# Interfaces---------------------------------------------------------------------------

3 Levels of class inheritance

1. Regular class Parent => Child

Polymorphism works here  *(Virtual in parent and override in child) optional*

1. Abstract class Parent => Child

Abstract parent must have a child class to be used.

1. Interfaces

Problem with classes is that we can only inherit from 1 class

*Think natural family : you can only belong to one family*

Interface is a structure that allows us to ‘use’ code from more than one place

*Think garden shed : tools which we ‘implement’ or ‘use’*

Class => Inherit ONE

Interface => implement MANY

Interface IUseThisTool01{

//no fields

//yes properties

Int number01 {get; set;} // Assumed public without the keyword present

//yes methods

Void DoThis(){} // Assumed public

}

INTERFACES ASSUMES EVERYTHING IS A DEFAULT TO BEING PUBLIC

Class MyClass : IUseThisTool01{

Public int number01{get; set;}

Public override void DoThis(){}

}

**Class**: Picture : Class is like the architectural blueprint which an architect creates as a plan for building. It’s not a building but a builder can turn the plan into a real project.

Computing : Class is a template for creating real objects

Class Parent{} Car p = new Parent(); p is a real object INSTANCE created from class template

Class: blueprint or template from which we create real objects using ‘new’ keyword (this is called instantiation)

Class : template

New : instantiate real object

# Docker ------------------------------------------------------------------------------------

**DOCKER DESKTOP**14/03/2019

Docker is a command line too eg powershell : Docker or Docker -v

It enables virtualisation on a very small scale

Regualer virualisation

HyperV

ESX-I

VirtualBox

* Run complete operating system eg Linux, Win10 etc, Server2016

Parallers (on apple)

Typical OS is huge : several GB in size

Container Virtualisation

Container Virtualisation

Container Instances, Unique, Shielded from each other kernel (core of operating system, shared Hardware(shared)

Typical container size is a few hundred MB

Lab: find on internet a pre-build ‘container;, download and run it

Connect to the sql server on it using SQL management studio

1. Pull docker image using

*Docker pull kcornwall/sqlnorthwind*

Have you got a valid docker image yet?

*Docker ps -a*

If not let’s first create and run the image

*docker run -d --name sql -p 1433:1433 kcornwall/sqlnorthwind*

If your image running?

*Docker ps*

List running anf non-running images

*Docker ps -a*

Status of docker container

Docker ps

Can you start and stop image

Docker start sql

Docker stop sql

Docker ps -a

Docker ps

# ASP.NET---------------------------------------------------------------------------

15 – 05 – 2019

Most of the world build websites using HTML CSS JavaScript, However there operate inside a ‘browser’ to display the user.

The job of a ‘web server’ is to

* Store fixed pages, images, scripts etc
* Generate dynamically computer generated pages etc.

Client Facebook

IP address 10.11.12.13 🡪 http request 🡪 IP address 67.68.69.70

🡨 http response 🡨

Many sites traditionally use PHP to generate HTML at the server

Java can be used (but not so much now)

Python, Ruby (RubyOnRails) can be used

Google has ‘Go’ and many more

Someone worked out how to use JavaScript on the server also . They called it NODEJS and it has changed development forever.

*Client : HTML CSS JavaScript is running Server: NODE (JavaScript) +HTML CSS*

Now the language doesn’t need to be translated as they have the same languages.

**Microsoft servers run a special language called ASP.NET**

<html>…

<asp:runat=’server’>…button

ASP : Active Server Pages

ASPX = newer XML Version

**Traditional ASP has this format :**

Display Page .aspx xml file

Code behind .cs cs code behing XML

(As seen in WPF)

**Razor Sytnax**

.cshtml with both regular html on the page plus

@{

//put code here

}

Visual Studio we can build several different types of Web Ap

1. Blank
2. ASP Web Form
3. ASP MVC Model View Controller
4. API

First Demo 🡪 Regular ASP (not MVC)

Flat file structure, Each page has a .aspx and .cs file to make it

Second Demo 🡪 ASP MVC site

M Model Data

V View What the user sees

C Controller Sorts out your request, decides which page you want to see, gets the data from the model and sends it to the view for display .

