## What is immutability and how can you make your class immutable?

In object-oriented and functional programming, an immutable object (unchangeable object**) is an object whose state cannot be modified after it is created.** This is in contrast to a mutable object (changeable object), which can be modified after it is created. In some cases, an object is considered immutable even if some internally used attributes change but the object's state appears to be unchanging from an external point of view.

Strings and other concrete objects are typically expressed as immutable objects to improve readability and run time efficiency in object-oriented programming. Immutable objects are also useful because they are inherently thread-safe. Other benefits are that they are simpler to understand and reason about and offer higher security than mutable objects.
**To define a simple immutable class follow the below mentioned rules**

1. Don't provide "**set** " properties — methods that modify fields or objects referred to by fields.
2. Make all fields **readonly** and **private**.
3. Don't allow subclasses to **override** methods. The simplest way to do this is to declare the class as **sealed**. A more sophisticated approach is to make the constructor **private** and construct instances in factory methods.
4. **If the instance fields include references to mutable objects, don't allow those objects to be changed**:
5. Don't provide methods that modify the mutable objects.
6. **Don't share references to the mutable objects**. Never store references to external, mutable objects passed to the constructor; if necessary, **create copies, and store references to the copies**. Similarly, create copies of your internal mutable objects when necessary to avoid returning the originals in your methods.

```
class Date
{
    public int Year { get; set; }
    public int Month { get; set; }
    public int Day { get; set; }
}
sealed class MyEmployee
{
    private readonly String name;
    private readonly decimal salary;
    private readonly Date dateOfBirth;
    public MyEmployee(String name, decimal salary, Date dateOfBirth)
    {
        this.name = name != null ? name : "default name";
        this.salary = salary > 0 ? salary : 1m;
        this.dateOfBirth = dateOfBirth != null ?
                     new Date() {Year  = dateOfBirth.Year,
                                 Month = dateOfBirth.Month,
                                 Day   = dateOfBirth.Day }
                    : new Date() { Year = 2000, Month = 1, Day = 1 };
    }

    public String Name
    {
        get { return name; }
    }
    public decimal Salary
    {
        get { return salary; }
    }
    public Date DateOfBirth
    {
        get { return new  Date() { Year = dateOfBirth.Year,
                                   Month = dateOfBirth.Month,
                                   Day = dateOfBirth.Day }
                                 };
    }
}
```