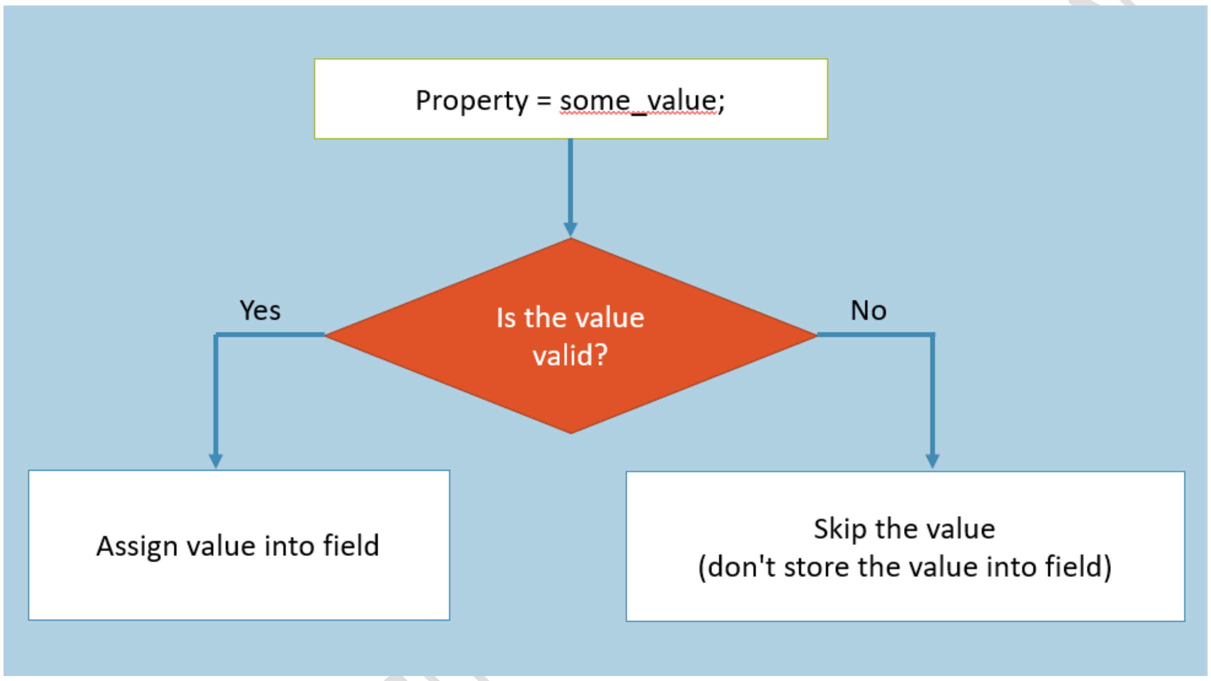
C# - Ultimate Guide - Beginner to Advanced | Master class

