C# 7.0

Lab Book

Document Revision History

Date	Revision No.	Author	Summary of Changes
10-Jun-2011	1	Ajit Jog	Content Creation
10-Jun-2011	1	Karthikeyan Muthukrishnan	Review
08-Aug-2012	2.0	Ganesh Desai	Content Revamp
12-May-2016	3.0	Nachiket Inamdar	Rectified mistakes and removed Windows Application Development content.
	4.0	Sangeetha C	Incorporated C# 6.0 Features
23-Jan-2019	5.0	Shital Patil	Incorporated C# 7.0 Features

Table of Contents

Lab 1. Working with Data Types and Classes

Objective	This Lab will help you understand 1. How to define Classes 2. Understand how to create DLL's and use them in client application 3. Understand various programming constructs
Time	90 Mins

Note : Use C# 6.0 and 7.0 features, wherever applicable

- While printing the output use String Interpolation
- Use using static
- Use is and as operator

Q1. ABC private Ltd wants to maintain Employee's Information. You need to define an Entity class to hold Employee Information and generate a DLL. You also need to test this class usage by writing a Console application as a client.

Task 1: Define a class called "Employee" with the following fields:

EmployeeId, Employee Name, Address, City, Department, Salary
Define the functions to set the values of each property and to get the value of the Salary in the class: Compile the class to generate a DLL.

Task 2: Create a Console application and use this class. Create an object of this class. Accept the values from the user and assign the members.

Task 3: Modify the console application to define an array of objects to hold 10 records of Employee. Accept the details of 10 employees from the user using a loop. Display the Employee Name and Salary of all the employees.

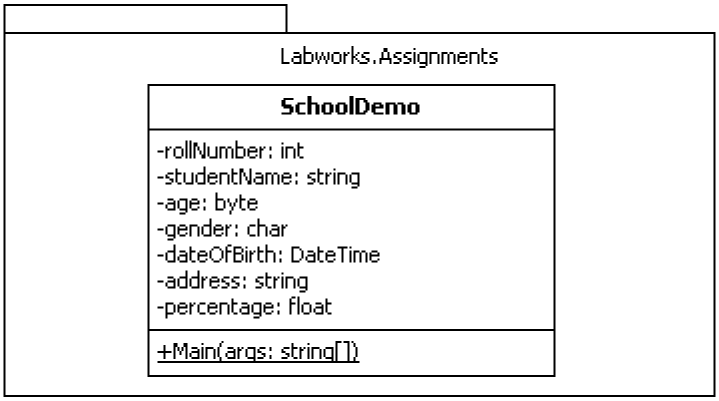
Task 4: Modify the class to add properties using get, set blocks. Modify the console application to use the properties.

Q2. Amit wants to develop an Arithmetic Calculator to perform some arithmetic functions. Develop a program to achieve the same.

Task 1: Create a library project to define a class ArithmeticOperations to perform the operations like Add, Subtract, Multiply, Divide and Modulus on two numbers of integer type, and double type.
Use the class in a console application. Accept the details from the user and perform the operation based on user's choice.

Q3: Create a Console application to test usage of Switch case construct. Accept some integer from user as command line argument and using a switch case construct, check if the value entered is 1, 2, 3, 4 or 5. Print some message in each case. If the value is other than the above values, then print error message.

Q4. St. Joseph school planned to create a system to store the records of students studying in their school. They need to store various kinds of data about their students. Write a C# program based on the class diagram given below and initialize the variables with proper values and print it.



Lab 2. Working with struct data type and arrays

Objective	This Lab will help you understand 1. How to define a structure type and add data and code member 2. How to work with arrays 3. Use Loops to iterate from arrays.
Time	60 Mins

Note : Use C# 6.0 and 7.0 features, wherever applicable

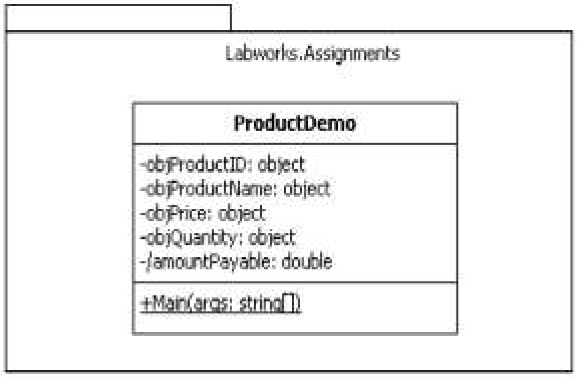
- While printing the output use String Interpolation
- Use using static
- Use is and as operator

1. Define a structure having one member variable as "Number". Add functions to display the square and cube of the number in the same structure. In the main function, Initialize the structure variable, and display the square or the cube based on user's choice.

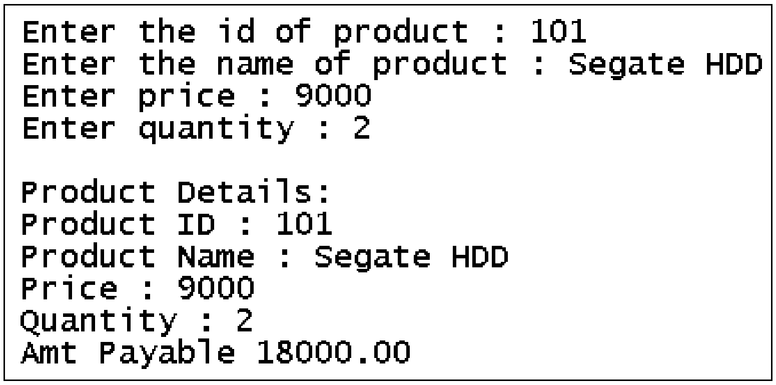
2. Write a program to define a two dimensional array of numbers to store 5 rows and 6 columns. Write a code to accept the data, assign it in array, and print the data entered by the user.

