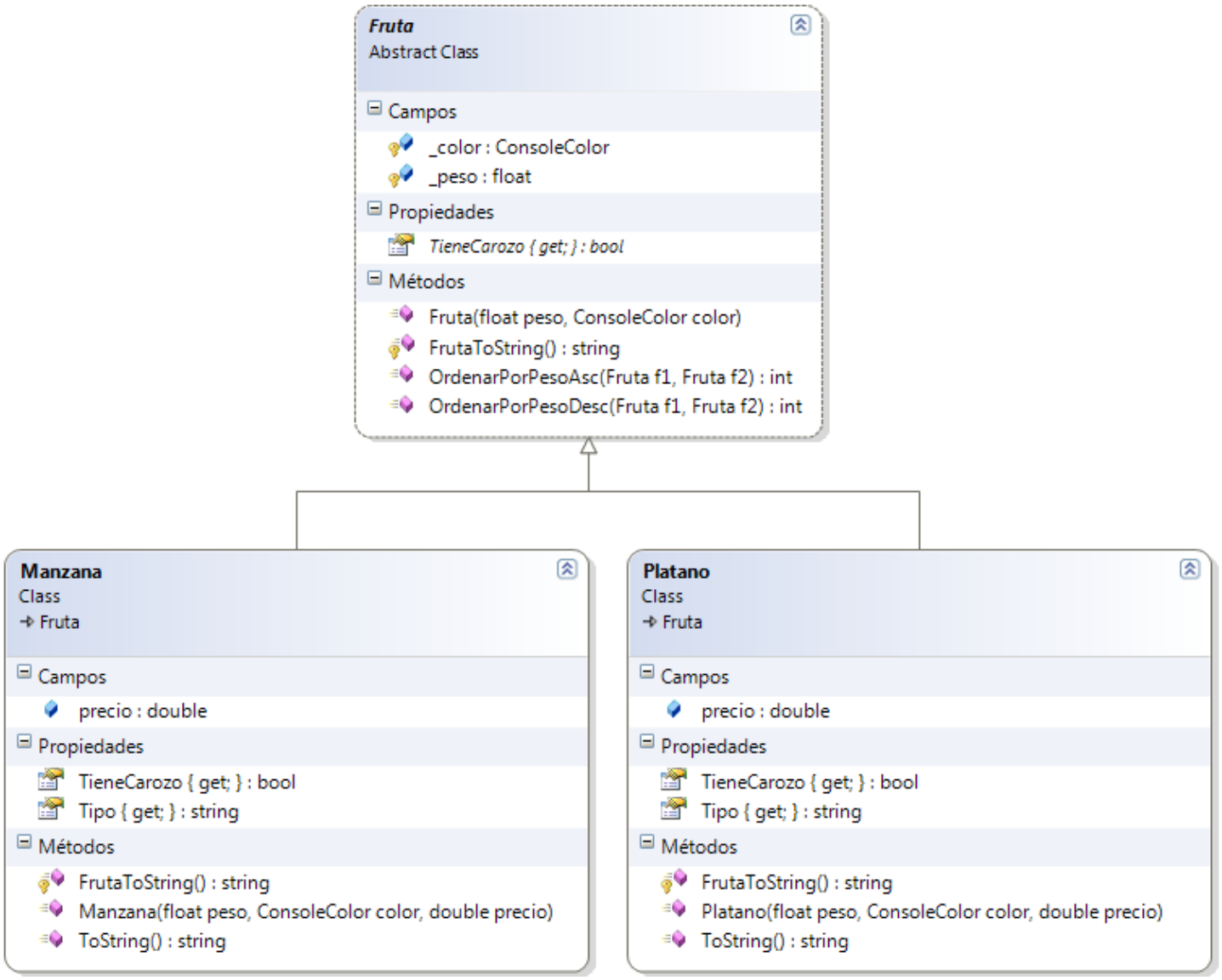
Universidad Tecnológica Nacional Facultad Regional Avellaneda										
Técnico Superior en Programación - Técnico Superior en Sistemas Informáticos										
Materia:										
Apellido:					Fecha:					
Nombre:					Docente(2):	NEINER, MAXIMILIANO				
División:					Nota(2):					
Legajo:					Firma(2):					
Instancia(1):	**PP**		**RPP**		**SP**		**RSP**		**FIN**	

(1) Las instancias validas son: 1er Parcial (**PP**), Recuperatorio 1er Parcial (**RPP**), 2do Parcial (**SP**), Recuperatorio 2do Parcial (**RSP**), Final (**FIN**) . Marque con una cruz. **(2)** Campos a ser completados por el docente.

Codificar en C Sharp la siguiente jerarquía de clases en un proyecto de tipo ***Class Library***.



Clase Fruta

Propiedad

TieneCarozo (abstracta; sólo lectura).

Métodos

FrutaToString:(de instancia; protegido; virtual) retorna la representación de la clase en formato string.

