Name : Gosai Priya

Module 4

Topics:

I. **Enumerations**
II. **Handling Exception**
III. **Events**
IV. **Basic file operations**
V. **Interface & Inheritance**

1. Enumeration:

- Enums are called enumerations.
- It is **a user define** datatype and it is a value type .
- It is like collection of some constant values which is referenced by integer number.
- It is used to assign constant names to a group of numeric integer value.
- In programming there is need to assign some integer number to a particular constant value because we can not remember the whole list, here we can use the concept of the enumeration. it is one type of data structure for all that lists.
- Each symbol stands for an integer value. we can assign any integral data type to the enums. It can be any numeric data type such as byte, sbyte, short, ushort, int, uint, long or ulong.
- By default its index will starts from 0.but we can customize it .we can give any index number we want and the next symbol after that will have the index number **incremented by 1**.
- The default underlying type of the enum is **int**.
- It makes code more readable and maintainable.
- The enum keyword is used for creating enums
- It is a strongly **typed constant** so complier will not do the conversion we have to do it explicitly. we have to do explicit conversion for the enum which value we want to store in another enum although the data under the enum is same we cannot do it directly. we have to do explicit conversion for one enum to assign the value in the other enum.
- It contains static methods like **getnames**() and **getvalues()** which can be used to list enum underlying type values and names in the enums.
- For enums we have class named Enum in the .net library. which has many methods that we can use in the programming.
- **The syntax of the enum is**:

  ```
  Public enum enum_name
  {
  //List of the constants symbol
  ```

}

- We can specify the data type of enum by,
  Public enum enum_name : Data_type
- **We can access the enums by this syntax**:
  Enum.member
- It is define directly under the namespace

2. Handling Exception:

What is Exception?

- An exception is an event or some action, which occurs during the execution of a program and it interrupt the normal flow of execution by terminated the program and showing unwanted error message.
- it occurs due to some program mistakes and it will not execute code after it .
- in general when c# code encounters a situations that it cannot deal with it raises an exception.
- For ex. If we divided the number with zero it will give an divide by zero exception.
- An exception is a c# object that represent an error.

Exception handling:

- That means exception has some predefined class in .net framework and when some exception occurs the following class will throw that exception object and it will catch by the c# catch mechanism .it will instantly stop the execution of the program, show a error message and will not execute the statements after the exception .
- So we don't want our program to terminate and give error message that's why we do exception handling and also excute the statements after the exception occurred.
- The exception handling in c# is one of the powerful mechanism to handle the runtime errors so that normal flow of the application can be maintained.
- For the exception handling done by the programmer it will use the try and catch block statements.
- In the try block we will put thae statements that we think they will might give an exception and in the catch statements we will write the user friendly error message to handle the exception.

- We can pass the reference object of the exception class in the catch block if the exception will occur it will catch by that object and handled.
- That's how we can handle the exception.
- It will not affect the other statements after the exception and also the flow of the execution will be maintained.

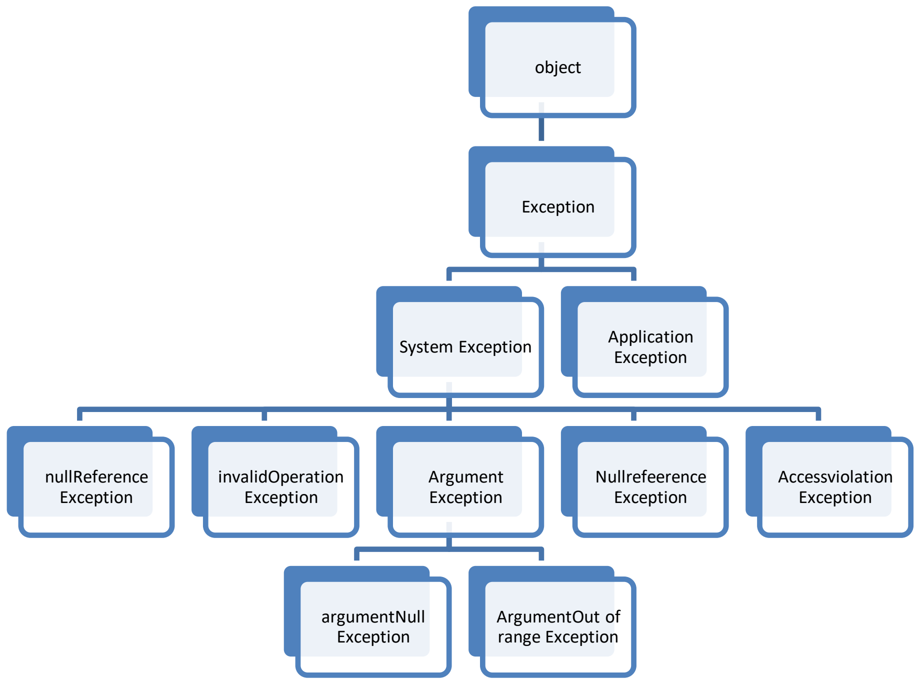
Syntax:

```
Try
{
    //statements to be executed;
}
Catch(reference objject)
{
}
Finally
{
}
```

- Every object of exception has a property named message that will show the message as same as system error message.
    - **Synax:** ex.message
- If we not sure about that which type of exception will be generated we can use multiple catch blocks.
- The finally block is the part of the code that has to be executed irrespective of if the exception was generated or not. it is compulsory part that will execute every time.

