**STATIC/NON-STATIC**

**Define static. What is an example?**

Static is a modifier used to declare a static member that belongs to the type itself rather than a specific object. Cannot be instantiated.

**Define non-static. What is an example?**

Non-static is bound to the object and can be accessed through the objects. Can be used through instantiation.

**NULLABLE TYPES, NULL OPERATORS**

**What is a nullable type?**

A nullable type can represent a range of values along with an additional null value. Reference types by default are nullable, but other value types such as int require a null operator.

**Define the different null operators**

Null-coalescing operator = ??

Null-conditional operator = ?.

Ternary operator = ?:

**GENERICS**

**What are generics? How would this be useful?**

Generics allows you to design classes and methods without specified types. They types are assigned when a class or method is declared and instantiated.

Use generic types to maximize code reusability, type safety, and efficiency.

**ATTRIBUTES**

**What is an attribute? What is an example of an attribute?**

Attributes add metadata to your program. Metadata is information about the types defined in a program.

Obsolete Attribute used to mark a program entity that is no longer in use.

[System.Obsolete("use class B")]

class A

{

public void Method() { }

}

class B

{

[System.Obsolete("use NewMethod", true)]

public void OldMethod() { }

public void NewMethod() { }

}

**CUSTOM ITERATORS**

**Come up with an example where it makes sense to use a custom iterator. (Hint: IEnumerable)**

You might want to use a custom iteration to retrieve specific data from an enum through a foreach loop, instead of everything within the enum.

**OVERRIDING**

**Define overriding.**

Overriding is used to extend or modify the abstract or virtual implementation of an inherited method, property, indexer, or event. Override provides a new implementation of a member that is inherited from a base class.

**OVERLOADING (OPERATOR, FUNCTION)**

**Define overloading:**

Overloading is what happens when you have two methods with the same name but different signatures. At *compile time*, the compiler works out which one it's going to call, based on the compile time types of the arguments and the target of the method call.

**What is overloading an operator?**

Overloaded operators are functions with special names the keyword **operator** followed by the symbol for the operator being defined.

**ACCESS MODIFIERS**

**What is the difference between public, private, protected?**

**PUBLIC**

The type or member can be accessed by any other code in the same assembly or another assembly that references it.

**PRIVATE**

The type or member can only be accessed by code in the same class or struct.

**PROTECTED**

The type or member can only be accessed by code in the same class or struct, or in a derived class.

**PROPERTIES, FIELDS**

**What are properties?**

A property is a member that provides a flexible mechanism to read, write, or compute the value of a private field.

**What are fields?**

Variable of any type declared directly in a class or struct.

**Why would you use one over the other?**

Fields should be used in the private internal working of a class, while properties can be used to access the fields without breaking its public interface.

Project Wpf - terms: mvvm; data binding; xmal;