Static Polymorphism

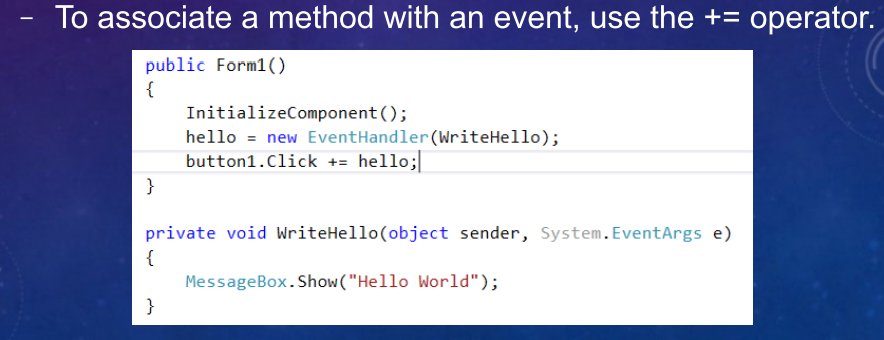
* The response to a function is determined during compile time

Dynamic Polymorphism

* Decided during run-time

namespace i.e, area => 1 dll => internal access modifier affecting the code produced here

Delegates for callbacks - research this



## Extension Methods

Extension methods are defined as static methods but are called by using instance method syntax. Their first parameter specifies which type the method operates on. The parameter is preceded by the [this](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/this) modifier. Extension methods are only in scope when you explicitly import the namespace into your source code with a using directive.

The following example shows an extension method defined for the [System.String](https://docs.microsoft.com/en-us/dotnet/api/system.string) class. It's defined inside a non-nested, non-generic static class:

namespace ExtensionMethods

{

public static class MyExtensions

{

public static int WordCount(this String str)

{

return str.Split(new char[] { ' ', '.', '?' },

StringSplitOptions.RemoveEmptyEntries).Length;

}

}

}

static => cannot new Class() it. Cannot change it.

Serialization

* Turning object into some form of data that can be stored or transmitted

Stack vs Heap

* Stack - Last in first out, self-maintaining without the need of external garbage collection
* Heap - Allows random-access at any time, need to be handled by the garbage collector

Boxing vs Unboxing

* Expensive operation to transfer from stack to heap or vice versa

int i = 123; // a value type

object o = i; // boxing

int j = (int)o; // unboxing

String test = “text”;

var text = test ?? “Default value if null”;

if test is null then it will print the text. If not, it will be the text value.

All reference types will default to null. int, char, double and bool doesn’t default to null.

String Comparison => string\_var.Equals(another\_string\_var, [StringComparison.OrdinalIgnoreCase]);

Casting

* Use as keyword vs using (cast) because it will return null if the cast is invalid
* i.e, variable as String;

Params as a parameter

void Function(params int[] value) { }

Function(1, 2, 3, 4, 5);

Use String.Format() instead of +’s because null values will throw an exception.

You can also use StringBuilder().

Verbatim String Literals

* Simplify string with escape characters by putting an @ in front of it
* Doesn’t work with quotes
* @’String with an \ escape character’

Object Initializers

new ObjectClass {

Id = 1;

Name = Nik;

}

Printing Array

* Requires manually creating a string
* int data[] = { 1, 2, 3, 4 }
* String.Join(“, “, data);

try catch finally block

* catch will only run if there’s an exception
* finally will run regardless

Using keywords as variable names

var @return = “A string variable name return”;

