C#

Readonly vs private set

<https://softwareengineering.stackexchange.com/questions/72495/net-properties-use-private-set-or-readonly-property/72500>

Use private set when you want setter can't be accessed from outside.

Use readonly when you want to set the property only once. In the constructor or variable initializer.

public class Configuration

{

public Color BackgroundColor { get; private set; }

public Configuration()

{

BackgroundColor = Color.Black;

}

public void ResetConfiguration()

{

BackgroundColor = Color.Black;

}

}

public class ConfigurationReadOnly

{

public readonly Color BackgroundColor;

public ConfigurationReadOnly()

{

BackgroundColor = Color.Black;

}

public void ResetConfiguration()

{

BackgroundColor = Color.Black; // compile error: due to readonly keyword

}

}

<https://stackoverflow.com/questions/7975661/which-is-better-between-a-readonly-modifier-and-a-private-setter>

The basic usage of the class will look exactly the same: code in other classes will only be able to read the value, not change it. Also, the code to read the value will look exactly the same.

The first one is a read-only field, while the second one gets compiled as a pair of methods (get and set method)

The first one (using readonly) will mean that the object can't even modify its own field's value, once the object has been instantiated, and others can never modify it. A readonly field can only be assigned to at declaration or in the constructor. **The value assigned to a readonly field cannot be changed (at least not in a normal way) and it is guaranteed that every thread will see the correctly, initialized value after the constructor returns**. Therefore, a readonly field is inherently thread-safe.

The second one (using private set) will mean that object can modify the value of its field after it's been instantiated, but others can never modify it.