Exception classes :

- C# .net includes built-in exception classes for every possible error.
- The **Exception** class is the base class of all the exception classes.
- There is a hierarchy of the exception classes in .NET.



- The SystemException and Application Exception are child class of the Exception class
- The other given exceptions are the child class of the system exception.
    - NullReferenceException
    - InvalidOperationException
    - ArgumentException
    - NullReferenceException
    - AccessViolatingException

- The applicationException class should be derive to create own custom exception class.
- The custom exception class can be created for business rule violations or for other application related errors.

Othe built-inmost occur exception class is given here.

- **DivideByZeroException**- Rasied when an integer value is divided by zero.
- **FileNotFounfException** – Raised when a physical file does not exist at the specified location.

- **FormatException** – raised when a value is not in a appropriate format to be convert from a string by a conversion method such as parse.
- **KeyNotFoundException** - Raised when specified key for accessing a member in a collection is not exist.
- **OverFlowException** - raised when an arithmetic, casting or conversion operation result in the overflow.
- **OutOfMemoryException -** Raised when a program does not get enough memory to execute a code
- **StackOverFlowException -** Raised when a stack memory is overflow.
- **TimeOutException** - Raised when the time interval allotted to an operation has expired.

3. Events:

What is a event?

- Event is generally a user generate action or any system notifications like click ,mouse movement, button press etc.
- Technically event is a notification sent by an object to signal the occurrence of an action.
- It is an encapsulated delegate. it is dependent on the delegate.

How event is raised ?

- The events are declare and raised in the classes.
- The class containing the event is called the publishers class and the other class that accepts this event is calles subscriber class.
- Publisher is an object that contain the definition and the declarion of the events and delegate. A publishers class object invokes the event and it is notified to othr object.
- A subscriber is an object that accepts the event and provides an event handler method. The delegate in the publisher class invoke the method of the subscriber class.
- The publisher will decide when we have to raise the event and the subscriber will determine what response will be given to that event.
- Event can contain many subscribers.

How to declare an event?

- To declare an event we have to use the keyword event.
- First of all we have to declare a delegate for that event.
  Synax: public delegate delegate_type delegate_name(arguments);
- Then we can declare event as shown:
  Event delegate_name event_name;

- Then e have to define the event handler method in the subscriber class and also register the event with the += operator.
- We can invoke the event only from within the class where we declared the event.
- Event does not return any value. it may take parameter as an argument.
- Eventhandler method is define in the subscriber class first it will register for that event and after that event will be handled by the event handler method.
- In windows form we can design such events like button press ,click event etc. we can make custom event handler for that occurred event.

4. Basic file operations:

- Generally, the file is used to store the data.
- The term file handling refers to the various operations like creating file, Reading from the file, appending the file, etc…
- There are two basic operations which is mostly used in file is reading and writing.
- The file becomes stream when we open file for writing and reading.
- Two streams can be formed from file one is input stream which is used to read the file and another is output stream is used to write in the file.
- There is System.IO namespace contains classes which handle input and output streams and provide information about file and directory structure.
- Here is some classes and description of these classes.

Class Type	Description
FileStream	Is used to read from and write within a file.
StreamReader	Is used to read characters from abyte stream.
StreamWriter	Is used to write characters to stream.
BinaryReader	Reads primitive data from binary stream.
BinaryWriter	Writes primitive data in binary format
DirectoryInfo	Used to perform operation on directories.
FileInfo	Use to perform operations on file.

FileStream Class

- We need to create a FileStream object to open existing file or create new file .
- Syntax:
  FileStream <object_name> = new FileSream(<File_name> <FileMode><FileShare>)
- FileMode enum have create, createnew ,open, openorcreate, append,

Truncate.

- FileAccess enum have read write and readwritw.
- FileShare enum have Inheritable, none, read, write and readwrite members.

StreamReader and StreamWriter

- We can the StreamReader and StreamWriter class to read and write data in files.
- We can create an object of StreamWriter and StreamReader by paasing the filename to their constructor.

StreamReader and StreamWriter

- We can the StreamReader and StreamWriter class to read and write data in files.
- We can create an object of StreamWriter and StreamReader by paasing the filename to their constructor.
- There is some methods in StreamWriters class such as close, write, writwLine.
- And also some methods in STreamREader class such as close, Read, ReadLine and ReadToEnd.