3. Define a single dimension array of strings to hold the name of City. Accept some values from the user and store them in the array. Use foreach loop to print all the data of the array.

4. Create a class named **ProductDemo** which accepts the details of the product, converts the details into reference types using boxing and displays them by converting them into their relevant types using unboxing and calculate the amountPayable. Refer the class diagram given below.



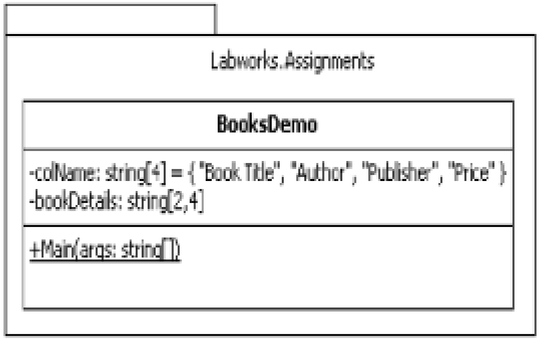
Output



```
Enter the id of product : 101
Enter the name of product : Segate HDD
Enter price : 9000
Enter quantity : 2

Product Details:
Product ID : 101
Product Name : Segate HDD
Price : 9000
Quantity : 2
Amt Payable 18000.00
```

5. Create a Class named **BooksDemo** accepts and displays the details of books using multidimensional array.


Labworks.Assignments
BooksDemo
-colName: string[4] = { "Book Title", "Author", "Publisher", "Price" }
-bookDetails: string[2,4]
+Main(args: string[])

Lab 3. Defining a Class with Constructor, Fields and Properties

Objective	This Lab will help you understand 1. How to use constructors 2. Define Property & Property Value Validation 3. Sub classing and overriding
Time	90 Mins

Note : Use C# 6.0 and 7.0 features, wherever applicable

- While printing the output use String Interpolation
- Use using static
- Use is and as operator

Q1. Corporate university wants to maintain the information about participants scores in various modules. Write a program to store the details of the marks scored in various modules.

Task 1: Create a library project and define a class called Participant. Define the private members and public properties as follows:

EmpId, Name, Company Name, FoundationMarks, WebBasicMarks, DotNetMarks, Total Marks, ObtainedMarks, Percentage

Initialise the Total Marks to 300. ObtainedMarks and Percentage are calculated fields.

Task 2: Add 3 constructors, one Default, one parameterised to initialize the members and one static constructor to initialise the company Name to “Corporate Unniversity”

Task 3: Add the following functions to the class:

a. To calculate Total Marks.
b. To calculate the percentage.
c. To return the percentage

Task 4: Create a console application to accept the data about participants, and create the object. The console application should call the appropriate functions to calculate the Total Marks and Percentage. And then Display the percentage.

Task 5: Modify the class created above to valide the marks. Write the validators in properties. The valid range is 0 to 100. If any invalid value is passed, then assign 0 to that module. Later, you can create Exception classes and raise the exception in case of Invalid data.

Q2. Sanjay has written the following code. The purpose is to create a Bird class and implement function overloading. The code has some errors. Debug the code, and find out what’s wrong with it and correct the code.

```
         private class Bird
        {
                public string Name;
                public double Maxheight;

                public Bird()  //Default Constructor
                {
                        this.Name = Mountain Eagle;
                        this.Maxheight = “500”;

                        //
                        // TODO: Add constructor logic here
                        //
                }

                public Bird(string birdname, double max_ht) //Overloaded Constructor
                {
                        this.Name = “Another Bird”;
                        this.Maxheight = null;
                }

                public void fly()
                {
                        Console.WriteLine(“this.Name is flying at altitude this.Maxheight”);
                }

                public void fly(string AtHeight)
                {
                        if(AtHeight <= this.Maxheight)
                                Console.WriteLine(this.Name + " flying at " + AtHeight.ToString());
                        elseif
                                Console.WriteLine(this.Name  cannot fly at this height);
                }
        }
```

The code in the Main function is as follows:

```
Bird b = new Bird(“Eagle” , double.Parse(“200”));
b.fly();
b.fly(double.Parse(“300”));
```

Q3. You need to write a program to manage the Inventory of the used cars.

Task 1: Create a simple text-based “Console Application” in C# to maintain a catalog of used cars. The catalog keeps track of each car's make, model, year, and sale price.

The program begins with an empty catalog. The program can perform the following operations:

- Adding a new car
- Modify the details of a particular car
- Search for a particular car in the Catalog
- List all the cars in the Catalog
- Delete a car from the Catalog
- Quit

If an unknown command is entered, the user should be informed and asked to enter another command.

