Fields hold the data in our application. Naming it correctly helps us hold the data appropriately and access it appropriately.

Can be defined types or new classes, read-only or read write, variables or constants, public or private, built in or created classes.

Backing Fields – the most common type we use in our classes. A variable in a class that holds the data we instantiate for the class.

Data encapsulation/info hiding – an objects data should only be accessible to the object it exists. This is achieved by using access modifier private. If an outside object needs the data it should be ,ade accessible through a property getter/setter.

Accessiblility modifiers are optional. Protected is the default. Initialisation is possible in the declaration.

Nullable type – a value type, built in or defined, that allows a value to be an instance or ‘null’. It’s defined by putting a ‘?’ after the type definition. A nullable value is useful if you want to distinguish between a value ‘not set’ or its default value. C# will initialise a variable to its default value (dependent on type) if it’s not stated explicitly.

If defining a nullable type variable we must check each instance of the variable using a null test ‘?’ in the rest of the code.

‘not set’ is a null in a nullable type. Null types have a HasValue property that returns true if the nullable type has a value.

Date.ToShortDateString();

Constant – info that’s hard coded in the app. Expected to never changed. Uses the keyword ‘const’. Examples include the value of pi, RGB values….

Constant can only be a number, bool or string.

Constants must be part of a declaration that can be fully evaluated at compile time. The compiler compiles the constant into every location its referenced.

Constants are static. We don’t need an instance of a class to access them. Often publically accessible to the application. It must be initialised to a value.

Read-only fields – data fields that are set and never changed for the lifetime of the object. Declared using the ‘readonly’ keyword. It’s a variable in a class that’s initialised and then not changed. Must be initialised in declaration or constructor. Think of it as a run-time constant value.

Read-only applies to any data type, even objects. But if the read-only variable is an object, it only applies to the object and not the properties of the object.

FAQ’s

1. Explain the data encapsulation principle.

Data related to an object should only be directly accessible by the object it’s defined in. Allows control of data through controlled exposure. Backing fields should be marked as private.

1. When should you use a backing field?

To store data the object needs.

1. When should you use a constant?

A compile time constant value that will not be changed through the lifetime of the application. When defining a field with a simple data type that will never change.

1. When should you use a read-only field?

For a value that will never change for the lifetime of an object. A value initialised from a file, table or code that should not be changed again.