Read-Only and Write-Only Properties When encapsulating data, you might want to configure a *read-only property*.

To do so, simply omit the

set block.

Likewise, if you want to have a *write-only property*, omit the

get block.

For example, assume

you wanted a new property named SocialSecurityNumber, which encapsulates a private string variable named empSSN. If you want to make this a read-only property, you could write:

public string SocialSecurityNumber

{

get { return empSS; }

}

Now assume our class constructor has a new parameter to let the caller set the SSN of the object.

Since the SocialSecurityNumber property is read only, we cannot set the value as so:

public Employee(string name, int age, int id, float pay, string ssn)

{

Name = name;

Age = age;

ID = id;

Pay = pay;

// OOPS! This is no longer possible if the property is read

// only.

SocialSecurityNumber = ssn;

}

Understanding Automatic Properties

When you are building properties to encapsulate your data, it is common to find that the set scopes have

code to enforce business rules of your program. However, in some cases you may not need any

implementation logic beyond simply getting and setting the value. This means you can end up with a lot

of code looking like the following:

// A Car type using standard property

// syntax.

class Car

{

private string carName = "";

public string PetName

{

get { return carName; }

set { carName = value; }

}

class Car

{

// Automatic properties!

public string PetName { get; set; }

public int Speed { get; set; }

public string Color { get; set; }

}

**Note** Visual Studio provides the prop code snippet. If you type “prop” and press the Tab key twice, the IDE will

generate starter code for a new automatic property! You can then use the Tab key to cycle through each part of

the definition to fill in the details. Give it a try!

// *Automatic properties must be read and write.*

public string PetName { get; set; }