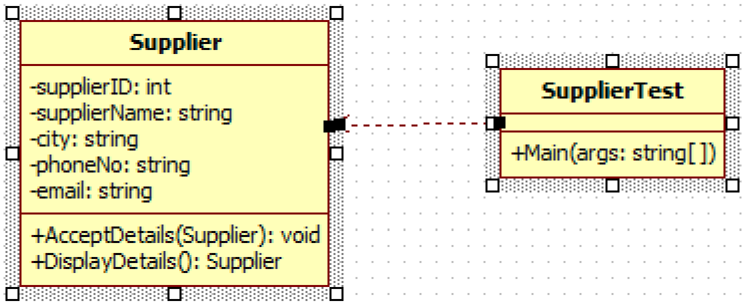
Hint:

1) Create a class called as Car. Create appropriate constructors (Default and Parameterized), Properties for the Car class.

2) Use **Array** to store the Objects of a car.

Once the code is ready, get a peer review done. Maintain the list of issues / bugs identified during the review. You are supposed to fix those issues.

Q4. Create Supplier instance in SupplierTest class, invoke AcceptDetails method to accept the details of the supplier from the user and invoke DisplayDetails method to display the given details of the supplier.



Lab 4. Implementing polymorphism using overriding in C#

Objective	This lab will help you understand 1. Implementing Inheritance and achieving Polymorphism through overriding 2. Understand the c# keywords required to achieve it
Time	90 Mins

Note : Use C# 6.0 and 7.0 features, wherever applicable

- While printing the output use String Interpolation
- Use using static
- Use is and as operator

Q1. You have created Employee class in Lab 1. You need to extend this class and create two derived classes from this class. The derived classes will be ContractEmployee and PermanentEmployee.
The contract Employee class will have Perks as an additional property. The PermanentEmployee will have NoOfLeaves and ProvidendFund Properties.

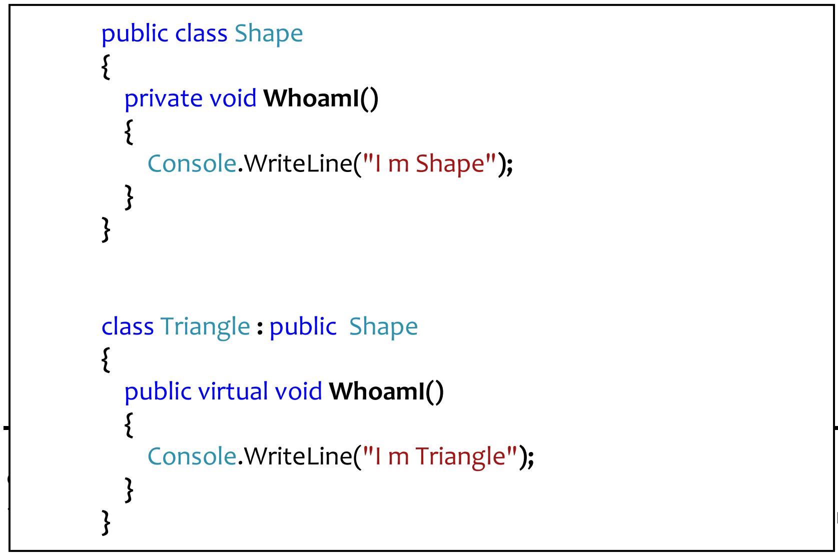
Task 1: Create these two classes by inheriting from the Employee class.

Task 2: Override the GetSalary Method in these two classes. For Contract employee the new salary will be Salary + Perks. For Permanent Employee the new salary will be Salary – Providend Fund.

Task 3: Create a console application to use these classes. Create a Menu driven application to select the Type of employee. Based on the user selection create the object and accept the details from the user.
Also display the salary of the Employee.

Task 4: As we only need to create instance of Contract Employee and Permanent Employee classes, Convert the Employee class to Abstract class. Also make GetSalary method Abstract in the Base class.

Q2. Mahesh has created the following code. The purpose is to create Circle and Triangle class by inheriting the Shape Class. Both the inherited classes should override the WhoamI() method of the Shape Class. The code has some bugs. Identify the Bugs and fix them.



```
public class Shape
{
  private void WhoamI()
  {
    Console.WriteLine("I m Shape");
  }
}


class Triangle : public  Shape
{
  public virtual void WhoamI()
  {
    Console.WriteLine("I m Triangle");
  }
}
```

```
public class Circle : public Shape
{
        void WhoamI()
        {
               Console.WriteLine("I m Circle");
        }
}
class Program
{
         static void Main(string[] args)
        {
             Shape s;
             s = new Triangle();
             s.WhoamI();

             s = new Circle();
             s.WhoamI();

            Console.ReadKey();
        }
}
```

Lab 5. Using Interface, Abstract and Concrete classes concept

Objective	This lab will help you understand 1. The Use of Interface Concept 2. How to use Abstract Class and Concrete Classes 3. Understand how Re-Usability and code sharing can be achieved using abstract base class and overriding concept. 4. Use of "**is**" operator in C#
Time	180 Mins

Note : Use C# 6.0 and 7.0 features, wherever applicable

- While printing the output use String Interpolation
- Use using static
- Use is and as operator

Q.1 Let's build a sample banking program to perform the common tasks like Withdraw and Deposit.

Task 1: Create a class library project and Add a Class called BankAccount. This class needs to implement the IBankAccount Interface.