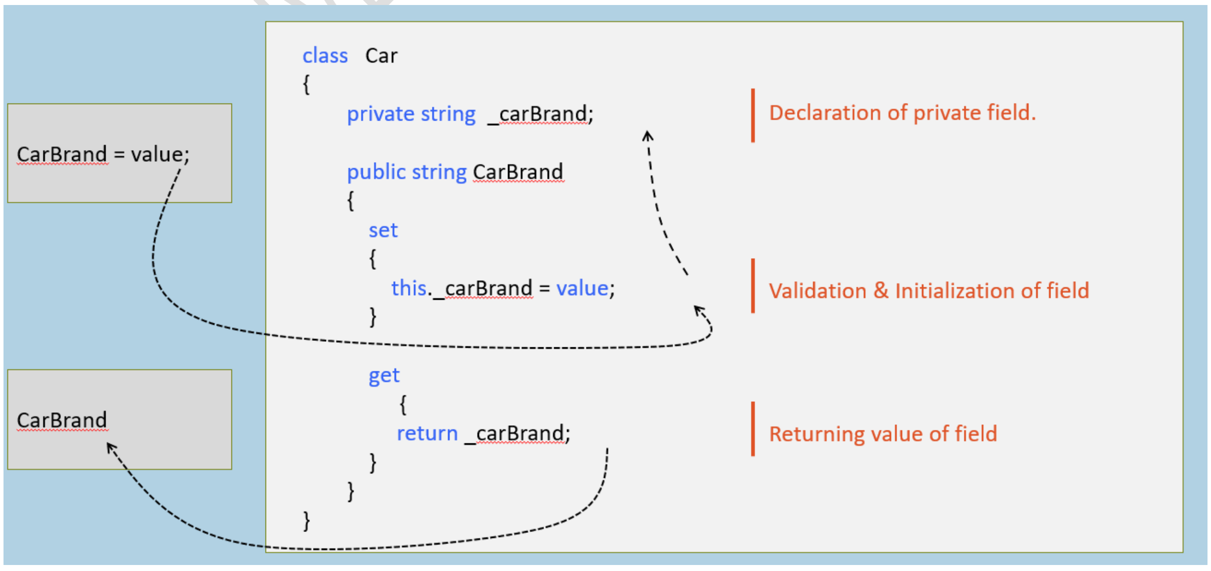
Section 9 - Properties

Properties

Receive the incoming value; validate the value; assign value into field.



Property is a collection of two accessors (get-accessor and set-accessor).

