercambio de datos la concurrencia liviana es más indicada.

**Objeto**:

Cualquier cosa real o abstracta sobre la cual se posee una comprensión intelectual. Posee:

* *Estado:* Conjunto de características y propiedades, junto con sus valores actuales.
* *Comportamiento*: Conjunto de acciones y reacciones del objeto.
* *Identidad:* Lo que distingue el objeto de los demás.
* *Tipos de relaciones* entre objetos:
  + Enlace: conexión física o conceptual, denota una relación de igual a igual. Se envían mensajes.
  + Agregación: denota una relación jerárquica del tipo “Todo-parte”.
    - Con contención física (composición). Los ciclos de vida del “Todo” y las “Partes” están íntimamente relacionado, no hay existencia de las “Partes” sin un todo.
    - Sin contención física. Las “Partes” tiene existencia independientemente del “Todo”.

**Mensajes entre objetos:** Utilizados para colaborar entre sí a través de los enlaces. Típicamente son unidireccionales. Poseen roles:

* Cliente/Actor: Puede enviarles mensajes a otros objetos, pero los demás no pueden enviarle mensajes a él.
* Servidor: Puede recibir mensajes de otros objetos, pero no puede enviar mensajes.
* Agente: puede actuar como actor o servidor.

Deben estar sincronizados para poder comunicarse:

* Secuencial: El buen funcionamiento del objeto pasivo (servidor) está garantizada por la presencia de un único objeto activo (cliente).
* Vigilada: El buen funcionamiento del objeto pasivo está garantizada por la presencia de múltiples hilos de control, los objetos activos colaboran para lograr la exclusión mutua.
* Síncrona: El buen funcionamiento del objeto pasivo está garantizada por la presencia de múltiples hilos de control, el servidor es quien administra la exclusión mutua.

**Clase**:

Especificación que representa a un conjunto de objetos reales que comparten una estructura y un comportamiento en común. Podemos identificar dos partes:

* Interfaz: Proporciona una visión externa. Enfatiza la abstracción y oculta los detalles de la implementación.
* Implementación: Proporciona una visión interna. Enfatiza el encapsulado.
* Tipos de relaciones entre clases:
  + **Asociación**: Relación bidireccional entre dos clases. Posee cardinalidad, la cual establece la cantidad de elementos de un tipo que se relacionan con elementos de otro tipo. Cardinalidades: uno a uno, uno a muchos y muchos a muchos. Se da cuando las clases se conectan en forma conceptual.
  + **Herencia** (“es-un”): Relación en donde una o más subclases pueden heredar la estructura y comportamiento de una o más superclases.
    - Simple: dos o más subclases heredan estructura y comportamiento de una superclase.
    - Múltiple: dos o más subclases heredan estructura y comportamiento de dos o más superclases.
  + **Agregación:** Denota una relación jerárquica del tipo “Todo-parte”. Mantiene un paralelismo con el mismo concepto tratado para los objetos.
    - Por valor: Los ciclos de vida están relacionado y el todo se encarga de crear y destruir las partes.
    - Por referencia: Las partes son independientes del todo.
  + **Uso**: Asociación refinada donde se establece quien opera como cliente y quien opera como servidor. Se establece quien va a hacer uso de quien.

Una clase también es una construcción que permite obtener y representar las representaciones que se necesitan en una solución orientada a objetos. Podrá contener:

* Campos: Variable de cualquier tipo que se declara en una clase.
* Propiedades: Miembro que proporciona un mecanismo flexible para leer, escribir o calcular el valor de un campo privado, es decir, para acceder a ellos. ***Get***: devolver valor. ***Set***: asignar valor.
* Métodos: Bloque de código que tiene una serie de instrucciones. Tiene un nombre, **una firma** (nivel de acceso, valor de retorno, nombre del método y parámetros).   
  Parámetros por Valor (Copia de los datos). Parámetros por referencia (Posición de memoria de los datos)
* Eventos: Mecanismos que permiten que un objeto reaccione ante un estímulo externo.

**Clases Anidadas:** Clase definida dentro de otra clase.

**Clases Abstractas**: Indica que a lo que se le esté aplicando carece de una implementación o bien la implementación es incompleta. Se utiliza para indicar que una clase solo pretende ser una clase base de otras clases. Las clases abstractas tienen las siguientes características:

• No puede ser instanciada.

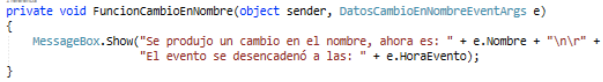
• Puede contener métodos abstractos.

• No es posible tener una clase **abstracta** y a la vez sellada. El modificador **sealed** evita que una clase se herede y el modificador **abstract** requiere que una clase se herede.

• Una clase no abstracta derivada de una clase abstracta debe implementar todos los métodos abstractos heredados.

***Eventos***: Mecanismo de enlace tardío, permiten que un objeto reaccione ante un estímulo externo. Los objetos pueden suscribirse a un evento y ser notificados cuando se produce el mismo. La suscripción a un evento también crea un acoplamiento entre los dos objetos.

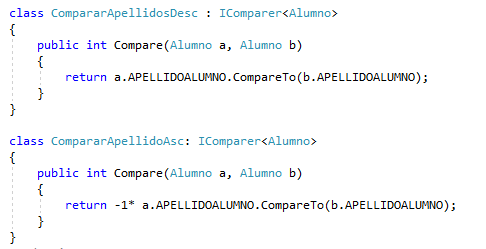
**Evento Personalizado:**

Definición del evento:  
  
Desencadenamiento:  
  
Suscripción:  
  


**Interfaces:** Se utilizan para poder definir propiedades, métodos y eventos que luego deberán implementar aquellas clases que implementen la interfaz.

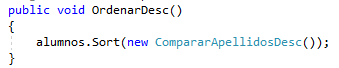
* Las interfaces tipan, es decir, cuando una clase implementa una interfaz también adquiere un nuevo **tipo**, que es el **tipo** de la interfaz.
* Permiten que la programación actual pueda interactuar con piezas de código del futuro.
* Pueden crear polimorfismo.

**IComparer**



Implemento lo que tiene

Creo la interfaz



La ejecuto