# Creating a Date class

### Due: Demonstration due at the end of the class

We will assume that there are 12 months in a year and each month is 30days long.

|  |
| --- |
| **Date**  Class |
| **Fields**  - year : int  - month : int  - day : int |
| **Methods**  + «constructor» Date(year : int, month : int, day : int)  + ToString() : string  + Add(days : int) : void  + Add(month : int, days : int) : void  + Add(other : Date) : void  - Normalize() : void |

## Description of field members:

1. **year**: this private member represents the year value of this object.
2. **month**: this private member represents the month value of this object. You may interpret the months as you normally do i.e. 1 = January, 2 = February

Alternately you may choose to have 0 = January, 1 = February

1. **day**: this private member represents the day value of this object.

## Description of constructor:

1. **public Date(int year, int month, int day)**: this is the constructor of this class. It takes three integer arguments and assigns them to the appropriate fields. You will assume that all the arguments will be in the correct range.

## Description of action members:

1. **public override string ToString()**: this method overrides the same method in the base class. It does not take any argument but it returns a string representing this object.
2. **public void Add(int days)**: this method takes a single integer argument representing the amount to increase the day field by and adds it to the appropriate field.
3. **public void Add(int months, int days)**: this method takes two integer arguments representing the amount to increase the month field and the day field. It adds the argument to the appropriate fields.

Having multiple methods with the same name is called method-overloading.

It simplifies the design process

1. **public void Add(**Date **other)**: this method takes a single integer argument representing a date. The is actually three values (year, month and day) packaged into a single date object. You will have to separate each packed value and increase the year, month and day fields by the appropriate amount.
2. **private void Normalize()**: this method does not take any argument nor does it return a value. It does the following:

This member is private because it is not meant to be used outside the class.

* 1. If the days field is more than the number of days in a month, then it subtracts the number of days in the month from the day field and increases the month field by one.
  2. If the month field is more than 12 then, 12 is subtracted from the month field and the year field is increased by 1.

This method should be called at the end of each Add method.

## In your main method write the code to do the following:

1. Create a Date object. You decide on the arguments.
2. Display the above object reference on the console.
3. Call the **AddDays()** method of the object reference to verify that the method works correctly. (How would you display the modified object?).
4. What would you add to the class to make the display more sensible? i.e. the day and month value should be in range

## Enhancements:

1. Display the text for the month i.e. Jan or January for the month 1 or 0 depending on your implementation.
2. Handle the different month length properly   
   "**30 days** has September, April, June, and November. All the rest have **31**. Save February, with **28 days** clear, and **29** each leap year."