# C# Understanding the Object Model

## Structures and Enumerations

---

# I. Overview

This chapter covers two new data types: enumerations and structures. By the end of the unit, you should be able to:

- Declare an enumeration type and create an enumeration variable.
- Declare a structure type and create a structure variable.
- Explain the differences in behavior between a structure and a class.

To see a video demo, watch: https://youtu.be/9M3KBKmfF70

---

# II. Enumerations (Enums)

An enum is a data type that you define. It is used to assign constant text-based names to a group of numeric integer values. Because you are using text-based names, it makes the values easier to read and understand. For example, WeekDays.Sunday is easier to understand than referring to the number 0 as the start of the week.

## A. Creating Enums

Syntax:

```
enum typeName{
value1,
value2,
value3,
valueN
}
```

Where typeName is the name of the enumeration. You will use this name to declare variables the same way you would use any data type. The only difference is the values you can place into the variable must be in the enumeration list.

Note that the list is inside curly braces and each value has a comma after it EXCEPT for the last value in the list

When options are limited to specific values, an enum is ideal because it helps keep the data accurate (you cannot place anything into an enum variable that isn't in the enumeration list)

Example:

```
enum WeekDays
{
Sunday,
Monday,
Tuesday,
Wednesday,
Thursday,
Friday,
Saturday
}
```

By default, the compiler will associate a 0 with the first value in an enumeration list, a 1 with the second value, a 2 with the third value etc.

You can control the integer that is assigned using an assignment statement. You can assign integers to each value OR you can assign an integer to the first value and allow the compiler to increase the value for the rest

Example with the first value initialized:

| Enumeration | Compiler Assigned Integer value |
|---|---|
| enum WeekDays{ | |
| Sunday=1, | 1 |
| Monday, | 2 |
| Tuesday, | 3 |
| Wednesday, | 4 |
| Thursday, | 5 |
| Friday, | 6 |
| Saturday, | 7 |
| Sunday | 8 |
| } | |

Another example with the fifth value initialized:

| Enumeration | Compiler Assigned Integer value |
|---|---|
| enum Departments{ | |
| CIT, | 0 |
| Accounting, | 1 |
| Marketing, | 2 |
| Warehouse, | 3 |
| HR=10, | 10 |
| Maintenance, | 11 |
| Manufacturing, | 12 |
| Sales | 13 |
| } | |

Example with every value initialized:

| Enumeration | Compiler Assigned Integer value |
|---|---|
| enum Departments{ | |
| CIT=1, | 1 |

| | |
|---|---|
| Accounting=5, | 5 |
| Marketing=10, | 10 |
| Warehouse=20, | 20 |
| HR=25, | 25 |
| Maintenance=30, | 30 |
| Manufacturing=40, | 40 |
| Sales=50 | 50 |
| } | |

By default, enumeration lists use integer values. You can change the data type to any whole number data type by using a typecast (the only reason to do this would be to conserve memory)

Example:

```
enum Departments: short{
CIT,
Accounting,
Marketing,
Warehouse,
HR=10,
Maintenance,
Manufacturing,
Sales
}
```

## B. Using Enum variables

Anytime you want to access an enum, whether it is to assign it to a variable or display it's contents, you need to use the enum data type name you created, a dot and a constant value from the list.

1. Creating a variable

```
enum Departments: short{
CIT,
Accounting,
Marketing,
Warehouse,
HR,
Maintenance,
Manufacturing,
Sales
}

Departments myDept = Departments.CIT;
```

You can create enum members in classes, structures, methods etc.