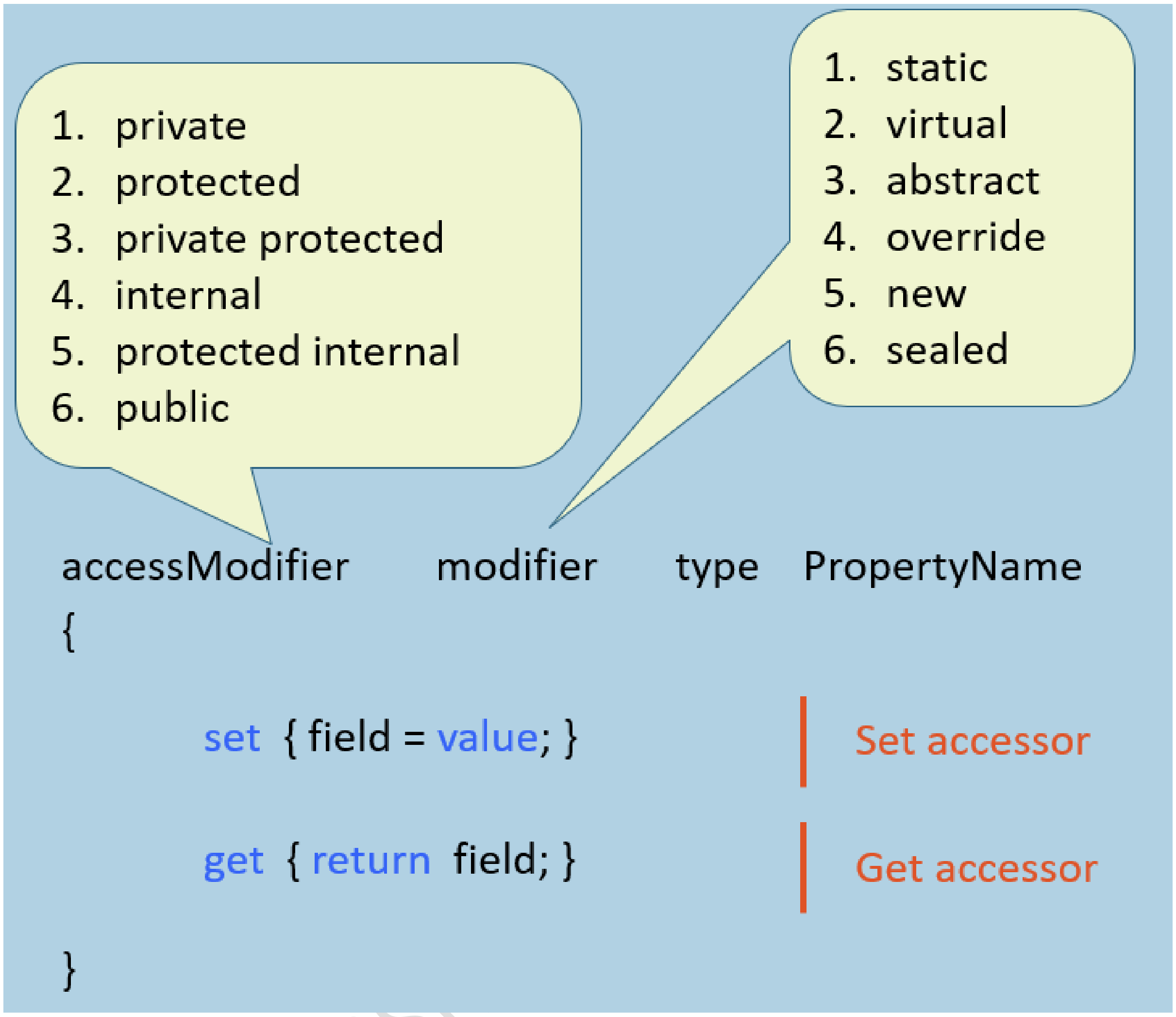

CarBrand = value;
CarBrand
Declaration of private field.
Validation & Initialization of field
Returning value of field

```
class  Car
{
      private string  _carBrand;

      public string CarBrand
      {
         set
         {
            this._carBrand = value;
         }

         get
            {
            return _carBrand;
         }
      }
}
```

Syntax of Property



Set Accessor [vs] Get Accessor

Set Accessor

```
set
{
 field = value;
}
```

1. Used to validate the incoming value and assign the same into field.
2. Executes automatically when some value is assigned into the property.
3. Has a default (implicit) parameter called "value", which represents current value i.e. assigned to the property.
4. Can't have any additional parameters.
5. But can't return any value.

Get Accessor

```
get
{
  return field;
}
```

1. Used to calculate value and return the same (or) return the value of field as-it-is.
2. Executes automatically when the property is retrieved.
3. Has no implicit parameters.
4. Can't have parameters.
5. Should return value of field.

Features and Advantages of Properties

Properties create a protection layer around fields, preventing assignment of invalid values into properties & also do some calculation automatically when someone has invoked the property.

No memory will be allocated for the property.

Access modifier of accessors:

Access modifier is applicable for the property, set accessor and get accessor individually. But access modifiers of accessors must be more restrictive than access modifier of property.

Eg:

```
internal  modifier  data_type  PropertyName
{
    private  set { property = value; }
    protected  get { return property; }
}
```

Read-Only [vs] Write-Only Properties

Readonly Property

```
accessModifier type PropertyName
{
 get
 {
  return field;
 }
}
```

1. Contains ONLY 'get' accessor
2. Reads & returns the value of field; but not modifies the value of field.

Write-only Property

```
accessModifier type PropertyName
{
 set
 {
  field = value;
 }
}
```

1. Contains ONLY 'set' accessor.
2. Validates & assigns incoming value into field; but doesn't return the value.

Auto-Implemented Properties

Property with no definition for set-accessor and get-accessor.

Used to create property easily (with shorter syntax).

Creates a private field (with name as _propertyName) automatically, while compilation-time.

Auto-Implemented property can be Read-only (only 'get' accessor) property; but it can't be Write-only (only 'set' accessor).

```
accessModifier  modifier  data_type  PropertyName
{
    accessModifier  set;
    accessModifier  get;
}
```

Useful only when you don't want to write any validation or calculation logic.

Auto-Implemented Property Initializer

New feature in C# 6.0

You can initialize value into auto-implemented property.

```
accessModifier    modifier  type  propertyName  { set;  get; } = value;
```

Properties: Key Points to Remember

It is recommended to use Properties always in real-time projects.

You can also use 'Auto-implemented properties' to simplify the code.

Properties doesn't occupy any memory (will not be stored).

Properties forms a protection layer surrounding the private field that validates the incoming value before assigning into field.

