**Capabilities Covered:**

* Use C# Programming to implement business layer of your application
* Handle exceptions to maintain the normal flow of the application.
* Use appropriate classes using collections to handle group of business objects
* Use ADO.NET features to connect to database and perform a simple query operation

**Assignment 1: Date Formats**

**Pe**

**Estimated time: 10 Mins**

**Problem description:** Write aC# Program to Display today’s date in the below Formats :

yyyy-MM-dd, dd-MMM-yy, MM/dd/yyyy, MM/dd/yy, M/d/yy

and take 10 days before today’s date, compare it to today’s date and print the date which occurs first.

Below is sample code to display the date in various formats:

|  |
| --- |
| Console.WriteLine(date.ToString("yyyy-MM-dd")); |

Below is the sample code to take the 10 days before today’s date:

|  |
| --- |
| date.AddDays(-10); |

**Assignment 2: Method Overriding**

**Problem description:**

* Complete main method to display the output like below:

**Estimated time: 30 Mins**

Name : Ram

RNO : 44

Student ID : ABC

Student Address : USA

|  |
| --- |
| public class Person  {  protected string RNO = "44";  protected string name = "Ram";  public virtual void GetInfo()  {  Console.WriteLine("Name: {0}", name);  Console.WriteLine("RNO: {0}", RNO);  Console.WriteLine();  }  }  class Student : Person  {  public string id = "ABC";  public override void GetInfo()  {  base.GetInfo();  Console.WriteLine("Student ID: {0}", id);  }  }  class Stud : Student  {  private string StudentAddress = "USA";  public void GetInfo()  {  base.GetInfo();  Console.WriteLine("Student Address: {0}", StudentAddress);  }  }  class TestClass  {  public static void Main()  {  //Implement the code as instructed above  }  } |

* Below scenarios generates an error. Debug and find the errors and explain why errors are coming:

|  |
| --- |
| **1.**  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  namespace Generics  {  class A  {  public virtual void show()  {  Console.WriteLine("Hello: Base Class!");  Console.ReadLine();  }  }  class B : A  {  public virtual(or ‘ new’) override void show()  {  Console.WriteLine("Hello: Derived Class!");  Console.ReadLine();  }  }  class C : B  {  public new void show()  {  Console.WriteLine("Am Here!");  Console.ReadLine();  }  }  class Polymorphism  {  public static void Main()  {  C c1 = new C();  c1.show();  A a2 = new B();  a2.show();  A a3 = new C();  a3.show();  B b3 = new C();  b3.show();  }  }  }  **2.**  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  namespace Generics  {  class A  {  public virtual void show()  {  Console.WriteLine("Hello: Base Class!");  Console.ReadLine();  }  }  class B : A  {  public sealed override void show()  {  Console.WriteLine("Hello: Derived Class!");  Console.ReadLine();  }  }  class C : B  {  public override void show()  {  Console.WriteLine("Am Here!");  Console.ReadLine();  }  }  class Polymorphism  {  public static void Main()  {  A a1 = new A();  a1.show();  B b1 = new B();  b1.show();  C c1 = new C();  c1.show();  A a2 = new B();  a2.show();  A a3 = new C();  a3.show();  B b3 = new C();  b3.show();  }  }  } |

**Note: Focus should be on the highlighted keywords as these are the main features of C# in inheritance, test other scenarios also.**

**Assignment 3: Delegates**

**Estimated time: 15 Mins**

* **Problem description:** Write a c# programme to display results of addition and subtraction of two numbers using delegates.

**Step 1:**

Create a delegate.

**public** **delegate** **<datatype>** <DelegateName>(<**parameters>**);

**Step 2:**

Create a Results class which contains summation and subtraction method

**Step 3:**

In main method, create Result class object and delegate object and call the summation method through delegate object like below sample code:

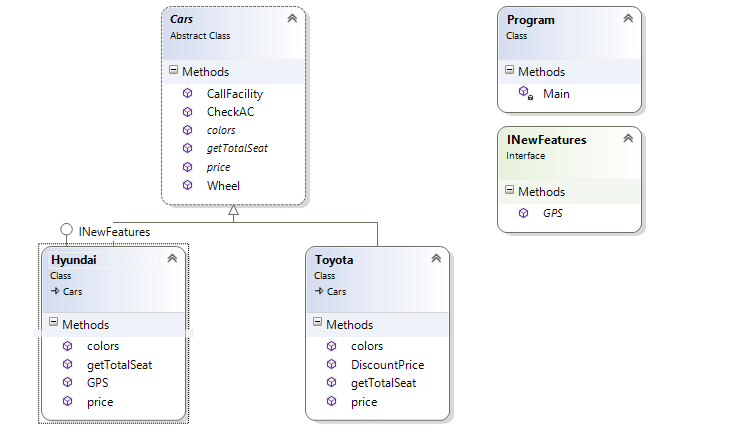
|  |
| --- |
| Delegate delegateobj= new Delegateclass(method)  Use multicaste operator  Delegateobj (method arguments) |