If you want to print the enum constant, you can print the variable it was assigned to OR the enumType.value

Examples:

```
Console.WriteLine(Departments.CIT);
or
Console.WriteLine(myDept);
```

If you want to print the integer instead of the value, you can typecast the value or variable storing the value

Examples:

Console.WriteLine((int)Departments.CIT);
or
Console.WriteLine((int)myDept);

If you aren't sure if the variable will be used, ou can create a nullable version:

Departments? myDept = null;

**Programming Example (you will notice that the results are the same when using GetValueOrDefault() which is used with nullable values)**

```
using System;
namespace movies
{
    class Program
    {
        enum Genre
        {
            Action,
            Horror,
            SciFi,
            RomCom,
            Drama
        }
        static void Main(string[] args)
        {
            Genre? movieGenre;
            Console.WriteLine("What genre of movie do you like?");
            Console.WriteLine("A - action\nH - horror\nS - SciFi\nR - RomCom\nD - Drama");
            char g = char.Parse(Console.ReadLine());
            switch(g)
            {
                case 'A':
                    movieGenre= Genre.Action;
                    break;
                case 'H':
                    movieGenre = Genre.Action;
                    break;
                case 'S':
                    movieGenre = Genre.SciFi;
                    break;
                case 'R':
                    movieGenre= Genre.RomCom;
                    break;
                case 'D':
                    movieGenre= Genre.Drama;
                    break;
                default:
                    movieGenre = null;
                    break;
            }
            Console.WriteLine($"You picked {movieGenre} as your favorite.");
            Console.WriteLine($"The genre you picked is number {(int)movieGenre} on the list.");
            // Using GetValueOrDefault has the same results
            Console.WriteLine($"You picked {movieGenre.GetValueOrDefault()} as your favorite.");
            Console.WriteLine($"The genre you picked is number {(int)movieGenre.GetValueOrDefault()} on the list.");
        }
    }
}
```

**Sample Output:**

What genre of movie do you like?
A - action
H - horror
S - SciFi
R - RomCom
D - Drama
R
You picked RomCom as your favorite.
The genre you picked is number 3 on the list.

You picked RomCom as your favorite.
The genre you picked is number 3 on the list.

---

# III. Structures

A structure is a composite data type similar to a class, BUT it is stored in the stack instead of the heap, so it is a value type.

Structures can contain fields, methods, constants, parameterized constructors, properties, indexers, operators and even other structure types.

## A. Creating a Structure

To create a structure, you need to use the keyword struct followed by the name of the structure, curly braces and the fields, methods, properties etc that you want included in the structure.

Syntax:

```
 AccessModifier struct  StructName {
   // member variables
         AccessModifier dataType variable1;
         AccessModifier dataType variable2;
         ...
         AccessModifier dataType variableN;
   // member methods
         AccessModifier returnDataType method1(parameter_list) {
            // method body
         }
         AccessModifier returnDataType method2(parameter_list) {
            // method body
         }
         ...
         AccessModifier returnDataType methodN(parameter_list) {
            // method body
         }
}
```

Example:

```
        struct Movie
        {
            private string Title;
            private string Description;
            private string Rating;
            private Genre? Genre;

            // parameterized constructor
            public Movie(string title, string description, string rating, Genre genre)
            {
                Title = title;
                Description = description;
                Rating = rating;
                Genre = genre;
            }
            // method that does the same thing as a parameterized constructor
            public void setValues(string title, string description, string rating, Genre genre)
            {
                Title = title;
                Description = description;
                Rating = rating;
                Genre = genre;
            }
            public string Display()
            {
                return "Title: " + Title + "\nDescription: " + Description +
                    "\nRating: " + Rating + "\nGenre: " + Genre;
            }
```

```
        }
```

While structures may look like classes, they are not the same thing.

A few key concepts to remember include:

1. The compiler always creates a constructor to intialize values, so you cannot create your own.
2. You can initialize values using another method name or a parameterized constructor
3. You cannot initialize fields by setting them equal to a value such as private string Title="";
4. Structures cannot be used as a base class for inheritance
5. You can create nullable variables in structures. private Genre? Genre; can be included in a structure.

The table below summarizes the differences:

| Question | Structure | Class |
|---|---|---|
| Is this a value type or a reference type? | A structure is a value type. | A class is a reference type. |
| Do instances live on the stack or the heap? | Structure instances are called *values* and usually live on the stack (see the following note). | Class instances are called *objects* and usually live on the heap. |
| Can you declare a default constructor? | No | Yes |
| If you declare your own constructor, will the compiler still generate the default constructor? | Yes | No |
| If you don't initialize a field in your own constructor, will the compiler automatically initialize it for you? | No | Yes |
| Are you allowed to initialize instance fields at their point of declaration? | No | Yes |

## B. Using a Structure

To create a structure variable, you use similar syntax to a class:

Syntax: StructureName variableName = new StructureName();

Example: Movie movie = new Movie();

When you use the new keyword followed by the structure name, the default constructor is called. Strings are set to blank, numbers are set to zero. If you do this, you can create a method to fill in values, but that method will not be a constructor.

