# Progression Framework

## .Net

* + The .Net Framework is used to create both Form-Based and Web-Based applications and Web Services.
  + #Clarify stuff about .net

## Inheritance and Polymorphism

* + Inheritance is when one class is allowed to inherit features(fields and methods) of another class
  + This is better known as when a child class inherits from a parent class/ or a base class inherits from a super class
  + Polymorphism is when a child class inherits from a parent class but changes the behaviour of that method
  + It can also be described as providing an ability to take more than 1 form.
  + This is a combination of 2 words.. poly means multiple and morphs means forms so basically it means many forms

## Interfaces

* + An interface represents a contract between an object and its user. It is a collection of methods and property declaration’s
  + explicit interfaces is when a single class inherits from multiple interfaces that have the same method signature

## Classes

* Class is a user-defined blue print, basically a class combines the fields and methods into a single unit

## Generics

* Generics allow the user to define classes and methods with a placeholder, this basically allows types and user defined types to be a parameter to methods, classes and delegates
* #maybe expalin more, with T , also the limiting

## Delegates

* A delegate is a reference type variable that holds the reference to a method

## Git

* Another doc added with git commands
* Git is a type of version control that allows you to track changes to your files

## Css3

* Is the latest of the cascading style sheet language. it allows things like shadows, gradients, animations etc

## Html

* Html5 is the latest version of the Hyper text Markup Language.
* It provides more features that previously had to be done in javascript
* Html is used to structure and present the content for the web

## Linked List

* + - Linked list is a linear data structure which consists of a group of nodes in a sequence.
    - Each node contains 2 parts:
      * Data-Each node can store data
      * Address-each node contains an address to the next node if a single linked list but if a doubly linked list then it contains the previous aswell
    - Lined list are easy to implement when inserting and deleting data
    - It has a faster access time.
    - Backtracking is possible

## Stack

* Stack is a special case collection which represent a LIFO concept
* The process of adding to a stack is with a push, to remove is done with a pop

## Queue

* A queue represents a FIFO.
* It is used when you need a first in first out access of items
* To add to a queue is with enqueue, and to remove is with dequeue

## 

## SOLID #Work some more on these especially L and D

### S – single responsibility principle

A class/method should do one thing(job) and do it well. Its based more on the context of it eg. You

could have a userManagement controller that would have all the methods in that class (Add, edit,

delete) and you could also have a controller for each user function(AddUserController etc.) they say

methods should have 2-3 lines but that’s not really true.. it more depends on the context of the

code.

### O – open closed principle

Open for extension and closed for modification.

Simply means a class should be easily extendable without modifying the class itself.

### L – liskov substitution principle

This allows you to replace objects of a parent class with objects of a sub class without breaking the

app. This requires the subclass to behave the same way as the parent class.

Eg of this was the duck example.. the parent class is a duck and the method is quack.

When a diff type of duck inherits from the duck class it still quacks with a quaaack.

But if a rubber duck inherits from a duck .. it cannot quack so it returns a not implemented exception

which then breaks the app.

### I – interface segregation

This basically states that no class should be forced to depend on methods that it does not use.

You should instead create a new interface and let you class implement multiple interfaces.

### D – dependency inversion

This is a way to decouple modules. We use dependency injection to use other classes methods in our app.

## Pair Programming

* Pair programming is when 2 developers work together on one work station.
* One is the driver, he or she has the keyboard and does the typing
* The other is the observer or navigator who reviews the code as its being typed.
* The two should switch roles often.

## Periodic retrospectives

* This is a technique that is done to find out if the team can improve in any ways
* Usually its a like, dislike, more off and keep doing
* Its the dislikes and more off that’s the most important feedback
* Can be done every 3-6 months, depending on team and size of the team

## -Kaizen

* This is a Japanese philosophy
* Kai meaning change and zen meaning good
* This means to continuously improving
* A good example of this would be to do katas often which will improve dev skills

## Continuous integration

* CI is a practice where developers integrate code into a shared repository frequently. at least 1 commit a day is best.
* Each integration can then be verified by an automated build and automated tests.
* Example of this is azure devops

## Office 365

* I would say I am an advanced beginner in Office 365 as I attended a course when I was at campus.

## Sonar

* I used this tool once in Fortel, it basically gives an overview of the project and what can be improved.
* It has a rating of high, medium and small issues. Some of the issues have examples of how it can be fixed.
* This is a good tool if you are looking to improve your application and also helps with maintainability of the project

## Windows

* It is an operating system produced by Microsoft
* -#not sure how to explain windows