Http:// mysite.com/Controller/Action

http:// mysite.com/api/customers/ all customers

/customers/ALFKI one customer

/Home controller

/Home/About Home controller, About Action which

retrieves the About View

Controller : Deciding which View to use

Home

Index Default

Returns View Page and look for page called Index.cshtml

Pass data from Controller to View

We have seen the default Viewas . ViewData but let’s add to this

Var string01 = “…”

Return View(“Index”,string01);

# TESTING-------------------------------------------------------------------------

21/05/2019

CI-CD Continuous Integration – Continues Deployment

Every day, the developers write code. They commit it. At the same time they have often built code

TDD – Test driven development

Unit Tests(raw code testing) : create tests in code, makes sure that your code passes these tests

BDD - Behaviour driven development

Cucumber

Selenium

As a … I want to … With outcome… (user stories)

Gherkin Syntax : Write some tests like

As a USER when I click on the ‘login’ button the login form appears

As a USER when I entered my credentials I am authenticated

Run tests

If pass then push code to production => call this a ‘pipeline’

Devops

CICD

PIPELINE

Coders code +push to repo (version control) + test 🡺 PASS

🡺 automate push to production on a regular basis

**QA : Quality Assurance**

Use automated testing to verify the ‘code coverage’ of our tests and ensure high quality of code

Static test 🡺 human being reads your code

Dynamic test 🡺 computer tests your code

**Testing CAN NEVER REMOVE ALL BUGS**

RED writing test first but will fail as no code is present

GREEN write code which passes.

REFACTOR Check our code : Can we make it more efficient / readable / maintainable??

MSTEST Microsoft

NUNIT Industry standard

XUNIT Newer

# LINQ ----------------------------------------------------------------------

21/05/2019

**LINQ Select**

We have used a lot of LINQ LAMBDA but not much of the simpler LINQ plain syntax

***LAB 70 was created here. (IQueryable)***

***LAB 71 was created here. (IEnumerable)***

**IQueryable VS IEnumerable**

*Both use lazy loading ie queries not run until actually needed.*

**IQueryable** Run a query against a remote data base

Only results are returned

Lazy loading : defer until needed

**IEnumerable** Run against a local dataset

Or can specify from a database

WHOLE DATASET IS RETURNED FIST INTO MEMORY

We can enumerate (count) over the output

Lab 70

**- select all customers from these two cities.**

using (var db = new Northwind()) {

var customers =

from customer in db.Customers

where customer.City == "Berlin" || customer.City == "London"

select customer;

//Data actually loaded here

foreach (var customer in customers)

{

Console.WriteLine($"{customer.CustomerID,-15} {customer.ContactName,-20} from {customer.City}" );

}

* **Select one customer via pk**

var oneCustomer = db.Customers.Find("ALFKI"); // only works with primary key

Console.WriteLine($"One Customer is {oneCustomer.ContactName} from {oneCustomer.City}");

* **Orderby**

var allCustomers =

from c in db.Customers

orderby c.City

select c;

foreach (var customer in allCustomers)

{

Console.WriteLine($"{customer.CustomerID,-15} {customer.ContactName,- 30} from {customer.City}");

}

**Doing statistical analysis : GroupBy**

Simple queries have been handled so far but we have not collated and stats eg Sum, Average, Min, Max , Count.

We can go through a data set and create these stats and then group the results to show stats by category

**SMART LINQ QUERIES**

**RANDOM**

String.format($”{field.value}”) has string interpolation in there as well

:n number

:d date

:t time

: percentage

**.NET**

Framework Legacy existing structure in Windows(4GB)

Standard Attempt to unify all versions of .Net going forward

Core New and lightweight version, built for the web Linux, MacOS, Windows.

# SQLite-----------------------------------------------------------------------------------

**Create table**

PS C:\data> sqlite3 test3.db

SQLite version 3.28.0 2019-04-16 19:49:53

Enter ".help" for usage hints.

**View/select table**

sqlite> .database

main: C:\data\test3.db

**Create table.** Integer means the id is incrementing

sqlite> create table animals(ID integer primary key, name text not null);

sqlite> .tables

animals

**Insert Data**

sqlite> insert into animals(name) values ('fluffy'),('bob'),('garfield');

**View all the data in this table**

sqlite> select \* from animals;

1|fluffy

2|bob

3|garfield

sqlite>

# Tech terms----------------------------------------------------------------------

## Serial computing

Serial means one by one

Serial line : cable or channel where data is sent in individual bit or bytes or packets or ‘buffers’ in a stream

Serial is good for long distance communications

Anything outside your mother board is ‘long’

USB

SATA hard drive

Internet line

## Parallel computing

Inside a mother board the distances are very small but predictable

Create 64 channels and send data 64 bits at a time

= Parallel processing

All computers have today

Also to make computers go faster we add more processors ‘in parallel’

Ie

## Interfaces

I….. at the start (convention)

### IEnumerable

Has a Numeric INDEX eg. Array, List not Queue, Stack

myArray[0], myList[0] not myQueue[0]

Power of array is instant access to any member through the index

RAM : Random Access Memory

### IQueryable

Querying over a database : LINQ query output

### IDisposable

C# is Strongly Typed : All types determined at compile time.