Task 2: Define a enum as follows. This enum will be used as a property in the interface.

```
//Enum type definition to specify possible set of values
  public enum BankAccountTypeEnum
  {
     Current=1,
     Saving=2
  }
```

Task 3: Define IBankAccount interface and add the following fields to it.

```
double GetBalance();
void Deposit(double amount);
bool Withdraw(double amount);
bool Transfer(IBankAccount toAccount, double amount);
BankAccountTypeEnum AccountType { get; set; }
```

Task 4: Create an abstract class called as BankAccount and implement the class with the interface defined above. Add a property called Balance in this class

```
protected double balance;
```

Task 5: Implement only the Deposit method to increment the Balance. Keep the other two methods abstract in the class.

Task 6: Now let's create concrete classes which are inherited from the BankAccount class.

```
//Concrete Bank Account Classes having their own rules for Minimum Balance
                    class ICICI // Inherit this from BankAccount
                 {

                     Withdraw() // Override this method
                     {
                         // If Balance – amount is >= 0 then only WithDraw is possible.
                         // Write the code to achieve the same.
                     }

                     Transfer() //Override this method
                     {
                        // If Balance – Withdraw is >= 1000 then only transfer can take place.
                          // Write the code to achieve the same.
                     }

                 }

                 class HSBC // Inherit this from BankAccount
                 {
                          Withdraw() //Override this method
                     {
                            // If Balance – amount is >= 5000 then only WithDraw is possible.
                         // Write the code to achieve the same.

                     }

                     Transfer() //Override this method
                     {
                                    // If Balance – Withdraw is >= 5000 then only transfer can
             take place.
                          // Write the code to achieve the same.

                     }
                 }
```

Note: GetBalance() and Deposit() code is shared by both classes via Base Abstract class.

Task 7: Create a console application. The Main() function needs to create the objects and execute the functionality as per the instructions.

```
        Main( ) //Write this function
        {
    // Task to be performed:
           Create a Object of ICICI
           Set the Account type to Saving (Use enum)
           Deposit Rs. 50000 to this account

          Create another Object of ICICI
           Set the Account type to Current (use enum)
           Deposit Rs. 20000 to this account

          Print the Balance of both these account objects.

          Now call the Transfer function to transfer the money from Savings account to Current Account.
    The amount to be transferred is Rs. 5000.

    e.g.      a1.Transfer(a2,5000);


    Now print the Balance after the Transfer from both the accounts.

    Similarly, create two accounts of HSBC Bank. Transfer Rs. 30000 from Saving to Current and display the
    balance.

     }
```

Q2. There is a Change request from the customer. It is as follows:

You need to calculate Interest paid by banks on Saving Account.

Task 1 : Add a function declaration "void CalculateInterest()" in the interface. Define the functions in the concrete classes such as ICICI accounts get 7% interest and HSBC gives 5% interest.

Lab 6. Manage Exceptions

Objective	This lab will help you understand 1. How to handle Exceptions 2. How to Define our own Exception classes and Handle the Exceptions
Time	90 Mins

Note : Use C# 6.0 and 7.0 features, wherever applicable

- While printing the output use String Interpolation
- Use using static
- Use is and as operator

Q1. ABC Corp wants to maintain list of Customers. While accepting the data, you need to validate CreditLimit property. If the value is invalid, you need to raise Exception. We need to implement custom exception class to implement the same.

Task 1: Define a Customer class with following members
CustomerId, Customer Name, Address, City, Phone, CreditLimit

Task 2: Define the properties for all these members.

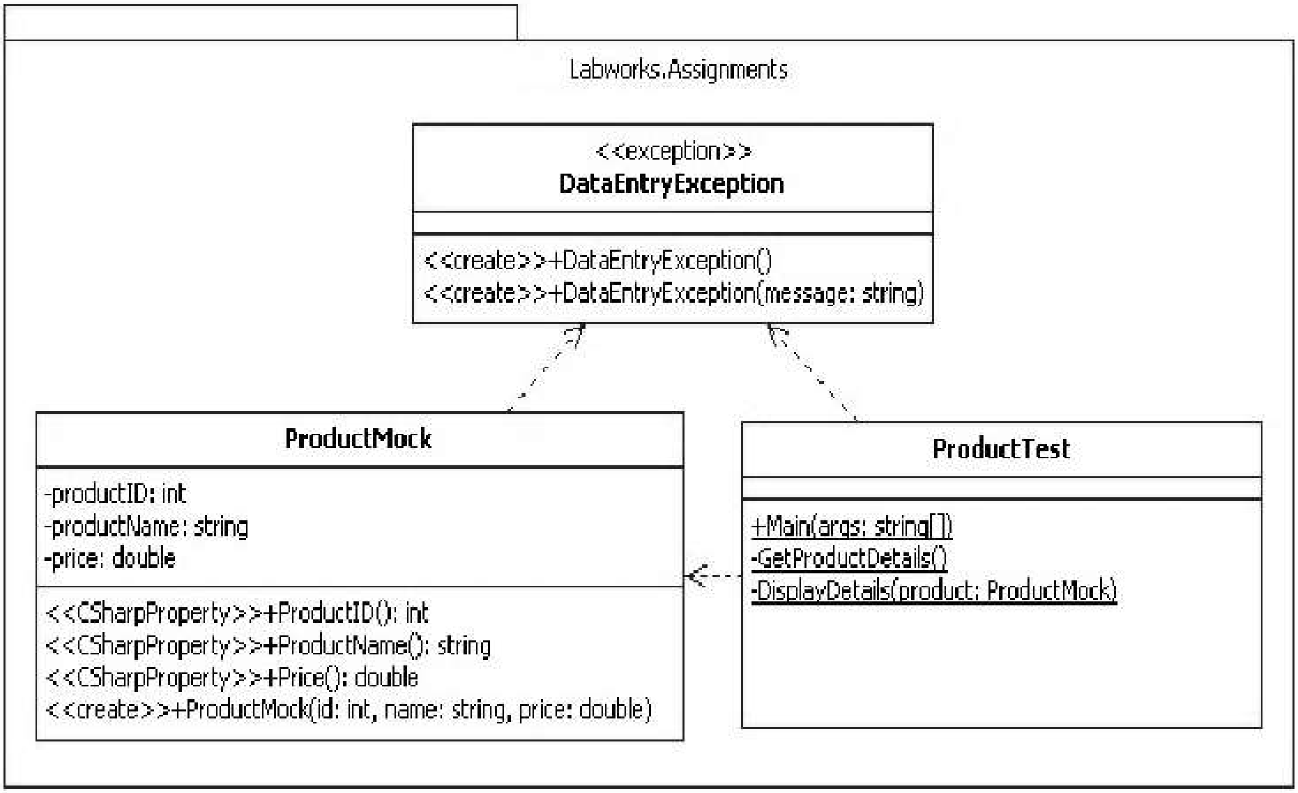
Task 3: Define two constructors (Default and Parameterised) to assign the values.

