|  |
| --- |
| Day 10(4feb) Assignment  by Ramakrishna |

|  |
| --- |
| 1. Write the two points discussed about inheritance in the class. |
| * Inheritance is process of re-using baseclass method in derived class * Inheritance main goal is : re-usability to remove duplicate code   Types:   * Single Inheritance * Multiple Inheritance * multilevel Inheritance |

|  |
| --- |
| 2 .Why multiple inheritance is not supported for classes in in C# |
| * The C# compliler is designed ina way that it will not support multiple inheritance ,because it causes ambiguity (derived class having more than one parent class that defines property[s] and/or method[s] with the same name.)of methods from different base class * this cause by diamond shape problems of two classes. if two classes B and C is inheritance from A ,so that multiple inheritance is not possible in C# |

|  |
| --- |
| 3. Write example code for: a. Single inheritance |
| Code: using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace single\_inhertiance  {  class Algebra  {  public int add(int a, int b)  {  return a + b;  }  public int Sub(int a, int b)  {  return a - b;  }  }  class Totalmaths:Algebra  {  public int mul(int a, int b )  {  return a \* b;  }  }  internal class Program  {  static void Main(string[] args)  {  Totalmaths tm = new Totalmaths();  Console.WriteLine();  Console.ReadLine();  }  }  } |

|  |
| --- |
| b. Multi level inheritance |
| Code: using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace miltiple\_inhertiance  {  class Algebra  {  public int add(int a, int b)  {  return a + b;  }  public int Sub(int a, int b)  {  return a - b;  }  }  class Totalmaths : Algebra  {  public int mul(int a, int b)  {  return a \* b;  }    }  class Allsubjects : Totalmaths  {  public string water()  {  return "h2O";  }  }  internal class Program  {  static void Main(string[] args)  {  Allsubjects obj = new Allsubjects();  Console.WriteLine();  Console.ReadLine();  }  }  } |

|  |
| --- |
| 4 . Write sample code for method overloading |
| Code: Code: using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace over\_loading  {  class Algebra  {  public int Add(int a, int b)  {  return a + b;  }  public int Add(int a, int b, int c)  {  return a + b + c;  }  public int Add(int a, int b, int c, int d)  {  return a +b + c + d;  }  }  internal class Program  {  static void Main(string[] args)  {  Algebra obj = new Algebra();  Console.WriteLine();  }  }  } |

|  |
| --- |
| 5. Research and write sample code for method overriding  using virual, override keyword. |
| Code: using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace over\_riding\_\_\_\_virtual  {  public class abc  {  public virtual int data(int a,int b)  {  return a + b;  }  }  public class pqr :abc  {  public override int data(int a ,int b)  {  return(a + b)\*2;  }  }  internal class Program  {  static void Main(string[] args)  {  pqr pqr = new pqr();  Console.WriteLine();  Console.ReadLine();  }  }  } |

|  |
| --- |
| 6 .Write sample code for method overloading |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace over\_riding  {  class englishmessage  {  public void printHI()  {  Console.WriteLine("Hi");  }  public void printHello()  {  Console.WriteLine("Hello");  }  public void printGoodevening()  {  Console.WriteLine("Goodevening");  }  }  class Hindimessage : englishmessage  {  public new void printGoodevening()  {  Console.WriteLine("GV");  }  }    internal class Program  {  static void Main(string[] args)  {  Hindimessage Hm = new Hindimessage();  Console.WriteLine();  }  }  } |
|  |

|  |
| --- |
| 7.What is polymorphism. |
| * polymorphismIt is the ability to take take many forms * the ability of objects of different types to provide a unique interface for different implementations of methods. It is usually used in the context of late binding, where the behavior of an object to respond to a call to its method members is determined based on object type at run time. |

|  |
| --- |
| 8. Pictorially represent 3 types of inheritance discussed  in the class. |
|  |

|  |
| --- |
| multiple inhertiance |

|  |
| --- |
|  |