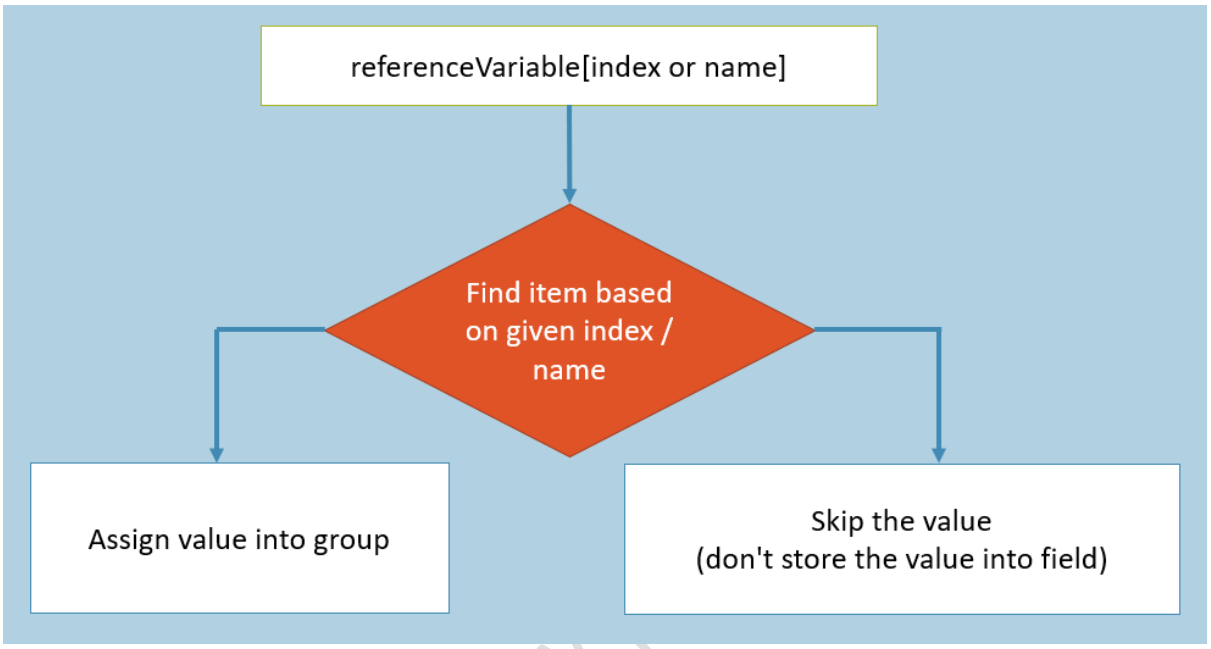
Read-only property has only 'get' accessor; Write-only property has only 'set' accessor.

Properties can't have additional parameters.

Indexers

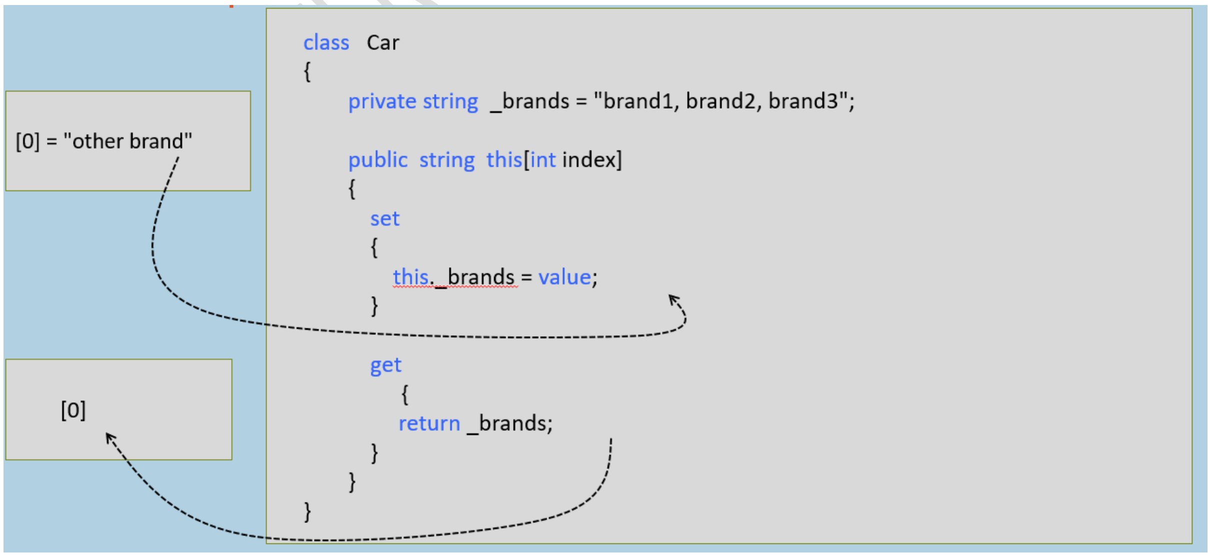
Receive a number / string. Search for the particular item among a group of items; set or get value into the group of items.

It provides shorter syntax to access a group of items.


referenceVariable[index or name]
Find item based on given index / name
Assign value into group
Skip the value (don't store the value into field)

Indexer is a special member of class, which contains set-accessor and get-accessor to access a group of items / elements.

