ORMS? – Object Relational Mapping. A programming technique for converting data between incompatible type systems using OOP.

In the context of the course, lambdas provide a concise way of defining in-line event handlers.

worker.WorkerEvent += (s, e) => Console.WriteLine( $"{s.ToString()} {e.Hours}");

Above is an example of a lambda. (s, e) are the lambda params. The compiler will infer what type the params are at compile time so there’s no need to state explicitly the types as you would for an anonymous method.

=> is the lambda operator/separator. Separates the signature and the body.

If the lambda body doesn’t use ‘{}’ then the compiler assumes the expression will return the result of the processing. If ‘{}’ is used the ‘return’ keyword is used.

‘{}’ are needed if the lambda includes multiple lines of code.

Delegates can be used to pass methods as arguments so the executing method can be passed business rules and is blind to how the data processing will be executed.

Action<T> - a built in delegate in the .NET framework. A generic delegate that allows you to define params but doesn’t return a result. It’s a one-way pipeline. Saves you the time of writing delegate void TheDelegate( param).

Action<T> can assign a method name with matching param or a lambda with matching param.

Func<T, TResult> - a built in delegate that allows you to pass in multiple params and return a result.

Func<string, bool> TheFunc = ( s ) =>

{

Console.WriteLine( s );

return true;

};

TheFunc( "This is a Func. They can have multiple params and must have a return type" );

The generic delegate types are generated by the compiler at runtime. System.Runtime.CompilerServices.

Func is also used behind the scenes with LINQ technologies.

Lambda expressions can be constructed to query collections using LINQ.

Coll.Where( c => c.Name == “Pete”);

The above will return an IEnumerable containing all objects with the name Pete. The above is an example of LINQ filters with lambdas.

Coll.Where(c => c.Name == “Pete”).OrderBy( c => c.LastName );

The above will return an ordered collection by last name.

Coll.Where(c => c.Name == “Pete” && c.Age < 24 ).OrderBy( c => c.LastName );