Task 4: You need to validate the CreditLimit. If the value is above 50000, then you need to raise Exception to handle this. Create InvalidCreditLimit custom Exception class to achieve the same.

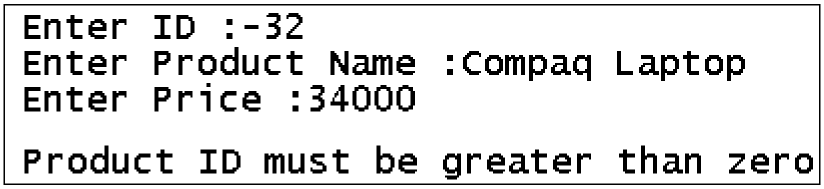
Task 5: Use the Exception class created to throw the exception. Ensure that the Client application catches the exception and handles the error properly.

Q2. Create ProductMock Entity which throws DataEntryException when its properties initialize with following values. Get the Product details from the user and handle in-built and user defined exception. Refer the class diagram given below :

Condition	**Exception Message**
productID <=0	Product ID must be greater than zero
productName = = “”	Product Name cannot be left blank
price <=0	Price of product must be greater than zero.
productName	Product Name should have alphabets and numbers only


Labworks.Assignments
<<exception>>
DataEntryException
<<create>>+DataEntryException()
<<create>>+DataEntryException(message: string)
ProductMock
-productID: int
-productName: string
-price: double
<<CSharpProperty>>+ProductID(): int
<<CSharpProperty>>+ProductName(): string
<<CSharpProperty>>+Price(): double
<<create>>+ProductMock(id: int, name: string, price: double)
ProductTest
+Main(args: string[])
-GetProductDetails()
-DisplayDetails(product: ProductMock)

Output



```
Enter ID :-32
Enter Product Name :Compaq Laptop
Enter Price :34000

Product ID must be greater than zero
```

Lab 7. Using List<> Generic Collection Class

Objective	This lab will help you understand 1. How to use Generic Collection to maintain a list 2. How to perform CRUD operations on the list
Time	90 Mins

Note : Use C# 6.0 and 7.0 features, wherever applicable

- While printing the output use String Interpolation
- Use using static
- Use is and as operator

Q1. You need to maintain a Contact List in a generic List Collection. You need to perform the following tasks:

i. AddContact() – To add contact detail to List
ii. DisplayContact() – To display particular contact detail from List
iii. EditContact() – To modiy particular contact detail from List
iv. ShowAllContacts() – To display all contact details from List

Task 1: Create the Contact class with the following properties:

```
public int ContactNo{get;set;}
public string ContactName{get;set;}
public string CellNo{get;set;}
```

Task 2: Create a Console application and write the Code for the required functionality mentioned above.

Hint:

a. There is a loop in Main() function which accepts the selection option
b. There are additional 4 static functions to perform required tasks which are called based on selection
c. Use the List<Contact> generic collection to maintain the list.

Q2. Create a console application to accept Product Details like ProductNo, Name, Rate and Stock.

[Use Array List Collection]

Display the Menu to perform the following:

================

a. Adding New Product
b. Deleting Currently Searched Product
c. Searching Product

Searching will work as shown below:

- User will enter ProductNo.

- If the product with that productno exists in Collection, then the details should be shown, otherwise show appropriate message.

d. Save the New Product – The products should get saved in the sorted order of ProductNo.

Q3. One of the client has submitted the following request. Manish has developed the code for the same. The requirement is given below. You need to review the code, find out any issues / bugs with the code and correct the same. Also you need to develop a Console based Client application for the same requirement. The Client application should allow Adding new Employee's of specified type, Searching Records, Delete Records and View all records operations.

