

Fields And Properties

Fields and properties are used to store and control access to data (integers, strings, etc.) inside a class.

Fields store the values, and by default are "protected", they can only be accessed from methods inside the class (or a subclass). You can declare a field as "public", however (see example code below).

Properties are similar to fields, except that you define a getter and a setter method that are called when one tries to retrieve the value of the property or set the value of the property. This is useful for example if you want to check the value before setting it, or making a read-only property. The default access level of properties is "public".

Here is some code demonstrating different kinds of fields and properties:

```
class MyClass:
    //a field, initialized to the value 1
    regularfield as int = 1 //default access
level: protected

    //a string field
    mystringfield as string = "hello"

    //a private field
    private _privatefield as int

    //a public field
    public publicfield as int = 3

    //a static field: the value is stored in
one place and shared by all
    //instances of this class
    static public staticfield as int = 4

    //a property (default access level:
public)
    RegularProperty as int:
        get: //getter: called when you
```

```
retrieve property
    return regularfield
set: //setter: notice the special
"value" variable
    regularfield = value

ReadOnlyProperty as int:
    get:
        return publicfield

SetOnlyProperty as int:
    set:
        publicfield = value

//a field with an automatically
generated property
[Property(MyAutoProperty)]
_mypropertyfield as int = 5

//constructor is called when you create
a new MyClass instance
// like so:  m = MyClass()
def constructor():
    _privatefield = 3

//another constructor would let you pass
a value to the class
//when creating it:  m = MyClass(33)
def constructor(fieldvalue as int):
    _privatefield = fieldvalue
```

```
m = MyClass()
//print m.regularfield //error: field has
protected access
print m.publicfield
m.publicfield = 10
print m.publicfield
print m.staticfield
print MyClass.staticfield //another way to
access static values
print m.RegularProperty
m.RegularProperty = 100
print m.RegularProperty

//this was the automatically created
property
print m.MyAutoProperty
m.MyAutoProperty = 11

print m.ReadOnlyProperty
//m.ReadOnlyProperty = 333 //error: only has
getter, no setter
```

```
//print m.SetOnlyProperty //error: only has  
setter  
m.SetOnlyProperty = 555
```

Readonly and Constant Fields

Use the "final" keyword to create a field that cannot be modified. This is similar to "readonly" in C#. The value of the field is set only once when a class is instantiated (the constructor is called). If you mark a field "static final", then the value is hard-coded at compile time. This is similar to "const" in C#.

```
class C:  
    final A = 2 //like "readonly in C#,  
value set when class instantiated  
    final B as int //you can set this value  
yourself just once in the constructor  
  
    static final B = 3 //like "const" in C#
```

More Info

To better understand properties and fields in boo, you may find it helpful to search for information on properties and fields in C# and Visual Basic .NET, such as these resources:

- <http://www.csharp-station.com/Tutorials/Lesson10.aspx>
- <http://www.c-sharpcorner.com/Language/PropertiesInCSRVS.asp>
- <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/csref/html/vclrfusingproperties.asp>