Eg:

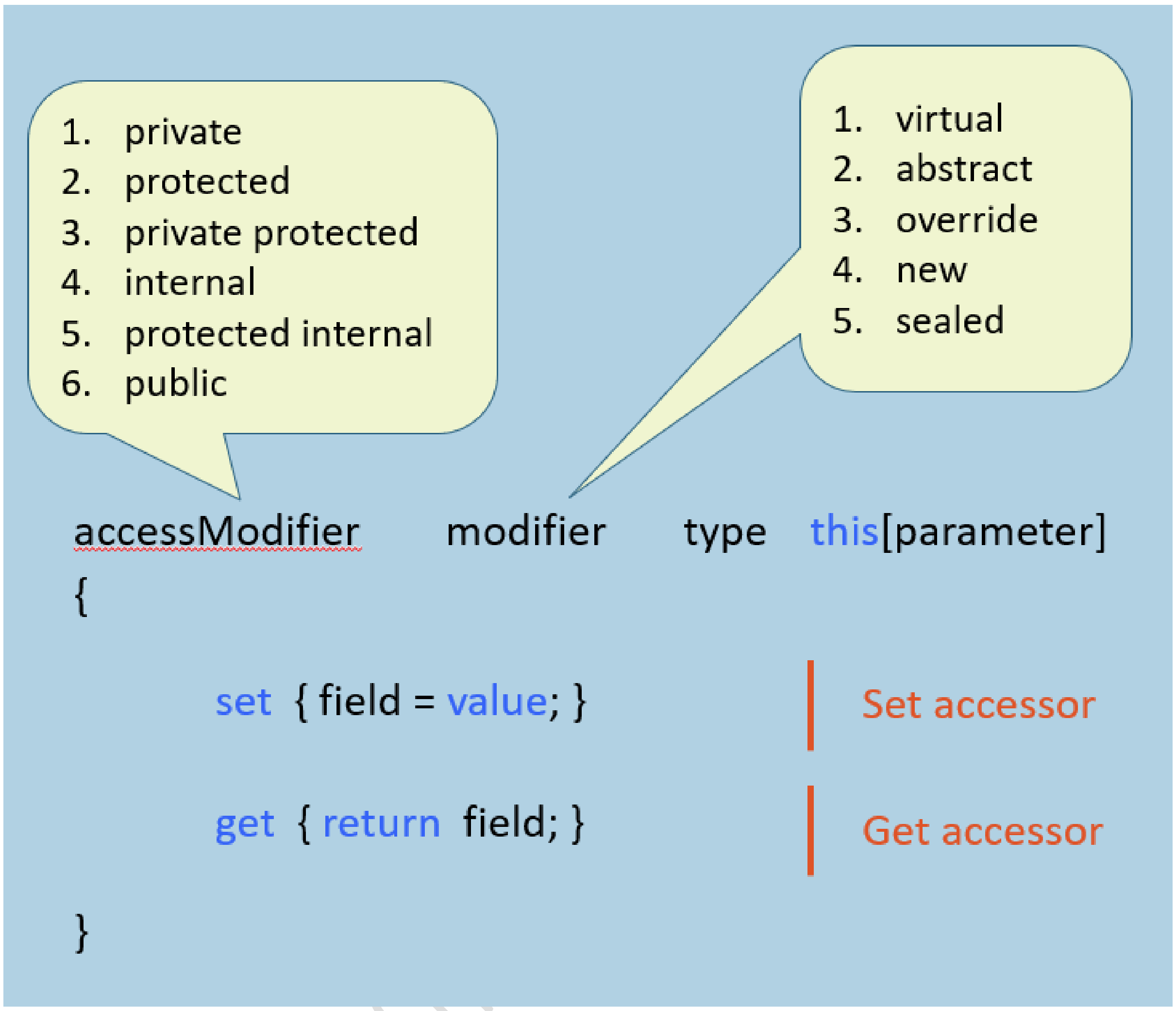


```
class  Car
{
      private string  _brands = "brand1, brand2, brand3";

      public  string  this[int index]
      {
         set
         {
            this._brands = value;
         }

         get
            {
            return _brands;
         }
      }
}
```

Syntax of Indexer



```
accessModifier    modifier    type   this[parameter]
{

        set  { field = value; }

        get  { return  field; }

}
```

Indexers: Key Points to Remember

- Indexers are always created with 'this' keyword.
- Indexers are generally used to access group of elements (items).
- Parameterized properties are called indexer.
- Indexers are implemented through get and set accessors along with the [ ] operator.
- Indexer must have one or more parameters.
- ref and out parameter modifiers are not permitted in indexer.
- Indexer can't be static.
- Indexer is identified by its signature (syntax of calling); where as a property is identified it's name.
- Indexer can be overloaded.