Problem Statement: XYZ computer Systems PVT Ltd. wants to develop an application to maintain employee details. You have to develop a .NET Application to accept new employee details and store the details in a Collection.

The steps involved in this are given below:

Task 1: Create a private DLL with a class Called Employee. Employee class will have Employee Number, Name and Basic Salary, and PF attributes. Define appropriate properties to access the attributes. Write 2 constructors, one default & one parameterized, to assign the values of the attributes when the object is created.

Task 2: Use List<Employee> collection.

Lab 8. Using Hashtable collection class

Objective	This lab will help you understand 1. How to Use a key-value based collection class. 2. How to perform basic operations on it.
Time	45 Mins

Note : Use C# 6.0 and 7.0 features, wherever applicable

- While printing the output use String Interpolation
- Use using static
- Use is and as operator

Q1. We need to maintain the list of RTO districts. e.g. MH01 – Mumbai, MH 04 – Thane etc. Write a program which uses a Hashtable to maintain this list.

Task 1: Create a console application to maintain the list. It should display a menu to perform the following tasks:

- Add Record in Hashtable
- Search record
- Display All Records
- To display Total count of Records at any point
- Remove any particular record

Q2. Sameer has written a code to create the Hash Table. The code is given below.

```
class Program
 {
          static void Main()
         {
         }

         static Hashtable GetHashtable()
         {
             // Create and return new Hashtable.
             Hashtable hashtable = new Hashtable();
             hashtable.Add("Area", 1000);
             hashtable.Add("Perimeter", 55);
             hashtable.Add("Mortgage", 540);
             return hashtable;
          }
 }
```

You need to perform some tasks on this code. Write the functionality in the main method.

Task 1: See if the Hashtable contains the key “Perimeter”.

Hint: Use ContainsKey and Contains Methods. Is there any difference between them?

Task 2: Print the value of “Area” with indexer.
Task 3: Remove the entry for “Mortgage”

Lab 9. Using Dictionary<> Generic collection class

Objective	This lab will help you understand 1. How to Use a generic key-value based collection class. 2. How to work with various operations on it.
Time	90 Mins

Note : Use C# 6.0 and 7.0 features, wherever applicable

- While printing the output use String Interpolation
- Use using static
- Use is and as operator

Q1. You need to maintain the file extensions along with file types in a dictionary class. Write the code to achieve the same.

Tasks to be performed:

- Create a new dictionary of strings, with string keys.
- Add some elements to the dictionary. There should not be duplicate keys, but some of the values can be duplicates.
- The Add method throws an exception if the new key is already in the dictionary. Test this by adding a duplicate key.
- The indexer can be used to change the value associated with a key. Try changing the value of any record and display the updated value.
- If a key does not exist, setting the indexer for that key adds a new key/value pair. Try this by adding a new value.
- The indexer throws an exception if the requested key is not in the dictionary. Try printing any such key which is not present and handle the exception.
- When you use foreach to enumerate dictionary elements, the elements are retrieved as Key/Value Pair objects. Use a foreach loop to print the values and test this.
- Use the Remove method to remove a key/value pair.

Lab 10. Using Delegate Concept

Objective	This lab will help you understand 1. How to define a Delegate 2. Use Delegate to make calls to function 3. Multicast Delegate concept
Time	45 Mins

Note : Use C# 6.0 and 7.0 features, wherever applicable

- While printing the output use String Interpolation
- Use using static
- Use is and as operator

Q1. You need to perform Arithmetic operations on two numbers. The operations include Add Numbers, Multiply Numbers, Divide Numbers, Subtract Numbers and Find Max Number.

Task 1: Define a class ArithmeticOperation having the above methods.

Task 2: Define a Delegate which can call these methods.

Task 3: Create a console application to accept two numbers and arithmetic operation to be performed from the user. Based on the choice, the Delegate instance will hold the address of the appropriate method.

Task 4: Execute the delegate to get the required result.

Q2. Do a peer review of this code and suggest if any improvements can be done.

Task 5: Mahesh has suggested that we should define one function called PerformArithmeticOperation( ) which will take 2 numbers and the Delegate as parameters. Instead of calling / executing the delegate from Main() function, let this function manage the delegate call. You have to implement this functionality.

Hint: The PerformArithmeticOperation() function signature is:

static void PerformArithmeticOperation(int num1, int num2, MyDelegate arOperation){…}

Lab 11. Using Delegate Concept For Event Model

Objective	This lab will help you understand 1. How to define an event based on delegate 2. Raising a event for notification 3. Handling the event at client code.
Time	90 Mins

Note : Use C# 6.0 and 7.0 features, wherever applicable

- While printing the output use String Interpolation
- Use using static
- Use is and as operator

Q1. ICICI Bank want's to implement SMS alert fascility to the customers whenever they pay the Credit Card Bill. The CreditCard class contains the following fields.

CreditCardNo, CardHolderName, BalanceAmount, CreditLimit.

The class has three functions. GetBalance(), GetCreditLimit(), and MakePayment(). Whenever the MakePayment() method is called, it should update the BalanceAmount and raise an event. The event should send a message. (You don't need to write a logic to send the sms. Only print the message saying amount is credited.)

Task to be performed:

Define a class CreditCard
Define a Delegate to Handle the Event
Define an event which will be raised whenever the payment is made.
Create a Console client application, and try the functionality.

