|  |
| --- |
| **Day-10 assignment**  **By**  **Bhanu Rama Krishna Prakash Jakkamsetti**  **3/2/2022** |

1. Write the two points discussed about inheritance in the class.

* Inheritance is the process of Re-using base class method to the derived class.
* Inheritance main goal is Re-usability, and to remove duplicate code.

2. Write example code for:

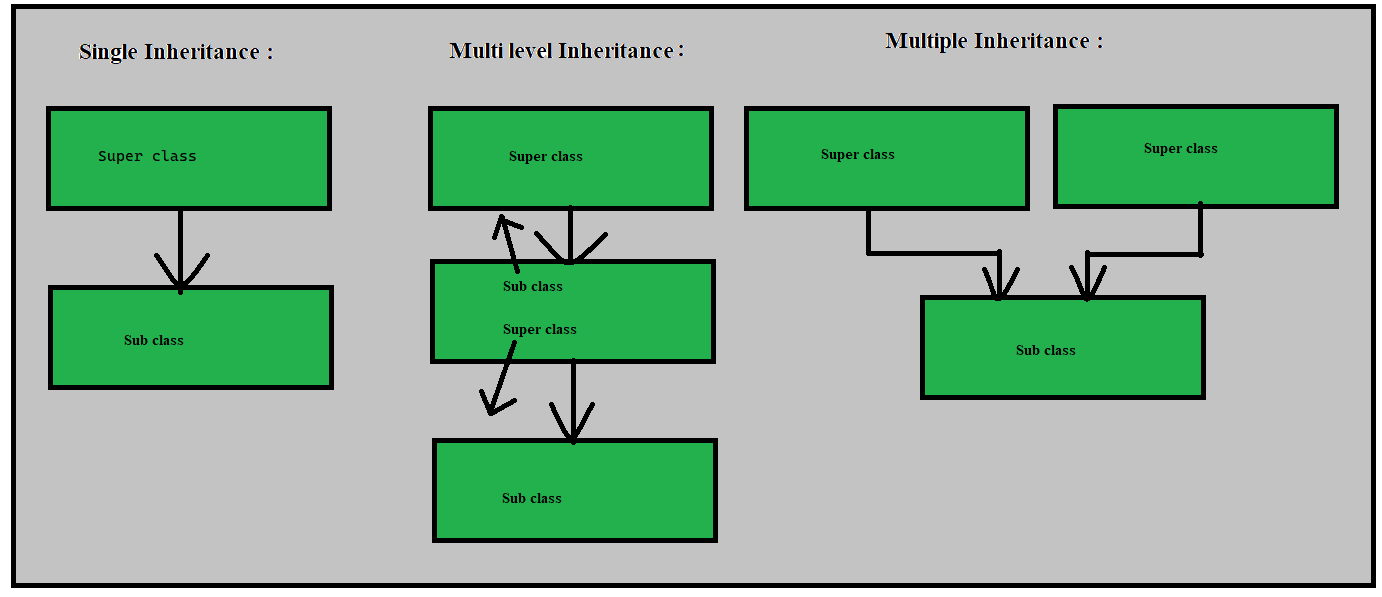
a. Single inheritance

b. Multi level inheritance

|  |
| --- |
| 1. Single inheritance |
| Code: |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Day10\_project1  {  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Author:Bhanu Rama Krishna Prakash Jakkamsetti  \*Purpose: creting Single inheritance  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  class Algebra  {  /// <summary>  /// addition operation  /// </summary>  /// <param name="a"></param>  /// <param name="b"></param>  /// <returns>a+b</returns>  public int Add(int a, int b)  {  return a + b;  }  /// <summary>  /// difference operation  /// </summary>  /// <param name="a"></param>  /// <param name="b"></param>  /// <returns>a-b</returns>  public int Sub(int a, int b)  {  return a - b;  }  }  class Totalmaths : Algebra  {  /// <summary>  /// multiplication operation  /// </summary>  /// <param name="a"></param>  /// <param name="b"></param>  /// <returns>a\*b</returns>  public int Mul(int a,int b)  {  return a \* b;  }  }  internal class Program  {  static void Main(string[] args)  {  Totalmaths t = new Totalmaths();  Console.WriteLine( t.Add(20,10));  Console.WriteLine(t.Sub(20, 10));  Console.WriteLine(t.Mul(20, 10));  }  }  } |
| Output: |
|  |