Speed: at run time code is now much faster as much less checking to do on ‘types’

Normal C# realm at RUNTIME : Garbage Collector by itself erase all objects when finished with, on a random schedule set by itself

But!!

Interact with databases, files systems, networks or memory objects : there are outside neat C# framework. TYPES cannot be neatly encapsulated inside c#

Dynamic keyword : treat type as a regular ‘object’ of no fixed type

Eg importing Microsoft Office Libraries

The ‘garbage collector’ does not know when to get rid of memory references to there outside

Implement IDisposable and call the ‘Dispose()’ method manually when done with these objects

### IComparable

Implements the method ‘CompareTo’ where we can create a customer comparison between 2 objects

Apple1.CompareTo(Apple2);

Output 1, 0, -1

1 if apple bigger, -1 if smaller, 0 if same

## Checked / unchecked Lab 75 max length of ints so loops to the minimum (-numbers)

Checked : throws an exception if your number gets too big or small

Unchecked : turns off again (default)

Dynamic : turns off type checking at compile time

Dynamic x = 10;

x = true;

(note : Javascript does this by default )

*This came up in the sparkhire interview …*

**Dynamic language :** loose checking of types: event at runtime the type can possibly change as is the case with JavaScript and also C# with the ‘dynamic’ keyword.

**Statically typed /strongly type language :** types are strictly defined at compile time.

Var keyword: type is still strictly inferred at compile time

**Javascript –** Loose type checking

Typescript – plug in javascript where types are more strictly controlled

Using Strict; is Javascript in ‘strict’ mode

## Static Keyword

Attached to parent class, not instance required.

# .Net Core --------------------------------------------------------------------------------------

Useful as it’s cross- platform and now open source

.Net Framework is huge(4.2GB) and Windows only

.NetCore is lightweight and Windows /Linux/Mac

Let’s add these packages

Microsoft.EntityFrameworkCore.SQLServer

Microsoft.EntityFrameworkCore.SQLite

Working with NuGet Console ( case – insensitive )

NuGet : graphic utility

Improve as programmer 🡺 efficient with command line

Use **Get-Package** to see what is installed already

Find-Package EntityFramework

**Install**

Install-package microsoft.entityframeworkcore.sqlite -ProjectName lab\_77\_Entity\_Core

Install-package microsoft.entityframeworkcore.sqlserver -ProjectName lab\_77\_Entity\_Core

Now we have to build our code – but this time from raw!! Quite a lot of work

GitHub : Philanderson888 : c-sharp : Entity\_10\_Core\_Northwind

Products

Categories (related table)

One Product will belong to one category

[Above Certain Columns : add Description or Type of Change of Name]

Relationships

Table relationships expressed with

Class Category{// properties

//end : relationship

Public virtual ICollection<Product> Products {get;set;}

**Configure the database**

Class Northwind : DBContext{

Public DBSet<Category> Categories {get;set;} // create table

}

FluentApi is name given to overridden

# Overnight lab : build your own To-Do Application in WPF

**Goals**

Users UserID UserName

Categories CategoryID CategoryName (Home Work Family)

Tasks TaskID TaskName DateCreated DateDone Done CategoryID UserID

(my goal.. add a order by function to orderby date done or and by category for one person)

**Build an app which**

Creates database from a manually run script (either SQL or c# code-first)

Populates database with initial data

Correctly related tables : One task has one user and one category

Minimum : List/ Add/ Remove Tasks

Maximum : FULL CRUD ie List/Add/Remove/Update Tasks+Users+Categories

**Trainer Help : SQL Script**

Use Northwind; // try not to use if can

drop database ToDo

go

create datebase ToDo

use ToDo

go

CREATE TABLE Categories(CategoryID INT NOT NULL IDENTIFY PRIMARY KEY, CategoryName NVARCHAR(50) NOT NULL);

CREATE TABLE Users(UserID INT NOT NULL IDENTIFY PRIMARY KEY, UserName NVARCHAR(50) NOT NULL);

CREATE TABLE Tasks(TaskID INT NOT NULL IDENTIFY PRIMARY KEY, TaskName NVARCHAR(50) NOT NULL, Done Bit Null, );

TaskID TaskName DateCreated DateDone Done CategoryID UserID

# Things to remember/practise-----------------------------------------

## Remember

Instantiating Var c = new Class() *you create a constructor*

abstract void DoThis(); *It’s doesn’t use the code implementation {}*

## Practise

Dynamic

Ternary operator (condition)? Action 1 : action 2

List<Customer>

Using (var db = new db())

# ASYNC

It’s like a butler.

You tell them what tasks to you need … they do it while you work. – fast – non-stop

Unlike sync – you wait while the task gets done – stops coding process - slow