enum enumeration

- Making code easier to read
- Used for set of values (days of the week, color,….etc)
- Define enum as a data type

```
enum days
        {
            Sat,
            Sun
        }
```

- By default First value =0

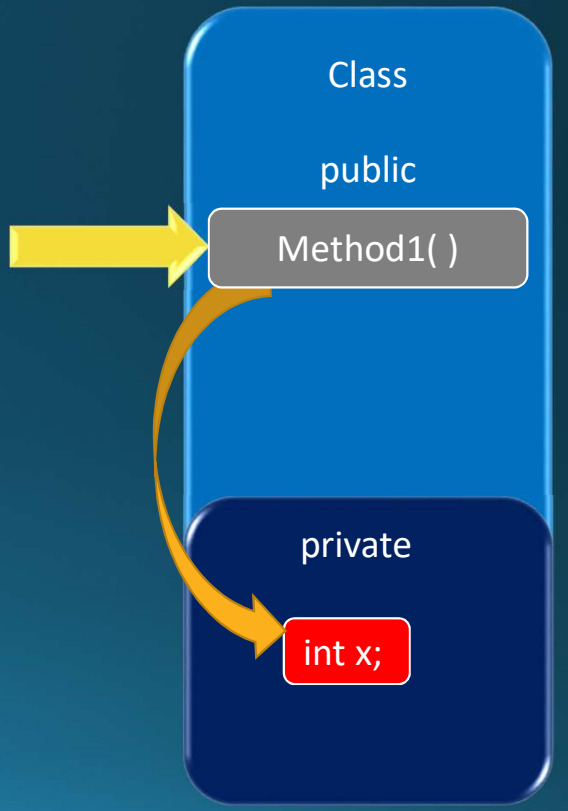
```
enum Day {Sat=1, Sun, Mon, Tue, Wed, Thu, Fri};
```

- Declare and initialize a variable

```
days d = days.Sat;
days d2 =(days) 1; // must use cast
```

Class

- Object Oriented Concept
  - Encapsulation
  - Polymorphism
  - Inheritance
- Class is a reference type



Class (cont.)

- Defining and instantiating a class
  - Declare a class
  - Declare a reference to object
  - Instantiate an object (using ***new*** keyword)

```
employee emp; // declare Reference
emp = new employee();
///or
employee emp = new employee();
```

```
namespace xyz
{
    public class employee
    {
        ….
    }
}
```

Class

- Class Access levels
  - public
  - private
  - Internal (default) (within assembly)
    - intro to class library

```
namespace xyz
{
    public class employee
    {
        ….
    }
}
```

Class

- Class members & their access modifiers
  - Access modifier
    - Public
    - Private
    - Internal
    - Protected
    - Protected internal

```
public class employee
    {
        public int employeeID;
    }
```

Class

- Member variable
- Member method
  - Constructor(s) Polymorphism
    - this keyword
  - Distructor
    - Finalize
    - Dispose
      - Using Statement

```
employee emp = new employee();
```

```
using ( employee emp = new employee() )
{
    //scope of emp variable
}
```

Class(cont.)

- Properties
  - Explicitly
  - Implicitly { set; get; } (Auto implemented)
  - Read only ,write only

```
class employee
{
    private int age;
    public int getAge()
    {
        return age;
    }
    public void setAge(int a)
    {
        age = a;
    }
}
```

```
private int _age;
public int Age
{
    set
    {
        _age = value;
    }
    get
    {
        return _age;
    }
}
```

```
class employee
    {
        public int Age { set; get; }
    }
```

Class(cont.)

- Object Intializer

```
employee em = new employee { ID = 10, Name = "Ahmed", Salary = 1000f };
```

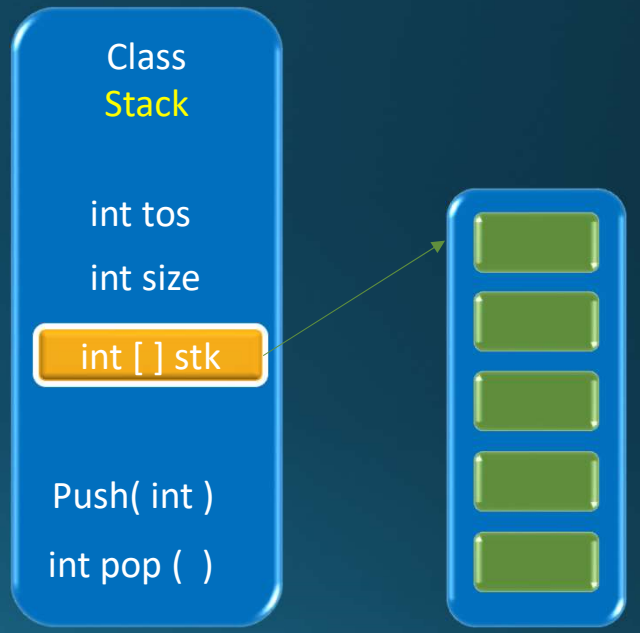
- Array of Object
  - Array of references

```
employee[] emparr;
emparr = new employee[]
{
    new employee(),
    new employee(),
    new employee()
};
```

Assignments

- Design a class represents Employee
  - Name
  - ID
  - Salary
- Adding an array of Employees to menu program
- Design class stack

w8



First in Last out

w8

```
class IndexedNames
  {
    private string[] namelist = new string[size];
    static public int size = 10;
    public IndexedNames()
    {
      for (int i = 0; i < size; i++)
      {
        namelist[i] = "N. A.";
      }
    }

    public string this[int index]
    {
      get
      {
        string tmp;

        if( index >= 0 && index <= size-1 )
        {
          tmp = namelist[index];
        }
        else
        {
          tmp = "";
        }

        return ( tmp );
      }
      set
      {
        if( index >= 0 && index <= size-1 )
        {
          namelist[index] = value;
        }
      }
    }
```

```
public int this[string name]
{
   get
   {
      int index = 0;
      while(index < size)
      {
         if (namelist[index] == name)
         {
          return index;
         }
         index++;
      }
      return index;
   }

}

static void Main(string[] args)
{
   IndexedNames names = new IndexedNames();
   names[0] = "Zara";
   names[1] = "Riz";
   names[2] = "Nuha";
   names[3] = "Asif";
   names[4] = "Davinder";
   names[5] = "Sunil";
   names[6] = "Rubic";

   //using the first indexer with int parameter
   for (int i = 0; i < IndexedNames.size; i++)
   {
      Console.WriteLine(names[i]);
   }

   //using the second indexer with the string parameter
```

```
            Console.WriteLine(names["Nuha"]);
            Console.ReadKey();
        }
    }
}
```

wael, 12/18/2016