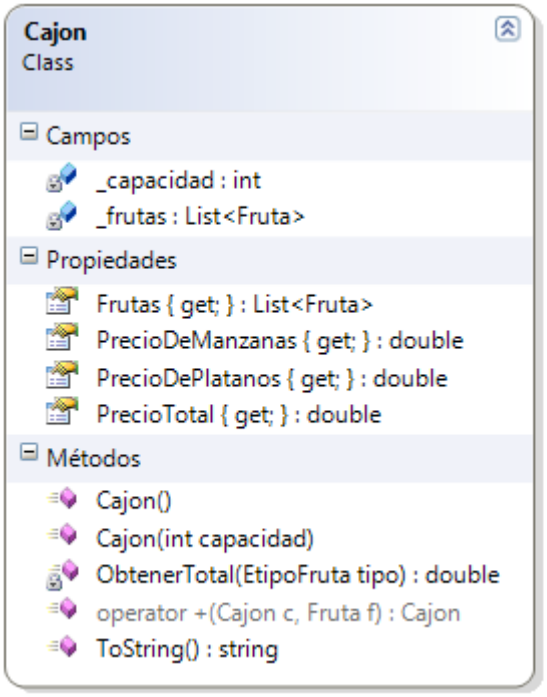
OrdenarPorPesoAsc:(de clase; público) será utilizado en el método Sort de List<Fruta>. Permite comparar dos objetos de tipo Fruta.

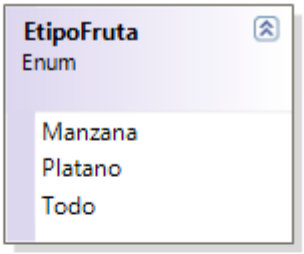
OrdenarPorPesoDesc:(de clase; público) ídem anterior. Reutilizar el método anterior (sólo una línea de código).

Clase Manzana (hereda de Fruta)
Propiedades
TieneCarozo:(sólo lectura) retornará **true**.
Tipo: (sólo lectura) retornará "**Manzana**".
Métodos
FrutaToString: Se deberá reutilizar agregándole los campos propios.
ToString: Invocará a FrutaToString.

Clase Platano (hereda de Fruta)
Propiedades
TieneCarozo:(sólo lectura) retornará **false**.
Tipo: (sólo lectura) retornará "**Plátano**".
Métodos
FrutaToString: Se deberá reutilizar agregándole los campos propios.
ToString: Invocará a FrutaToString.

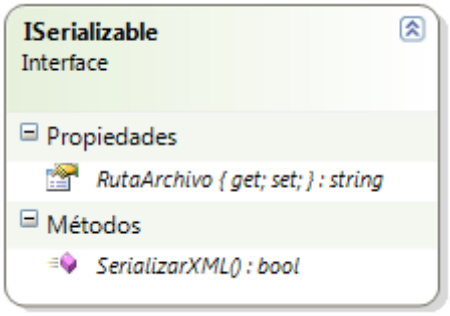
Agregar los siguientes elementos al proyecto:





Clase Cajon
Propiedades
Frutas:(sólo lectura) expone al atributo de tipo List<Fruta>.
PrecioDeManzanas:(sólo lectura) invoca al método ObtenerTotal, pasándole como parámetro EtipoFruta.Manzana.
PrecioDePlatanos:(sólo lectura) invoca al método ObtenerTotal, pasándole como parámetro EtipoFruta.Platano.
PrecioTotal:(sólo lectura) invoca al método ObtenerTotal, pasándole como parámetro EtipoFruta.Todo.
Constructores
Por default: es el único que se encargará de inicializar la lista de frutas.
Métodos
ObtenerTotal: (privado) retorna el acumulado de precios según el parámetro de tipo EtipoFruta que reciba.
ToString: Mostrará en formato de tipo string, la capacidad y el listado de todas las frutas contenidas en el cajón.
Reutilizar código.
Sobrecarga de operador
(+) Será el encargado de agregar frutas al cajón, siempre y cuando no supere la capacidad del mismo.

Agregar al proyecto el siguiente elemento



Interface ISerializable

Propiedad

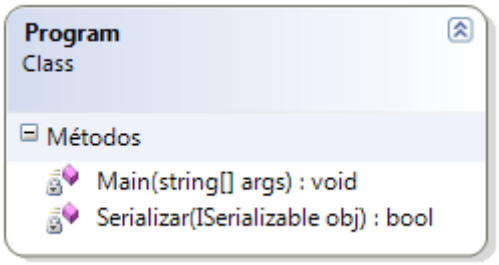
RutaArchivo: Establece/obtiene el path dónde se guardará el archivo XML.

Método

SerializarXML: serializará a XML la instancia que lo invoque, retornando true si lo pudo hacer; false, caso contrario.

Implementar la interface ISerializable en Manzana y Cajón.

Clase Program



Métodos

Main: punto de inicio de la aplicación. Se deberá:

a) Crear tres instancias de Manzana, tres instancias de Platano y una de Cajon (estableciendo el valor 5 como capacidad del mismo).
b) Intentar agregar las seis instancias de frutas. Ejemplo miCajon += miManzana;
c) Mostrar el cajón.
d) Ordenar el cajón (ascendentemente) y mostrarlo.
e) Ordenar el cajón (descendentemente) y mostrarlo.
f) Mostrar el precio total de manzanas, de plátanos y el total del cajón.
g) Serializar una manzana
h) Serializar el cajón de frutas.

Serializar:(privado; de clase) recibe como parámetro un objeto de tipo ISerializable y retornará true si pudo serializar; false, caso contrario. Reutilizar código (sólo una línea de código.)