Q2. The FileDownloader utility is developed by a team. The code has some bugs. It is supposed to raise an event, whenever the File is downloaded.

Review the code, find the Bugs and correct the code.

```
delegate void DownloadCompeteHandler(int perc);

    public class FileDownloader
    {
        protected string resourceUrl;
        protected string resourceSavePath;

        event DownloadCompeteHandler DownLoadComplete;

        public FileDownloader(string url,string savepath)
        {
            this.resourceUrl = url;
            this. resourceSavePath = savepath;
        }

        public void DownLoadResource()
        {
            //This is just download simulation place holder code
```

```
                    for (int i = 1; i <= 4; i++)
                    {
                        //Dummy loop to add a delay
                        for (int j = 1; i <= 10000; i++) ;
                            OnDownLoadComplete(i * 25);
                    }

                }

                protected void OnDownLoadComplete( )
                {
                        if (DownLoadComplete == null)
                            DownLoadComplete( );
                }

            }
```

The Main() function code:

```
        public static void Main(string[] args)
        {
            //Instantiate
              FileDownloader                    fd                    =                    new
 FileDownloader("http://www.microsoft.com/vstudio/expressv10.zip",
                                        "d:\\setups");

            //Register Event Handler
            fd.DownLoadComplete += new Handler();

              //Start the task...
             fd.DownLoadResource();

             Console.ReadKey();
          }

          static void fd_DownLoadComplete(int perc)
        {
              Console.SetCursorPosition(10, 10);
              Console.Write("Downloading  {0} Percent Complete", perc);
        }
```

Lab12. Using StreamReader / StreamWriter Classes

Objective	This lab will help you understand 1. How to read a text file using StreamReader instance. 2. How to write in a text file using StreamWriter instance 3. Handling error incase I/O operation fails.
Time	120 Mins

Note : Use C# 6.0 and 7.0 features, wherever applicable

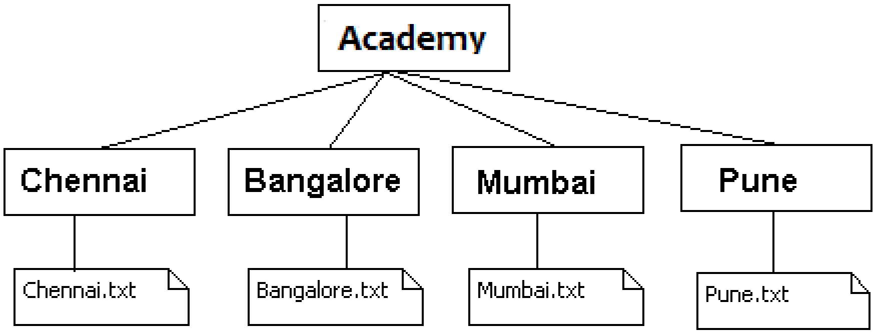
- While printing the output use String Interpolation
- Use using static
- Use is and as operator

Q1. Write a Code to Read and Display the contents of a text file. Accept the name of the file from the user. Handle all the exceptions that might occur during reading.

Q2. Write a Code to perform File Copy operation. You need to accept the source and destination file names. The data should be copied from source file to destination file.
Handle all the exceptions that might occur during the file copy operation.

Extended Assignment

An institute have decided to automate their batch details operations. The application is to be developed in such a manner that a proper directory structure is to be maintained to store the files. The directory structure to be maintained is as shown below:



You need to perform the following operations in C# application

Create a menu based application to store batch details.

- The first option in the menu should allow the user to create a directory structure and the files (if not exists) in the c drive as shown in the above figure.

- The second option should accept the batch details from the user. Based on the location given by the user append the batch details in the respective files.
- The third menu option allows the user to create a backup copy of the Academy folder in D Drive
- The fourth option should allow the user to view the details of the text files "Bangalore.txt", "Chennai.txt", "Mumbai.txt" and "Pune.txt"

Lab 13. Using Serialization to persist business data

Objective	This lab will help you understand 1. How to do binary serialization and de-serialization 2. How to do SOAP serialization and De-serialization 3. The attribute prerequisite for binary serialization
Time	90 Mins

Note : Use C# 6.0 and 7.0 features, wherever applicable

- While printing the output use String Interpolation
- Use using static
- Use is and as operator

Q1. You have already created the Contact class. You need to store List of Contacts in binary format on disk. Perform Binary Serialization to store the List.

To Do:
Write the Program to Accept the data for multiple contacts, store them in a List. Serialize the List using Binary formatter.

Q2. The Client has suggest an Enhancement to the above code. You need to write the code to Deserialize the data from the Binary file, and print the details.

Q3. There is a Change request from the client. Instead of serializing in Binary format, client wants to serialize it using SOAP format. Modify the above program to SOAP serialize the details.

Q4. You have already created the Supplier Class in Lab 3. You need to store the supplied details in XML format. Perform the XML serialization to store the supplied details in XML.

Q5. Create the array of suppliers and store this collection of suppliers in JSON object using JSON Serialization

Q6. Create a Student class with fields RollNo, Name, City, Degree. Implement the ISerializable interface in Student class. And perform binary serialization on the Student Object.

