Advance Information

*about*

C# Programming language

|  |  |
| --- | --- |
| Idea 1: | ‘params’ keyword |
|  | ‘params’ keyword used in method parameter to allow unknown number of variable pass into to the methords.  Ref: <http://msdn.microsoft.com/en-us/library/w5zay9db.aspx>  Example:  class Program  {  static void Main(string[] args)  {  **Country canada = new Country(new City("St. John's", 50000),**  **new City("Mt Pearl", 20000), new City("Toronto", 300000));**  }  }  public class Country  {  List<City> cities = new List<City>();  public Country(**params City[] cities**)  {  this.cities = cities.ToList();  }  }  public class City  {  private long population;  private string name;  public City(string name, long population)  {  this.name = name;  this.population = population;  }  } |
| Idea 2: | ‘implicit’ Operation |
|  | ‘implicit’ operation use to create user define conversion type.  Ref: <http://msdn.microsoft.com/en-us/library/z5z9kes2.aspx>  Example:  class Digit  {  public Digit(double d) { val = d; }  public double val;  // ...other members  // User-defined conversion from Digit to double  **public static implicit operator double(Digit d)**  **{**  **return d.val;**  **}**  // User-defined conversion from double to Digit  **public static implicit operator Digit(double d)**  **{**  **return new Digit(d);**  **}**  }  class Program  {  static void Main(string[] args)  {  Digit dig = new Digit(7);  //This call invokes the implicit "double" operator  **double num = dig;**  //This call invokes the implicit "Digit" operator  **Digit dig2 = 12;**  Console.WriteLine("num = {0} dig2 = {1}", num, dig2.val);  Console.ReadLine();  }  } |
| Idea 3: | ‘this’ operator as array of |
|  | ‘this’ operator can be use as reading an element from an index.  Ref: <http://msdn.microsoft.com/en-us/library/dk1507sz.aspx>  Example:  class Program  {  static void Main(string[] args)  {  Country canada = new Country(new City("St. John's", 50000),  new City("Mt Pearl", 20000), new City("Toronto", 300000));  Console.WriteLine(**canada[2].ToString()**);  Console.Read();  }  }  public class Country  {  List<City> cities = new List<City>();  public Country(params City[] cities)  {  this.cities = cities.ToList();  }  **public City this[int n]**  **{**  **get { return cities[n]; }**  **}**  }  public class City  {  private long population;  private string name;  public City(string name, long population)  {  this.name = name;  this.population = population;  }  public override string ToString()  {  return string.Format("{0}:{1}", this.name, this.population);  }  } |
| Idea 4: | Extension Modes |
|  | Extension methods enable you to add methods to existing types without creating a new derived type.  Ref: <http://msdn.microsoft.com/en-CA/library/bb383977.aspx>  Example 1:  class Program  {  static void Main(string[] args)  {  string email = "test@test.com";  if (**email.IsAnEmailAddress())**  {  //Do sometihing ...  }  }  }  public static class EmailExtension  {  **public static bool IsAnEmailAddress(this string str)**  **{**  **Regex rex = new Regex(@"^[\w-\.]+@([\w-]+\.)+[\w-]{2,4}$");**  **return rex.IsMatch(str);**  **}**  }  Example 2: |