BinaryReader and BinaryWriter

- The BinaryReader and BinaryWriter classes read and write data in binary format, rather than text.
- We can use the Write() method of the BinaryWriter class to write binary data in to a file.
- To read a binary data , we can use Read() method of BinaryReader class.

FileInfo Class

- This class provides method to create, copy, delete, move and open files.
- There is somemethods of FileInfo class such as CopyTo, Create,

MoveTo, Delete and properties like Length, Nmae, CreationTime, LastAccessTime, LastWriteTime etc.

DirectoryInfo Class

- The DirectoryInfo class shares almost all of the same properties as the FileInfo class, except that they operate on directories not files.
- There is some methods of DirectoryInfo class such as Create, CreateSubdirectory, MoveTo, Delete, GetDirectories, GetFiles etc. and properties like CreationTime, Fullname, Name, LastAccessTime, LastWriteTime, Root etc.

5. Interface & inheritance:

Inheritance:

- inheritance is one of the fundamental pillar of object oriented programming.
- It is a technique in c# by which one class have all the member and features of another class.

Super class:

- It is the class whose members an features are inherited in the other class is called super class or base class or parent class.

Sub class:

- It is the class which inherited the members and features of the other class .so it is called sub class or child class.
- Inheritance supports the concept of the “reusability” .

When we want to create new class we can have already the features of other class .we can derive new class from the existing class.

By doing this we are reusing the fields and methods of the existing class.

How to use inheritance:

The symbol used for inheritance is “ : “

Syntax:

```
Class classA : ClassB

{

        //methods and functions
}
```

Types of inheritance:

There are some types of inheritance that we can use .

1. Single inheritance
2. Multilevel inheritance
3. Hierarchical inheritance

Hybrid inheritance and multiple inheritance are not supported in c# but we can achive them with interface.

Single inheritance:

- In single inheritance the child class is derived from the one base class.

Multilevel inheritance:

- In multilevel inheritance the child class is derived from the one class and that one class is also dereived from the other base class.
- It is called multilevel inheritance.

Hierarchical inheritance :

- In hierarchical inheritance we can derive many classes from one class .
- One class serves as a superclass for more than one subclass.

Multiple inheritance:

- in multiple inheritance one class can have more than one superclass and inherit features from all the parent classes.
- C# does not support multiple inheritance directly.
- We can achieve multiple inheritance with interface.

Hybrid inheritance:

- It is a mixture of two or more of the above types of inheritance.
- C# doesn’t also support this type of inheritance with the classes.
- It is achieved by the interface.

In inheritance all the members of the derived class is inherited excepts the folloeing :

- Static constructor
- Instance constructors
- Finalizers

Derived classes can also override the inherited members by providing an alternate implementation if the base class maked with the virtual keyword.

Interface:

- It is also one of the impotant concept in c#.
- In the interface any class that implements the interface will implement the interface’s properties, methods and events.
- An interface contains no implementation, it can contain only the declaration of the interface members.
- It can not contain any instance member, constructor, or any static variable.
- The interface members are by default abstract and public we don’t have to declare it explicitly and if we try to do so it will give an error.
- The class who implements the interface will have to implement all the methods of interface.
- Interface cannot have fields.
- An interface is decalred using the keyword interface.

One main reason to introduce the concept of the interface in c# is to achive multiple inheritance.

Implimenting interface:

- An interface is implemented by a class in a way similar to inheriting a class.
- It is implemented by the symbol “ : ”.

- The methods declare in the class should have the same signature as defined in the interface.

- The class should implement all the methods which are there in the interface,if all methods are not implemented the class cannot be complied.

Multiple inheritance by interface:

A class Can implement more than one interface but can not inherit more than one class.

- With the help of interface we can also achieve multiple inheritance. c# directly can not do that that’s why the interface is one of most important concept.
- If we want to do multiple interface there should only one class and the other will be interfaces.
- Note that we an implement more than one interface but we can inherit only one class in multiple inheritance.
- In other words ,a sub class can only have one parent class but a subclass can implements or inherits one or more than one interfaces.
- If the class and interface have the same named method and we have implemented both the class and interface. Then if we want to invoke the class method it will have same way for invoking but if we want the interface method to be invoked we have to explicitly invoke that.
- For that we have create reference variable of that interface and by using that variable we can invoke the method of the interface in the implemented class.

Difference between abstract class and interface

Abstract class	**interface**
It can inherit multiple class and multiple interfaces.	It can inmplement multiple interfaces but cannot inherit multiple class
It can have method with body.	It will contain only abstract methods it can not have method with the body.
Its better option when you need to define some common methods and declare common abstract methods.	Interface is better option when you need to define only abstract methods.
It can declare constructor and destructor.	It can not contain constructor or destructor.
The abstract class method is implemented using override keyword.	There ise no need to use keyword it is implementing withot using override keyword.
Here abstract keyword is mandatory.	Abstract keyword is not mandatory.interfaces are abstract by default.