Programming Example: The example lets the compiler initialize the values. It uses the setValues method to change values and the Display method to format a print string.

```
using System;
namespace movies
{
    class Program
    {
        enum Genre
        {
            Action,
            Horror,
            SciFi,
            RomCom,
```

```
            Drama
        }
        struct Movie
        {
            private string Title;
            private string Description;
            private string Rating;
            private Genre? Genre;
            public Movie(string title, string description, string rating, Genre genre)
            {
                Title = title;
                Description = description;
                Rating = rating;
                Genre = genre;
            }
            public void setValues(string title, string description, string rating, Genre genre)
            {
                Title = title;
                Description = description;
                Rating = rating;
                Genre = genre;
            }
            public string Display()
            {
                return "Title: " + Title + "\nDescription: " + Description +
                    "\nRating: " + Rating + "\nGenre: " + Genre;
            }
        }
        static void Main(string[] args)
        {
            Movie movie = new Movie();
            Console.WriteLine("What is the title of your favorite movie?");
            string tempTitle=Console.ReadLine();
            Console.WriteLine("What is the plot?");
            string tempDescription=Console.ReadLine();
            Console.WriteLine("What is the rating?");
            string tempRating=Console.ReadLine();
            Console.WriteLine("What is the genre?");
            Console.WriteLine("A - action\nH - horror\nS - SciFi\nR - RomCom\nD - Drama");
            Genre tempGenre=Genre.Action;
            char g = char.Parse(Console.ReadLine());
            switch(g)
            {
                case 'A':
                    tempGenre = Genre.Action;
                    break;
                case 'H':
                    tempGenre = Genre.Action;
                    break;
                case 'S':
                    tempGenre = Genre.SciFi;
                    break;
                case 'R':
                    tempGenre = Genre.RomCom;
                    break;
                case 'D':
                    tempGenre = Genre.Drama;
                    break;
            }
            movie.setValues(tempTitle,tempDescription,tempRating,tempGenre);
            Console.WriteLine($"{movie.Display()}");
        }
    }
}
```

Sample Output:

What is the title of your favorite movie?
Harry Potter
What is the plot?

Young boy becomes a wizard
What is the rating?
PG
What is the genre?
A - action
H - horror
S - SciFi
R - RomCom
D - Drama
S
Title: Harry Potter
Description: Young boy becomes a wizard
Rating: PG
Genre: SciFi

If you want to create a use a parameterized constructor, then you will create the structure variable WHEN you call the constructor

Programming example that uses a parameterized constructor:

```
using System;
namespace movies
{
    class Program
    {
        enum Genre
        {
            Action,
            Horror,
            SciFi,
            RomCom,
            Drama
        }
        struct Movie
        {
            private string Title;
            private string Description;
            private string Rating;
            private Genre? Genre;

            public Movie(string title, string description, string rating, Genre genre)
            {
                Title = title;
                Description = description;
                Rating = rating;
                Genre = genre;
            }
            public string Display()
            {
                return "Title: " + Title + "\nDescription: " + Description +
                    "\nRating: " + Rating + "\nGenre: " + Genre;
            }
        }
        static void Main(string[] args)
        {
            Console.WriteLine("What is the title of your favorite movie?");
            string tempTitle=Console.ReadLine();
            Console.WriteLine("What is the plot?");
            string tempDescription=Console.ReadLine();
            Console.WriteLine("What is the rating?");
            string tempRating=Console.ReadLine();
            Console.WriteLine("What is the genre?");
            Console.WriteLine("A - action\nH - horror\nS - SciFi\nR - RomCom\nD - Drama");
            Genre tempGenre=Genre.Action;
            char g = char.Parse(Console.ReadLine());
            switch(g)
            {
                case 'A':
                    tempGenre = Genre.Action;
                    break;
                case 'H':
                    tempGenre = Genre.Action;
```

```
                    break;
                case 'S':
                    tempGenre = Genre.SciFi;
                    break;
                case 'R':
                    tempGenre = Genre.RomCom;
                    break;
                case 'D':
                    tempGenre = Genre.Drama;
                    break;
            }
            Movie movie = new Movie(tempTitle,tempDescription,tempRating,tempGenre);
            Console.WriteLine($"{movie.Display()}");
        }
    }
}
```