Lab 14. Developing Applications using Layered Architecture

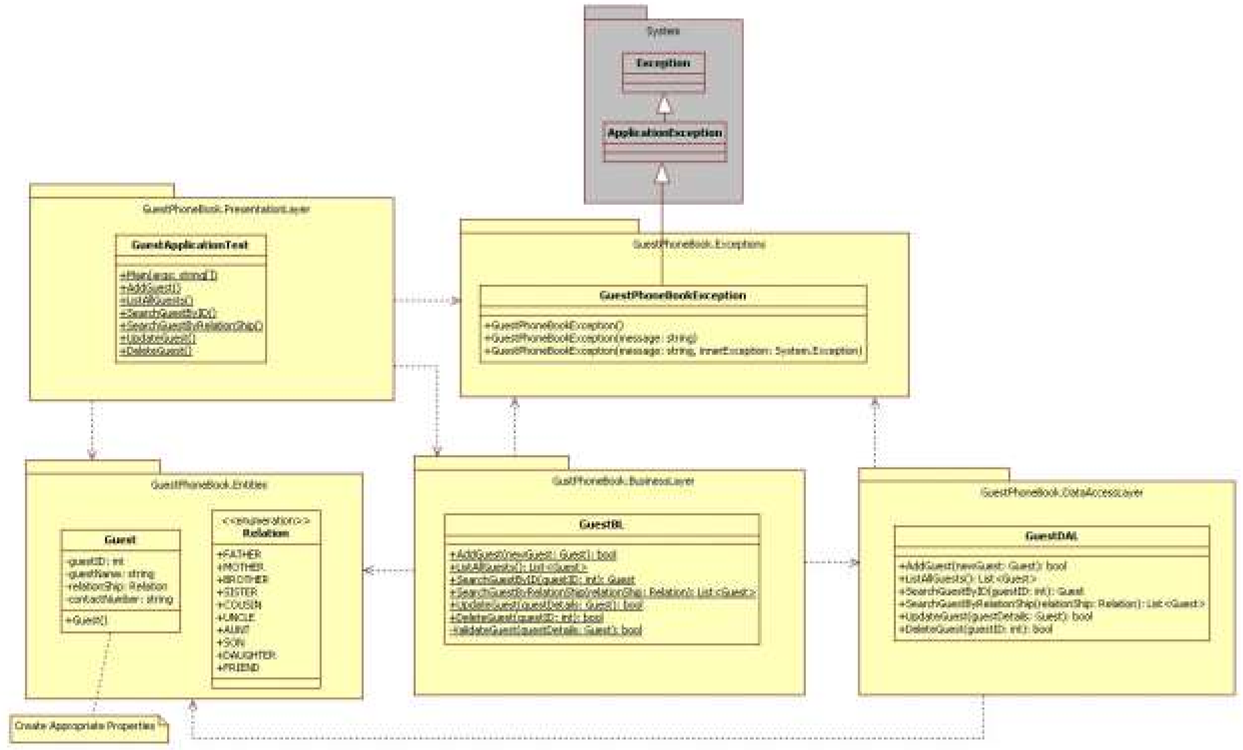
Objective	This lab will help you understand 1. How to develop Layered applications 2. Develop Console and Windows application.
Time	7 Hrs.

Note : Use C# 6.0 and 7.0 features, wherever applicable

- While printing the output use String Interpolation
- Use using static
- Use is and as operator

Q.1 Develop a Console Application to maintain the Contact details of Guests staying at Guest House.

Use Layering concept. The class Diagram is given below. (The faculty will share the copy of the class Diagram)



System
Exception
ApplicationException
GuestPhoneBook.PresentationLayer
GuestApplicationTest
+Main(args: string[])
+AddGuest()
+ListAllGuests()
+SearchGuestByID()
+SearchGuestByRelationShip()
+UpdateGuest()
+DeleteGuest()
GuestPhoneBook.Exceptions
GuestPhoneBookException
+GuestPhoneBookException()
+GuestPhoneBookException(message: string)
+GuestPhoneBookException(message: string, innerException: System.Exception)
GuestPhoneBook.Entities
Guest
-guestID: int
-guestName: string
-relationShip: Relation
-contactNumber: string
+Guest()
<<enumeration>>
Relation
+FATHER
+MOTHER
+BROTHER
+SISTER
+COUSIN
+UNCLE
+AUNT
+SON
+DAUGHTER
+FRIEND
Create Appropriate Properties
GustPhoneBook.BusinessLayer
GuestBL
+AddGuest(newGuest: Guest): bool
+ListAllGuests(): List<Guest>
+SearchGuestByID(guestID: int): Guest
+SearchGuestByRelationShip(relationShip: Relation): List<Guest>
+UpdateGuest(guestDetails: Guest): bool
+DeleteGuest(guestID: int): bool
-ValidateGuest(guestDetails: Guest): bool
GuestPhoneBook.DataAccessLayer
GuestDAL
+AddGuest(newGuest: Guest): bool
+ListAllGuests(): List<Guest>
+SearchGuestByID(guestID: int): Guest
+SearchGuestByRelationShip(relationShip: Relation): List<Guest>
+UpdateGuest(guestDetails: Guest): bool
+DeleteGuest(guestID: int): bool

Validations :

- All fields compulsory
- guestID should be 3 digits long
- guestName can accept alphabets only. It should start with Capital Alphabet and should have minimum 3 characters
- contactNumber should have 10 digits exctly. It should start with 6 or 7 or 8 or 9