Module 4 Practicals

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;

namespace ConsoleApp1
{
    class Program
    {
        enum Days { Sun, Mon, Tue, Wed, Thu, Fri, Sat };
         static void Main(string[] args)
         {


            /* ----------------Part-1-------------------*/

            //enums-An enumeration is a set of named integer constants. An
enumerated type is declared
            //using the enum keyword.C# enumerations are value data type. In
other words, enumeration contains
            //its own values and cannot inherit or cannot pass inheritance.Each of
the symbols in the enumeration
            //list stands for an integer value, one greater than the symbol that
precedes it. By default, the value
            //of the first enumeration symbol is 0.
            int WeekdayStart = (int)Days.Mon;
            int WeekdayEnd = (int)Days.Fri;
            Console.WriteLine("Monday: {0}", WeekdayStart);
            Console.WriteLine("Friday: {0}", WeekdayEnd);

            //events-refer module3-part5 for delegates and events
            Event.display();

            //handling exceptions
            //system exception
            try
            {
                int a = 4;
                int b = 0;
                if (b == 0)
                    throw new Exception("B can't be zero");
                else
                {
                    int ans = a / b;
                    Console.WriteLine(ans);
                }
            }
            catch (DivideByZeroException e)
            {
                Console.WriteLine(e.Message);
            }
            finally
            {
                Console.WriteLine("oops bye!");
            }

            //custom exception
            try
            {
                Age o = new Age();
                o.checkAge(0);
            }
            catch (AgeNotNegative e)
            {
                Console.WriteLine(e.Message);
            }

            Console.ReadKey();

        /* ----------------Part-2-------------------*/

                        //interfaces-An interface is defined as a syntactical contract
that all the classes inheriting
                        //the interface should follow. The interface defines the 'what'
part of the syntactical contract and
                        //the deriving classes define the 'how' part of the syntactical
contract.Interfaces define properties,
                        //methods, and events, which are the members of the interface.
Interfaces contain only the declaration
                        //of the members.It is the responsibility of the deriving class to
define the members.It often helps
                        //in providing a standard structure that the deriving classes
would follow.Abstract classes to some
                        //extent serve the same purpose, however, they are mostly
used when only few methods are to be declared
                        //by the base class and the deriving class implements the
functionalities.Interfaces are declared using
                        //the interface keyword. It is similar to class declaration.
Interface statements are public by default.
                        Transaction t1 = new Transaction("001", "8/10/2012",
78900.00);
                        Transaction t2 = new Transaction("002", "9/10/2012",
451900.00);
```

```
                    t1.showTransaction();
                    t2.showTransaction();

                    //inheritance-single[one base,one child]
                    //              -multi level[one base->child->grand child->....]
                    //              -multiple[two or more base->one child]-not
supported so implemented using interfaces
                    //              -hierarchical[one base,two or more child]
                    Rectangle Rect = new Rectangle();
                    Rect.setWidth(5);
                    Rect.setHeight(7);
                    Console.WriteLine("Total area: {0}", Rect.getArea());


    /* ----------------Part-3------------------
                      */
                    //file i/o
                    String path = @"C:\Users\SMART\Documents\Visual Studio
2019\Projects\file1.txt";


                    if (File.Exists(path))          //check if file exists
                    {
                         Console.WriteLine("File Exists");
                    }
                    else
                    {
                         File.Create(path);           //file creation
                         Console.WriteLine("File created");
                    }

                    String[] lines;                                //read from file
                    lines = File.ReadAllLines(path);
                    Console.WriteLine(lines[0]);
                    Console.WriteLine(lines[1]);

                    String lines1;
                    lines1 = File.ReadAllText(path);
                    Console.WriteLine(lines1);

                    Console.WriteLine("Please enter new content for the file:");
//write to file
                    string newContent = Console.ReadLine();
                    File.WriteAllText(path, newContent);

                    File.AppendAllText(path, newContent);      //append the file

                    String copypath = @"C:\Users\SMART\Documents\Visual
Studio 2019\Projects\file_copy.txt";       //copy the file
                    File.Copy(path, copypath);
```

```
                String movepath = @"C:\Users\SMART\Documents\Visual
Studio 2019\Projects\file_move.txt";
                File.Move(path, movepath);                    //move the file

                File.Delete(copypath);                     //delete the file
                File.Delete(movepath);

                Console.ReadKey();


        }
    }

//class for event
    class Event
    {
        public event NumberChanger MyEvent;
        public Event()
        {
            this.MyEvent = new NumberChanger(this.MultiplyFive);
        }
        public int MultiplyFive(int n) //same signature as of delegate
        {
            return n * 5;
        }
        public static void display()
        {
            Event obj1 = new Event();
            Console.WriteLine(obj1.MyEvent(5));
        }

    }

    //classes for exceptions
    class AgeNotNegative : ApplicationException
    {
        public AgeNotNegative(string s)
            : base(s)
        {

        }
    }
    public class Age
    {
        public void checkAge(int a)
        {
            if (a <= 0)
                throw new AgeNotNegative("Age is negative");
            else
```

```
                Console.WriteLine("Age is:" + a);
        }
}

    //for interface
      public interface ITransactions
      {
            // interface members
            void showTransaction();
            double getAmount();
      }
      public class Transaction : ITransactions
      {
            private string tCode;
            private string date;
            private double amount;

            public Transaction()
            {
                  tCode = " ";
                  date = " ";
                  amount = 0.0;
            }
            public Transaction(string c, string d, double a)
            {
                  tCode = c;
                  date = d;
                  amount = a;
            }
            public double getAmount()
            {
                  return amount;
            }
            public void showTransaction()
            {
                  Console.WriteLine("Transaction: {0}", tCode);
                  Console.WriteLine("Date: {0}", date);
                  Console.WriteLine("Amount: {0}", getAmount());
            }
      }
      //for inheritance
      class Shape
      {
            public void setWidth(int w)
            {
                  width = w;
            }
            public void setHeight(int h)
            {
                  height = h;
```

```
          }
          protected int width;
          protected int height;
     }

     class Rectangle : Shape
     {
          public int getArea()
          {
               return (width * height);
          }
     }

}
```