* **Problem description:** Whenever a new lead joins Orchard , People function group should get the details of that lead like below:

**Assignment 4: Abstract and Interface**

**Estimated time: 15 Mins**

**Problem description:** Write a C# programming to implement following class diagram where Hyundai and Toyota are two car companies which has some common features but now Hyundai wants to add one new feature “GPS” to their car.



Hints:

* Define an abstract class as Car.
* Put all the common functionality in simple methods and all the methods whose implementation is different but name is same. Make them Abstract method.
* To add new feature create an interface and implement that in the required class(Hyundai)

Note: From this implementation, it will be understood when to use Interface and abstract class.

**Assignment 5: Exception Handling**

**Problem description:**

**Find the correct output of following code snippets:**

|  |
| --- |
| **1)**  Will the following code compile ?            try              {                 throw new SystemException();             }                         catch(Exception e)             {             }             catch(ArgumentException e)             {             }  **2)**  class MyProgram  {  static void Main(string[] args)  {  int index = 6;  int[] arr = new int[5];  try  {  arr[index] = 100;  }  catch(IndexOutOfRangeException e)  {  Console.WriteLine("Index out of bounds occurred");  }  Console.WriteLine("Program execution continued after Exception Handling");  }  } |

|  |
| --- |
| **3)**  class MyProgram  {  static void Main(string[] args)  {  int index;  int value = 100;  int[] arr = new int[10];  try  {  Console.Write("Enter a number: ");  index = Convert.ToInt32(Console.ReadLine());  arr[index] = value;  }  catch(FormatException e)  {  Console.Write("Bad Format ");  }  catch(IndexOutOfRangeException e)  {  Console.Write("Index out of bounds ");  }  Console.Write("Remaining program ");  }  }  **4)**  class MyProgram  {  public static void Main(string[] args)  {  try  {  int rand = 0;  if (rand == 0)  {  throw new Exception ("Random = 0");  rand = 1;  }  if (rand ==1)  {  Console.WriteLine("Random = 1");  return;  }  Console.WriteLine("Random = 2");  }  finally  {  Console.WriteLine("Flow reaches Finally");  }  }  }  5)  public class A          {         public void methodA()          {             try              {                 B test = new B();                 test.methodB();             }              catch (Exception e)                         {                 throw e;                 Console.WriteLine("Exception from Class A");             }             finally                        {                 Console.WriteLine("Inside Class A Finally");             }         }     }      public class B      {         public void methodB()          {             try              {                 throw new Exception("Exception from Class B");             }              catch (Exception e)              {                 throw;                 Console.WriteLine("Exception from Class A");             }              finally              {                 Console.WriteLine("Inside Class B Finally");             }         }     } |

Note: By finding the output and the correct explanation of the above code snippets, will be able to understand the basics of C# exception handling. In the above snippets , very few types of exceptions are covered. There are a lot more type of exceptions are there i.e.

* System.InvalidCastException – If you try to cast an object to a type that it can’t be cast to
* System.InvalidOperationException – Common generic exception in a lot of libraries

Within these exceptions, some frequently occurred exceptions are:

* System.NullReferenceException – Very common exception related to calling a method on a null object reference
* System.Data.SqlClient.SqlException – Various types of SQL Server exceptions

Explore and implement the other exceptions in the code, specifically focus on the

* Custom Exceptions
* Exception Filters(Introduced in C# 6)

**Assignment 6: Linq**

**Problem description:**

Change the following SQL query to conventional Linq query and corresponding lambda expression:

* *Select \* from Employees order by name desc*
* select age*,* DepartmentId*,* SUM*(*Salary) from Employee group by age*,* DepartmentId
* SELECT p.\*

FROM Purchase p

LEFT OUTER JOIN

Customer c INNER JOIN Address a ON c.AddressID = a.ID

ON p.CustomerID = c.ID

WHERE

(a.State = 'WA' || p.CustomerID IS NULL)

AND p.ID in

(

SELECT PurchaseID FROM PurchaseItem

GROUP BY PurchaseID HAVING SUM (SaleAmount) > 1000

**KeyFocus Area of these assignments:**

* Method overriding
* Delegates
* Exceptional Handling
* Interface and Abstract class
* Linq

---- Thanks----