## C. Copying a structure

When you copy a structure, because it is a value type, you will end up with 2 sets of data in 2 different stack memory locations. This is one BIG difference from copying classes which are reference types (with classes there is only 1 set of data and you end up with 2 objects using the same set of data). Changes to one set does not affect the other

Programming Example: This example includes a parameterized constructor, set methods and a display method. The first movie structure will use the parameterized consttructor. It will be copied into a newMovie structure. 2 of the fields will be changed using set methods. The newMovie will print out showing that the 2 fields changed for that structure only (the first one did not change.

```
using System;
namespace movies
{
    class Program
    {
        enum Genre
        {
            Action,
            Horror,
            SciFi,
            RomCom,
            Drama
        }
        struct Movie
        {
            private string Title;
            private string Description;
            private string Rating;
            private Genre? Genre;

            public Movie(string title, string description, string rating, Genre genre)
            {
                Title = title;
                Description = description;
                Rating = rating;
                Genre = genre;
            }
            public void setTitle(string title)
            {
                Title = title;
            }
            public void setDescription(string description)
            {
                Description = description;
            }
            public void setRating(string rating)
            {
                Rating = rating;
            }
            public void setGenre(Genre genre)
```

```
            {
                Genre = genre;
            }
            public string Display()
            {
                return "Title: " + Title + "\nDescription: " + Description +
                    "\nRating: " + Rating + "\nGenre: " + Genre;
            }
        }
        static void Main(string[] args)
        {
            //Movie movie=new Movie();
            Console.WriteLine("What is the title of your favorite movie?");
            string tempTitle=Console.ReadLine();
            Console.WriteLine("What is the plot?");
            string tempDescription=Console.ReadLine();
            Console.WriteLine("What is the rating?");
            string tempRating=Console.ReadLine();
            Console.WriteLine("What is the genre?");
            Console.WriteLine("A - action\nH - horror\nS - SciFi\nR - RomCom\nD - Drama");
            Genre tempGenre=Genre.Action;
            char g = char.Parse(Console.ReadLine());
            switch(g)
            {
                case 'A':
                    tempGenre = Genre.Action;
                    break;
                case 'H':
                    tempGenre = Genre.Action;
                    break;
                case 'S':
                    tempGenre = Genre.SciFi;
                    break;
                case 'R':
                    tempGenre = Genre.RomCom;
                    break;
                case 'D':
                    tempGenre = Genre.Drama;
                    break;
            }
            // Creating the first movie structure variable
            Movie movie = new Movie(tempTitle,tempDescription,tempRating,tempGenre);
            // Copying the structure variable
            Movie newMovie = movie;
            //changing values in the new structure variable
            newMovie.setRating("PG13");
            newMovie.setTitle("Harry Potter Deathly Hallows");
            //Printing the new movie
            Console.WriteLine("Here's movie #2");
            Console.WriteLine($"{newMovie.Display()}");
            Console.WriteLine();
            //Printing the original
            Console.WriteLine("Here's movie #1");
            Console.WriteLine($"{movie.Display()}");
        }
    }
}
```

Sample Output

```
What is the title of your favorite movie?
Harry Potter
What is the plot?
Young boy becomes a wizard
What is the rating?
PG
What is the genre?
A - action
H - horror
```

```
S - SciFi
R - RomCom
D - Drama
S
Here's movie #2
Title: Harry Potter Deathly Hallows
Description: Young boy becomes a wizard
Rating: PG13
Genre: SciFi

Here's movie #1
Title: Harry Potter
Description: Young boy becomes a wizard
Rating: PG
Genre: SciFi
```

---