**Topic 2: Enumerated Types**

Top of Form

Bottom of Form

**Content**

**Reading**

[Reading](https://dmacc.blackboard.com/webapps/blackboard/content/listContent.jsp?course_id=_102593_1&content_id=_7286810_1)

Enumerated types

* + Microsoft:docs.microsoft.com
    - [Enumeration types](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/builtin-types/enum)
  + GeeksForGeeks: geeksforgeeks.com
    - [Enumeration](https://www.geeksforgeeks.org/c-sharp-enumeration-or-enum/)
  + TutorialsPoint: tutorialspoint.com
    - [Enums](https://www.tutorialspoint.com/csharp/pdf/csharp_enums.pdf)
  + W3Schools: w3schools.com
    - [Enums](https://www.w3schools.com/cs/cs_enums.asp)

**Enumerated Types**

[Enumerated Types](https://dmacc.blackboard.com/webapps/blackboard/content/listContent.jsp?course_id=_102593_1&content_id=_7286810_1)

You learned that you can use a switch to convert days to integers and also integer to days. For the latter, there is another option to represent a set of values numerical. This is call enumeration.

enum Day

{

Monday = 1,

Tuesday = 2,

Wednesday = 3,

Thursday = 4,

Friday = 5,

Saturday = 6,

Sunday = 7

}

Day workDay;

workDay = Day.Wednesday;

Cool, but how would you use that? Read some of the links in Reading to find out more uses.

NOTE: No semi-colon after the enum declaration.

**Enumeration as bit flags**

[Enumeration as bit flags](https://dmacc.blackboard.com/webapps/blackboard/content/listContent.jsp?course_id=_102593_1&content_id=_7286810_1)

Your enumeration could also be a combination of choices, such as setting a workout schedule and wanting to jog several days a week.

Then use bitwise logical operators OR (|) or AND (&) to combine choices.

For example from [docs.microsoft.com](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/builtin-types/enum)

[Flags]

public enum Days

{

None = 0b\_0000\_0000, // 0

Monday = 0b\_0000\_0001, // 1

Tuesday = 0b\_0000\_0010, // 2

Wednesday = 0b\_0000\_0100, // 4

Thursday = 0b\_0000\_1000, // 8

Friday = 0b\_0001\_0000, // 16

Saturday = 0b\_0010\_0000, // 32

Sunday = 0b\_0100\_0000, // 64

Weekend = Saturday | Sunday

}

public class FlagsEnumExample

public static void Main()

{

Days meetingDays = Days.Monday | Days.Wednesday | Days.Friday;

Console.WriteLine(meetingDays);

// Output:

// Monday, Wednesday, Friday

Days workingFromHomeDays = Days.Thursday | Days.Friday;

Console.WriteLine($"Join a meeting by phone on {meetingDays & workingFromHomeDays}");

// Output:

// Join a meeting by phone on Friday

bool isMeetingOnTuesday = (meetingDays & Days.Tuesday) == Days.Tuesday;

Console.WriteLine($"Is there a meeting on Tuesday: {isMeetingOnTuesday}");

// Output:

// Is there a meeting on Tuesday: False

var a = (Days)37;

Console.WriteLine(a);

// Output:

// Monday, Wednesday, Saturday

}

}

**Enumeration and Unit Testing**

[Enumeration and Unit Testing](https://dmacc.blackboard.com/webapps/blackboard/content/listContent.jsp?course_id=_102593_1&content_id=_7286810_1)

https://youtu.be/j7aDu\_d8OgU

**Enumeration and Unit Testing Practice**

[Enumeration and Unit Testing Practice](https://dmacc.blackboard.com/webapps/blackboard/content/listContent.jsp?course_id=_102593_1&content_id=_7286810_1)

You are planning to plant a garden. You have room for 6 rows of flowers and you want to print watering instructions.

Graphical user interface, text, application, email

Description automatically generated

Write a Console App solution named Module7 with project EnumerationTypes (you may need to rename it from Module7)



Your project enumerates the six flowers (tulip, carnation, rose, peony, petunia, and daisy) and the days of the week. Call the enumeration type Flower and WaterDays, respectively.

Write a method GardenPlan() that accepts the Flower enum variable as a parameter and uses case/switch statement to create a local string variable you will return for printing in the main method. This return value will assign the watering scheduling. You will use WaterDays enum in the method to build the string. The method then returns a concatenated string of the flower name + "needs watering"  + watering days. See output below, this also gives you the watering schedule to code.

|  |
| --- |
| Excepted Output: |
| A carnation needs watering on the Weekend  A peony needs watering on Wednesday, Saturday  A petunia needs watering on Monday, Friday  A daisy needs watering on Saturday  A rose needs watering on Tuesday, Friday  A tulip needs watering on Monday, Thursday |

In the Main() method, display each of the 6 rows by calling method GardenPlan() and printing to the console. There is no need to declare another variable, you can use the enum Flower.

Add an xUnitTest project to your Module7 solution. Copy from or download [UnitTest1.cs](https://dmacc.blackboard.com/bbcswebdav/pid-7286830-dt-content-rid-101466167_1/xid-101466167_1) and add the file to your xUnitTest project. Do not change any tests or test names. Don't forget to add the dependency and make your Program visibile to unit tests (public class Program). This will test your method. Replace the not implemented exception in the UnitTest1.cs with appropriate Unit Tests.

Give it a try before looking at a possible solution: [Program.cs](https://dmacc.blackboard.com/bbcswebdav/pid-7286830-dt-content-rid-101466168_1/xid-101466168_1)

[**Enumeration Quiz**](https://dmacc.blackboard.com/webapps/blackboard/content/launchAssessment.jsp?course_id=_102593_1&content_id=_7286832_1&mode=view)

This quiz covers enumration. It is not timed, you have 2 attempts