1. Create all Modifiers,Access specifiers, method parameters examples in visual studio.
2. Define each keyword covered as on today in one liner in an excel sheet.

|  |  |  |
| --- | --- | --- |
| Key Word (Modifiers) | | |
| **abstract** | Abstract is a modifier indicates that the thing being modified has a missing or incomplete implementation. Can be used with classes, methods, properties, indexers, and events. It is used in the declaration as an indicator that a class is intended only to be a base class of other classes not instantiated on its own. Abstract members must be implements by non-abstract classes that derive from the abstract class. | **abstract** class Dinosaur  {  public abstract string GetDinosaur();  }  class Stegosaurus : Dinosaur  {  string roar;  public Stegosaurus(string n) => roar = n;  public override string GetDinosaur() => roar;  static void Main()  {  var roar = new Stegosaurus("ROAR!!!!!!");  Console.WriteLine($"Sound = {roar.GetDinosaur()}");  }  }//Ran in VS Community as well |
| **const** | The const keyword is used to declare a constant field or a constant local. They are not variables as they cannot be modified. They can be numbers, Booleans, strings, or nulls. They are propagated so for changes codes will have to be recompiled to see updates. | **const** int = 0;  public **const** string = “Never!”  public **const** bool set = true;  private **const** string = null; |
| **readonly** | It is a keyword modifier that can be used 4 ways, a field declaration, as a read only struct to make it immutable, as an instance member declaration in a struct, and as a modifier on a method return. | **readonly** string strang = "I Heart reading";  public **readonly** struct StructRead  {  private **readonly** string StringRead;  } //I tried to do return readonly, but that was out of my league |
| **static** | The keyword modifier is part of the using\_static directive. When used the member it is declared on belongs to the type itself rather than the object. It can’t be used with indexers or finalizers and it can’t be instantiated. You have to access using dot | **static** class IAmStatic  {  public **static** string StaticString = "Static.";  public **static** void MoreStatic()  {  Console.WriteLine("Nothing ever changes.");  }  } |
| **sealed** | When applied to a class the sealed modifier prevents other classes from inheriting from it. | **sealed** class SealedClass  {  public string Sealing = "All sealed";  }  class DeSealing  {  static void Main()  {  var opening = new SealedClass();  opening.Sealing = "open sealed";  Console.WriteLine($"I just {opening.Sealing}!");  }  }//Accesses the sealed class. |
| **extern** | The access modifier is used toi declare a method that is implemented externally, this is commonly used for DllImport attribute when you are using a interop services (not really sure what that means yet) to call into unmanaged code. The method must also be called static. It is used to define an external assembly as well. | class ExternTest  {  [DllImport("User32.dll", CharSet = CharSet.Unicode)]  public static extern int MessageBox(IntPtr h, string m, string c, int type);  static int Main()  {  string myString;  Console.Write("Enter your message: ");  myString = Console.ReadLine();  return MessageBox((IntPtr)0, myString, "My Message Box", 0);  }  }//This is a code I used from tutorials online, but it is not something can I use yet. |

|  |  |  |
| --- | --- | --- |
| Access Modifiers | | |
| **public** | It is a keyword that is an access modifier for types and type members. It is the most permissive access level with no restrictions on accessing. | class NotPicky  {  **public** string access = “Come on in.”;  } |
| **private** | Is a keyword that is an access modifier that is the least permissive access level. They are only accessible within the body, struct, or the class or the strict in which they are declared. | class MembersOnly  {  **private** string access = “You’re not on the list so just move along.”;  } |
| **protected** | Is a keyword that is an access modifier that means it is only available within the class or the derived class instances. | public class TheBand  {  **protected** string BarAccess = "The bar on the bus.";  }  public class Groupie:TheBand  {  public static void Main()  {  var LetMeIn = new Groupie();  LetMeIn.BarAccess = "Your one of us!";  Console.WriteLine(LetMeIn.BarAccess);  }  }//Dotnetfiddle wouldn’t run so had to test in VS Comm |
| **internal** | Internal is a keyword for an access modifier, in internal they are only accessible within files of the same assembly. | public class InternalExample  {  internal static string StringInternal = "Sup?";  }  class InternalAccess  {  static void Main()  {  var MyInternal = new InternalExample(); // CS0122  Console.WriteLine(InternalExample.StringInternal);  }  }//Ran in VS Community as well |