|  |
| --- |
| 1. Multi level inheritance |
| Code: |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Day10\_project2  {/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Author:Bhanu Rama Krishna Prakash Jakkamsetti  \*Purpose: multi level inheritance  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  class Algebra  {  /// <summary>  /// addition operation  /// </summary>  /// <param name="a"></param>  /// <param name="b"></param>  /// <returns>a+b</returns>  public int Add(int a, int b)  {  return a + b;  }  /// <summary>  /// difference operation  /// </summary>  /// <param name="a"></param>  /// <param name="b"></param>  /// <returns>a-b</returns>  public int Sub(int a, int b)  {  return a - b;  }  }  class Totalmaths :Algebra  {  /// <summary>  /// multiplication operation  /// </summary>  /// <param name="a"></param>  /// <param name="b"></param>  /// <returns>a\*b</returns>  public int Mul(int a, int b)  {  return a \* b;  }  }  class Allsub :Totalmaths  {  /// <summary>  /// creating string  /// </summary>  /// <returns>CH4</returns>  public string Methane()  {  return "CH4";  }  }  internal class Program  {  static void Main(string[] args)  {  Allsub t = new Allsub();  Console.WriteLine(t.Add(20, 10));  Console.WriteLine(t.Sub(20, 10));  Console.WriteLine(t.Mul(20, 10));  Console.WriteLine(t.Methane());  }  }  } |
| Output: |
|  |

3. Pictorially represent 3 types of inheritance discussed in the class.



4. Why multiple inheritance is not supported for classes in C#

* Due to diamond problem. The diamond problem is an ambiguity that arises when two classes B and C inherit from A, and class D inherits from both B and C. If a method in D calls a method defined in A (and does not override it), and B and C have overridden that method differently, then via which class does it inherit: B, or C.

5. What is polymorphism.

* Polymorphism is the ability of an object to take on many forms.
* Polymorphism is a Greek word, meaning "one name many forms". In other words. "Poly" means many and "morph" means forms. Polymorphism provides the ability to a class to have multiple implementations with the same name. It is one of the core principles of Object Oriented Programming after encapsulation and inheritance. In this article, you'll learn what polymorphism is, how it works, and how to implement polymorphism in C#.
* Method overloading
* Method overriding

6. Write sample code for method overloading

|  |
| --- |
| Method overloading |
| Code: |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Day10\_project3  {  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Author:Bhanu Rama Krishna Prakash Jakkamsetti  \*Purpose: Method overloading  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  class Algebra  {  /// <summary>  /// addition operation with 2 paramaeters  /// </summary>  /// <param name="a"></param>  /// <param name="b"></param>  /// <returns>a+b</returns>  public int Add(int a, int b)  {  return a + b;  }  /// <summary>  /// addition operation with 3 parameters  /// </summary>  /// <param name="a"></param>  /// <param name="b"></param>  /// <param name="c"></param>  /// <returns>a+b+c</returns>  public int Add(int a, int b ,int c)  {  return a+b+c;  }  /// <summary>  /// addition operation with 4 parameters  /// </summary>  /// <param name="a"></param>  /// <param name="b"></param>  /// <param name="c"></param>  /// <param name="d"></param>  /// <returns>a+b+c+d</returns>  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 t = new Algebra();  Console.WriteLine(t.Add(20,10));  Console.WriteLine(t.Add(20,10,30));  Console.WriteLine(t.Add(20,10,30,40));  }  }  } |
| Output: |
|  |

7. Write sample code for method overriding [ using new key word ]

|  |
| --- |
| Method overriding |
| Code: |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Day10\_project4  {  class ENglanguage  {  public void Printhii()  {  Console.WriteLine("hii");  }  public void Printhello()  {  Console.WriteLine("hello");  }  public void Printgoodmrng()  {  Console.WriteLine("good morning");  }  }  class Tellanguage :ENglanguage  {  public void Printgoodmrng()  {  Console.WriteLine("shubudhayam");  }  }  internal class Program  {  static void Main(string[] args)  {  Tellanguage t = new Tellanguage();  t.Printhii();  t.Printhello();  t.Printgoodmrng();  }  }  } |
| Output: |
|  |

8. Research and write sample code for method overriding using virtual, override keyword.

|  |
| --- |
| Overriding by using virtual and override keyword |
| Code: |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Day10\_project5  {  class ENglanguage  {  public void Printhii()  {  Console.WriteLine("hii");  }  public void Printhello()  {  Console.WriteLine("hello");  }  public virtual void Printgoodmrng()  {  Console.WriteLine("good morning");  }  }  class Tellanguage : ENglanguage  {  public override void Printgoodmrng()  {  Console.WriteLine("shubudhayam");  }  }  internal class Program  {  static void Main(string[] args)  {  ENglanguage e = new ENglanguage();  e.Printhello();  e.Printhii();  e.Printgoodmrng();  Tellanguage t = new Tellanguage();  t.Printhii();  t.Printhello();  t.Printgoodmrng();  }  }  } |
| Output: |
|  |