**Collection**

* Can be used to store multiple objects
* < > this is called Generic type syntax in c#
* Class is a achchuwa and breaks are the objects of same shape as achchuwa. So achchuwa is the base for breaks
* Constructor is responsible for creating a new object of a class.
* Class are reference type. Which means main memory position only store the reference and real value is stored in heap. Heap is the place where objects reside.
* There is a default constructor for every class implicitly.
* Can write your own constructors
* Hiding some of the details that how software works this allows to use the components others written without knowing the inside details – Encapsulation
* Public key word implement abstraction
* By convention if you expose a private member of a class that member’s name need to start with a uppercase latter
* Encapsulation in simple

private List<Books> \_books;

public List<Books> GetBook()

{

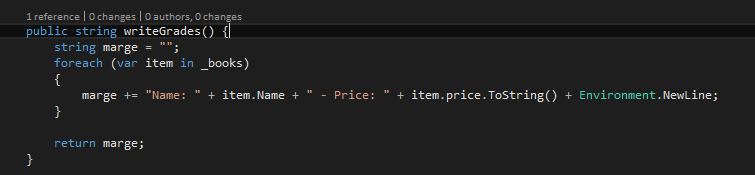
return \_books;

}

* Static keyword use to access a class member without creating an instance of that class
* General example MessageBox.Show()

OOP

* There are 3 Pillars of oop
  + Encapsulation
  + Inheritance
  + Polymorphism
* Encapsulation is the key for creating large enterprise application
* IT helps hide the complexity of the program.



* If we wrote this inside main method it will take length and when 10 like this combined it will hard to read.
* So this will help to build things those are **logically grouped together**
* **Inheritance** means define relationship in 2 classes such that one class will takes other classes methods and properties
* You can only inherit from one class in C# but you can inherit nested in C#
* You can assign a derived class variable to base class variable and its legal
* This key word is use to show the methods a properties of the current class including base classes’ types
* Base will give me list of methods and properties those are in the base class.
* **Protected** is a keyword that it allows to access a class member of its base class safely
* **Poly** means deferent forms in Greek. so polymorphism means when things are in many forms
* So objects can behave differently using polymorphism
* System.object is the ultimate base class of all. So I can assign any type of instance to a variable of type object

**Abstract**

* You cannot make a instance of an abstract class cus it’s implementation not complete.
* Virtual members are real working members. Cus they are already implemented and ready to go. if you want you can change the behavior of those
* Abstract members need to implement on derived class
* The use is you can derive as much as you want from an abstract class and assign those child class instance to the each of anyone

**Interfaces**

* Interface does not have an implementation at all.
* Interface ensure the very derive class guarantee to have members of the interface
* When implementing interface all members of the interface must implement on derived class
* A class can inherit from one or many interfaces

Constraints on Type Parameters

When you define a generic class, you can apply restrictions to the kinds of types that client code can use for type arguments when it instantiates your class. If client code tries to instantiate your class by using a type that is not allowed by a constraint, the result is a compile-time error. These restrictions are called constraints. Constraints are specified by using the where contextual keyword. The following table lists the six types of constraints: