

Lec 1	Lec 1 (cont)	Lec 2 (cont)	Lec 3 (cont)
<p><b>Abstraction</b> Removing( or hiding) unneccessary complication/detail Good: Array.Sort(); Bad: nested for loops etc.</p> <p><b>Abstraction Levels</b> Programming Languages(Assembly low) ← High Compiler OS Architecture Circuits Physics ← Low API- Defines operations (and efficiencies)</p> <p><b>API-Application Programming Interface</b> High level specs of ops, e.g. Array, Add, Remove, Contains, IsEmpty, Graphics.Draw, etc.</p> <p><b>Levels within Programs</b> API ← High Classes/Data Structs Functions/Methods Variables/Statements Types/Operators ← Low</p> <p><b>Software Practices Steps</b> 1. Analysis – what do we need to do? 2. Design – how do we do it? 3. Implement it - Coding Yay! 4. Test, debug, improve 5. Go back to 1-4 as needed</p> <p><b>Why Comment/Document?</b> Your code (in the RW™) will outlive you. Make your successor's life easier Make your (future) life easier</p>	<p><b>XML Syntax</b> &lt;tag&gt; info &lt;/tag&gt; e.g., &lt;summary&gt; ... &lt;/summary&gt; &lt;param name='val'&gt; ... &lt;/param&gt;</p> <p><b>When to Comment?</b> ALL files must have a header comment! ALL methods must have a header comment All fields/properties should have a comment</p> <p><b>Why XML?</b> Machine readable, parsable, intelligensable</p> <p>C# source files can have structured comments that produce API documentation for the types defined in those files. The C# compiler produces an XML file that contains structured data representing the comments and the API signatures. Other tools can process that XML output to create human-readable documentation in the form of web pages or PDF files, for example.</p>	<p>Delegates are fully object-oriented, delegates encapsulate both an object instance and a method. Delegates allow methods to be passed as parameters. Delegates can be used to define callback methods. Delegates can be chained together; for example, multiple methods can be called on a single event. Methods don't have to match the delegate type exactly. i.e. Variance in Delegates.</p> <p><b>Delegate is the definition</b> of the "Type" of function. Lookup is just a name I choose!</p> <p><b>Example</b> public delegate int Lookup( string name );</p> <p><b>Delegates as Params</b> Delegates allow you to "pass" functions to other functions int doit(string x){ ... } // ← Meets Delegate Requirements ... Evaluate("1+var", doit)</p>	<p><b>C# Differences w/ Java</b> C# functions are "first-class" (see delegates) Generics can use primitives (List&lt;int&gt; vs. &lt;Integer&gt;)</p> <p><b>C# Access Modifiers</b> public, protected, and private are the same as in Java. Internal means any method in the compilation unit (e.g., project) can treat the variable as public, but "outside" users treat it as private. Why do we have these? Answer: Because programmers are human and programming is hard. These modifiers "say": Only allow the person most familiar with the code to make changes to data; e.g., the library code itself has private variables (such as the stacks) and only the person writing that code "knows enough" to use them. The "outside world" user just uses the Evaluate method.</p> <p><b>Readonly declaration</b> readonly int max_stack_depth = 5; "variable" can only be set in constructor (or in field declaration)</p> <p><b>Interfaces</b> Same as java Contract for a method (somewhat like a delegate)</p>
	<p><b>Lec 2</b></p> <p><b>Delegates</b> A delegate is a way to provide a TYPE for a function return type, parameter list/types You can use delegate types to store functions in variables "call" those saved functions using the variable name</p>	<p><b>Lec 3</b></p> <p><b>C# Similarities w/ Java</b> Compiled to an intermediate form Run with a "runtime environment" Automatic memory management Syntax mostly the same (or one-to-one transition)</p>	



By [\\_allisonwalker](#)

Not published yet.

Last updated 16th March, 2023.

Page 1 of 14.

Sponsored by [CrosswordCheats.com](#)

Learn to solve cryptic crosswords!

<http://crosswordcheats.com>

### Lec 3 (cont)

#### Namespaces

Keep common terms separate

e.g.,

```
namespace Math { public
```

```
Vector{...} }
```

```
namespace Collections { public
```

```
Vector { ... } }
```

Requires "using" statement

using Math;

#### Escaped Strings

How do you put a "new line" into a string?

Answer: "\n" The \ (back slash) is called the "Escape Character"

Used to combine "regular characters", i.e., 'n' and \ to form a special character

Common escaped characters: /t, /n, /e

Example: string filename =

```
"c:\\documents\\files\\abc.txt";
```

Note: Unix uses slashes, so you don't have to "double up": (e.g.,

```
/home/germain)
```

#### Verbatim Strings

Use the @ symbol

Escaped strings are ugly :( hard to read )

Remove some of the ugliness:

```
string filename = "c:\\documents-
```

```
\\files\\abc.txt"; vs. BAD: string
```

```
filename = "c:\\documents\\files\\a-
```

```
bc.txt"; GOOD: string filename =
```

```
@ "c:\\documents\\files\\abc.txt";
```

Useful sometimes in regular

expressions

### Lec 3 (cont)

#### Interpolated Strings - \$

Do This! (Almost always!)

How do you put variable values into a string?

Old, Bad way: string s = "Your score of " + score + " is a good one!";

New, Good way: string s = \$"Your score of {score} is a good one!";

#### Solutions & Projects

Solution → Spreadsheet

Contains:

Projects FormulaEvaluator

(Library Application)

FormulaEvaluatorTester

(Console Application)

Projects Contain:

code files, etc.

Contain References to other

projects/libraries

Tester must "reference" Formul-

aEvaluator

#### Lambdas

Shorthand notation for: Defining

small anonymous functions

Inline

Useful for: Testing, "One offs",

Some GUI applications, etc.

Syntax:

```
( [param1, param2, ...] ) => {
```

```
code };
```

```
( [param1, param2, ...] ) =>
```

```
expression to return;
```

### Lec 4

#### Version Control Use

Collaborative development

Branching/Merging Code

archeology/File history

Differences

Backup

#### GIT Functions

Committing: Save work as a

"Version" with a "Message"

Branching: Try something else out which may or may not come back into the main branch later via Merging.

Push/Pull: Send changes from one place to another

TAG: Like a "bookmark" to find a particular state of your code

Code History: Compare one

Commit to Another

Diff Split vs. Unified

#### Extensions

Extensions allow us to add "dot methods" to classes we don't own, use this new functionality as if it is "built in"

Example: name.CapitalizeFirstLetters( );

### Lec 4 (cont)

#### IEnumerable

Guarantees that something is countable and can be iterated over Arrays, Lists, Trees, Dictionaries, etc ← all are IEnumerable

Can convert the above to "base" types using: toList, toArray

Allows support for: foreach loop

Has a templated type

Is an Interface: GetEnumerator

```
Current (array → int current = 0; )
```

```
MoveNext (array → current++; )
```

```
Reset() (array → current = 0; )
```

#### Interface

A specification guaranteeing that an implementing class will have certain functions

Guarantee made by the

compiler

There is no guarantee the

functions work, just that they are there.

Aside: Useful also with Polymorphism:

```
Movable x; // A Movable must
```

```
have a .move() method
```

```
Car c = new Car();
```

```
x = c; // public class Car :
```

```
Movable
```

```
x.move(); c.move()
```

#### Enumerator

Enumerator is an object that

allows iterating over the object



By [\\_allisonwalker](#)

Not published yet.

Last updated 16th March, 2023.

Page 2 of 14.

Sponsored by [CrosswordCheats.com](#)

Learn to solve cryptic crosswords!

<http://crosswordcheats.com>

### Lec 4 (cont)

#### IEnumerable

A high level type (interface) that allows us to apply actions to a list of objects  
Useful with foreach loop  
Useful when we don't care (or know) the underlying data structure  
Makes our code more robust and general!

### Lec 5

#### Unit Tests

Test the "smallest" pieces of the software system (E.g., methods)  
Simplifies tests, simplifies debugging  
"Easy" to find the failure point

#### Self Contained Testing

Tests should not rely on "outside" state  
Test should:  
define data (e.g., build a Dependency Graph)  
Execute algorithm (e.g., remove/add dependencies)  
Verify (e.g., using asserts)

#### Code Review

Higher Quality Code  
Fewer Defects  
Easier to Maintain  
Developer Training Learn about other parts of codebase Learn about other techniques

### Lec 6

#### Immutable

Strings are immutable, they cannot change.  
Thus once we write:  
string s = "jim";  
That string object will `_always_` contain "jim"  
Question: s = "jess" does not change the string, but instead creates a reference.

#### Mutable

Example: a stack is mutable (it can be changed),  
StringBuilder class allows us to modify strings

#### Methods that Modify(Mutate) an Object

are called mutators  
The stack class is "Mutable"  
Can be modified after creation  
The string class is "Immutable"  
Cannot be modified after creation  
Q: How can you make any object Immutable (to a programmer using your object?)  
A: Make all fields/Properties/-setters PRIVATE! (protected)

### Lec 6 (cont)

#### Properties

C# has a convenient notation for building getters/setters  
Can control public/private of get and set independently  
Often properties are "backed" by private member fields  
As always, member fields should default to private  
"private" properties can often just be private member variables  
Example:  
a private property could be used for lazy instantiation of an expensive field: e.g., private password ( get  if password==null, do expensive password get operation)

### Lec 6 (cont)

#### Properties

Properties are similar to named fields in the object  
Properties should be aspects of an object that can be understood independently of the rest of the representation  
Think, for example, of the current hour of the day (from a Date class where internal representation is seconds since a prior date)  
Public properties (which is usually the case) provide a "contract" with user of your code  
Has a convenient notation for building getters/setters Can control access to get and set (i.e., private) independently  
As always, don't make every member variable public  
"private" properties can often just be private member variables



By [\\_allisonwalker](#)

[cheatography.com/allisonwalker/](http://cheatography.com/allisonwalker/)

Not published yet.

Last updated 16th March, 2023.

Page 3 of 14.

Sponsored by [CrosswordCheats.com](http://CrosswordCheats.com)

Learn to solve cryptic crosswords!

<http://crosswordcheats.com>

### Lec 6 (cont)

#### Func v.s. Delegate Notation

If you aren't going to have (1) any useful/descriptive words, or (2) Or it is a "one off" then use "Func" notation

```
delegate int MathOnTwoNumbers(int x1, int x2);
int add(int a, int b) { ... }
AddTwoNumbers f1 = (a,b) => a+b; // lambda to fulfill Delegate
Func<int,int,int> f2 = (a,b) => a+b; // lambda to fulfill Func
AddTwoNumbers f3 = add; // regular function to fulfill Delegate
Func<int,int,int> f4 = add; // regular function to fulfill Func
```

#### Delgate v.s. Func

Why Use Delegate?  
Named, Documented, Provides More Context  
Evaluate(string formula, VariableLookupMethod lookup)  
Why Use Func?  
Shortcut, Types are "right there to see"  
Evaluate(string formula, Func<string,int> lookup)

### Lec 7

#### Global Variables

Data that is accessible from "anywhere" in the program  
If a method modifies one, it is a "hidden side-effect"  
Software Practice - Almost always:  
Best Case: Data → Function → return New Data  
Nothing else changes!  
Okay Case: Data → Function → parameters "change" (see out/ref params)

### Lec 7 (cont)

#### Memory

Stack  
Methods (and their Variables)  
Heap  
Objects  
Methods go on the Stack  
Variables in Methods go on the Stack  
Objects go on the Heap  
Changing something about a "Shared" (Aliased) variables is "seen" by all other aliased variables

#### Memory Diagram

Every function should be labeled on the stack  
Every variable is kept on the call/activation stack  
References (to objects) have "arrows" to heap  
Value variables have values on stack  
Object methods should show "this" explicitly on the stack  
Every new object is kept on the heap

#### Pass by Value & Ref

By Value  
- a copy of the value is added to the stack frame of the called method  
By Reference  
- a "pointer" to the object is added to the stack frame of the called method

### Lec 7 (cont)

#### In, Out, Ref Params

In  
- makes Reference constant  
WARNING: does not make object constant  
Out  
- refers to calling methods variable  
Places object "in" calling methods variable  
Must be assigned a value  
Cannot use "what was there before"  
Ref  
- refers to calling method variable  
Can use what is there  
Does not have to assign a value  
Out and Ref the same except compiler enforces semantics

#### Structs

Structs are like VALUE types  
They go on the Stack  
Unless part of an object  
Let's Draw a picture for  
struct Pt { int x; int y; } main() { Pt pt1 = new pt(); Pt pt2;

#### Structs

Constructors allowed, but not required  
new() → calls constructor  
Does not put object on heap  
Can have methods  
Getters/Setters/Properties  
Usually do not!  
Why do structs go on the stack?  
Efficiency!  
No allocation!  
No deallocation/Garbage Collection.

### Lec 7 (cont)

#### Memory Efficiency

the closer the data is to the CPU the faster the memory (and the more costly, thus there is less of it)

### Lec 8

#### IEnumerable

What is the "High Level English" meaning of an IEnumerable?  
Guarantees a "list" of "stuff" coming back.  
What are some classes that implement IEnumerable?  
List, HashTable, BST, Array, etc.  
IEnumerable iterable = new BST();  
IEnumerable iterable = new List<string>( ... ); // etc  
What does an IEnumerable "give you"?  
A way for OUTSIDE CODE to "walk" the data:  
foreach (var item in iterable) { Print(item); }

#### IEnumerable uses an Enumerator

Implemented by an Enumerator Class  
Enumerator has low level functionality  
Bool ← MoveNext // if there is another item, return true  
Current // return the current item



By [\\_allisonwalker](#)

Not published yet.

Last updated 16th March, 2023.

Page 4 of 14.

Sponsored by [CrosswordCheats.com](#)

Learn to solve cryptic crosswords!

<http://crosswordcheats.com>

### Lec 8 (cont)

#### Linq

Adds "Database-like" syntax to C# (somewhat modeled on SQL database language).

Can convert IEnumerable to Arrays or Lists

Using System.Linq;

```
public void doit( IEnumerable container ) {
```

```
List list = container.ToList();
```

#### Yield Return

C# has a way to Only partially execute a function

Can later return and complete more of it later!

This is called: Yield Return

Yield Return SAVES the entire execution state of the method call (i.e., the call (activation) stack) for continued execution at a future point!

#### Yield Return Explained

When the line with yield return is encountered:

The runtime saves the entire STATE of the method

Local Variables

Parameters

When the method is called

again, the runtime restores the state and continues

From the last point of execution!

### Lec 8 (cont)

#### Why we need Yield Return

Example: Imagine a GUI with a button that populates a textbox with the next prime.

```
static IEnumerator iterator = next_prime().GetEnumerator(); // actually need Enumerator void button_action() {
```

```
    iterator.MoveNext(); // hidden: computes next prime and saves state of method text_box.Text = iterator.Current; }
```

#### REGEX

[abc] ← Square Brackets: match any character in here

a|b ← OR - Match an 'a' or 'b' character

a ← Match zero or more (little) a's

d+ ← Match one or more (little) d's

ld+ ← Escape: Match one or more digits!

ls ← Match whitespace

a? ← Match zero or one little a's (a) ← If we find this, put it in "group" 1

a\$ ← a at end of string!

### Lec 8 (cont)

#### Yield Return

Use when you either

CANNOT create entire list of results

TOO COSTLY to create entire list of results

Need to SPREAD computation over longer time Might not need all values!

Question: Binary Search Tree with Nodes

```
{ Node left; Node right;}
```

How would you build the iterator?

Answer: yield return can use recursion!

#### Yield Return BST pseudocode

```
IEnumerable traverse( Node current )
```

```
{ if (current == null) return; yield return this.Value; traverse ( this.left ); traverse ( this.right ); }
```

#### Func ↔ Delegate shortcut

```
public delegate string Normalize(string); // Definition
```

```
Normalize function = s => s; // Usage with Lambda
```

```
Func<string,string> function = s=> s; // Func notation
```

Func is simply inline (shortcut) notation for delegates without a name.

### Lec 8 (cont)

#### What does == do?

== is REFERENCE equality

Are X and Y the same object in memory.

overloaded ==

Can mean anything you want it to mean, but...

Usually means VALUE equivalence "jim" == "jim"

.Equals

Means value equivalence

#### DRY -Don't Repeat Yourself

Dry is a fundamental principles of good software development

Don't repeat Data!

Violate sometimes for speed

(can be a good reason)

Don't repeat Code!

Violate sometimes because we are lazy (bad reason)

Example: using extensions is a good dry practice

#### Dry Code

Move common code to helper method(s) Make Library for common code Use Extensions Library

#### What is an Invariant?

A condition that must always be true.

For example:

If A1 depends on B1

Then B1 must have A1 as a dependee



By [\\_allisonwalker](#)

Not published yet.

Last updated 16th March, 2023.

Page 5 of 14.

Sponsored by [CrosswordCheats.com](http://CrosswordCheats.com)

Learn to solve cryptic crosswords!

<http://crosswordcheats.com>

### Lec 8 (cont)

#### How do we ensure the Invariant on this?

```
public void ReplaceDependents(string s, IEnumerable<string> newDependents) { }
```

Answer: this function should not touch Map 1 or Map 2  
 Answer 2: this function should only use addDependency and removeDependency

#### MVC

Model - Data (and some methods to compute on data)  
 View - What the user sees  
 Controller - Interactions between User and Model

#### NameSpaces

SpreadsheetUtilities  
 Internal namespace to your project for your use/protection  
 SS ← Added to project now  
 External namespace your project "shows" to the world, similar to System, or Math, or Collections...

### Lec 9

#### Abstract Class

Defines WHAT must be done (via methods)  
 Defines Return and Parameter Type Signatures  
 Defines generic (reusable) code  
 Defines generic data (if applicable)  
 A child class can only implement one  
 Can have a constructor

### Lec 9 (cont)

#### Interfaces

Defines WHAT must be done (via methods)  
 Defines Return and Parameter Type Signatures  
 No Code  
 No Data  
 A class can implement many  
 No constructor

#### MVC

Model - Working on this now  
 Data → Cells - Contents / Values  
 Functionality → Dependencies (See requirements in Abstract class)  
 View - GUI (not working on this now)  
 Controller - actions available to user  
 often initiated through the GUI  
 Not working on this now

#### Why Comment?

Shows that you understand your code  
 Allows partners and future developers to understand your code before reading it line by line  
 Can be used PRIOR to coding to set specifications/reminders  
 Can be "scraped" to provide external documentation  
**<inheritdoc>**  
 Does just that. {[nl]}Inherits documentation from parent (class)  
 Why is this DRY?  
 DRY since we don't repeat ourselves/code

### Lec 9 (cont)

#### How/When to Comment

Header Comment vs. Inline  
 Header - High-level overview of desired functionality  
 Don't "re-write" code in comments  
 Question: What is "comment rot"?  
 Inline  
 For sections of a long function (alternatively, helper methods)  
 For tricky code  
 Use descriptive variable names to alleviate the need for some comments

#### Self-Documenting Code

Wrong - "Don't Comment your Code!"  
 Right Comment for Why  
 Provide links to algorithms, examples, docs  
 Rename complicated expressions with understandable names

### Lec 9 (cont)

#### DRY

Do not repeat yourself  
 Do not repeat code  
 Use helper methods  
 Do not repeat data\*  
 Transform requests/inputs into a single standard form  
 Question: when "can" we repeat data  
 Answer: in a very local location (e.g., a single class file) where invariants are set (e.g., input into the dependents always updates dependees) and only if we get a strong efficiency boost that (in your expert opinion) is worth the SE hit.

#### Yield Return

Used for Enumerations (IEnumerableable)  
 Operationally: Saves Code/Method Call Stack  
 Used for: "infinite" or "costly" enumerations

### Lec 10

#### GSP- Good Software Practices

Good use of Versioning  
 Multiple Commits. Good Commit Messages  
 Tags at "Releases"  
 Good use of Testing  
 Extensive Unit Tests  
 Well named/useful tests.  
 Documentation  
 READMEs  
 Located in Solution Folder (and all Project Folders)  
 Hour tracking  
 Software Practice Section  
 Header Comments  
 Who, what, verification  
 Method and Field Documentation



By [\\_allisonwalker](#)

Not published yet.  
 Last updated 16th March, 2023.  
 Page 6 of 14.

Sponsored by [CrosswordCheats.com](http://CrosswordCheats.com)  
 Learn to solve cryptic crosswords!  
<http://crosswordcheats.com>

### Lec 10 (cont)

#### Diagrams

Intrinsically encode:  
Basic functionality of each piece  
How pieces fit together to form the whole  
Overall design philosophy

#### UML -Unified Modeling

##### Language

Unified Modeling Language  
"Syntax" for drawing diagrams  
Programming Language  
agnostic  
Not all fields/methods need to be shown  
Only those required for an understanding

##### UML Visibility Modifiers

- private  
+ public  
# protected  
underline static

##### Connectivity

Arrow indicates that one class is aware of another  
Direction indicates whom is aware of whom

##### Aggregation

Something is composed of independent entities  
When the "container" goes away, the entities do not ("weak" ownership)  
e.g. A Course containing Students, but the students don't "go away" when the course is cancelled.

### Lec 10 (cont)

#### Composition

Entities within are not whole on their own ("strong" ownership)  
e.g. A Student containing a Transcript, and the transcript "goes away" when the student leaves the University

#### Diagrams

UML:  
Formal  
Persistent documentation  
"Suggests" more information  
Whiteboard:  
Less Formal  
Might be erased  
Likely to "miss" information  
This information is likely to be somewhere else (API file)

#### Incremental Testing

Test incrementally!  
The longer you go without testing, the bigger your "haystack" becomes  
Catch design flaws early  
Problems will become apparent before bolting it all together  
Keep the "haystack" small!  
Testing is a lot like debugging

#### Complex Systems

Butterfly effect  
Even for a tiny change in an obscure region of the code Run the test suite!

### Lec 10 (cont)

#### Regression Testing

Regression refers to "going backward" (regressing)  
When new code is added the number of bugs goes up, therefore the code "regresses"  
Try to minimize this  
Run existing (old) tests regularly  
As code changes, tests might start failing  
Note: Sometimes the Test needs to be updated  
Test suite continuously grows  
Never discard a valid test!  
Must start test suite on "day one" or it won't get done  
Must keep tests up to date (prioritize them)  
Or it won't get done  
Must run tests every day  
Or it won't get done!

#### Smoke Tests

Subset of full suite  
should run in < 5 minutes  
if your entire suite runs quickly, use them all!  
Try to pick broad range of coverage  
Run them after every compile

### Lec 10 (cont)

#### Code Coverage

100% coverage is often difficult  
Huge systems – countless paths  
Irregular interaction (e.g. GUIs)  
That being said:  
"Model" code is easier to cover  
Eliminate "dead" code  
You have access to the "white box"

#### Gray Box Tests

Combine Black Box and White Box  
First, design tests of the specification (black)  
Then, design tests for code coverage (white)  
Keep them both in a test suite

#### Debug Assertions

Assertions in regular code:  
Debug.Assert(some condition);  
Condition should always be true – fails otherwise  
Sprinkle these throughout your code where invariants should hold  
Get removed in release builds – important for performance!  
Use them if your code assumes preconditions  
Good use of assertions can save countless hours of debugging  
Fail Fast!



By [\\_allisonwalker](#)

[cheatography.com/allisonwalker/](https://cheatography.com/allisonwalker/)

Not published yet.

Last updated 16th March, 2023.

Page 7 of 14.

Sponsored by [CrosswordCheats.com](https://CrosswordCheats.com)

Learn to solve cryptic crosswords!

<http://crosswordcheats.com>

Lec 10 (cont)	Lec 10 (cont)	Lec 11 (cont)	Lec 11 (cont)
<p><b>Variable/Object Review</b></p> <p>An "empty" (class) variable is a placeholder for an object</p> <p>An "empty" (value type) variable is a value</p> <p>For example:</p> <pre>int x; Student s;</pre> <p><b>Explicit Nullables Syntax</b></p> <p>? - The Question Mark is used to denote that a variable is allowed to contain nulls:</p> <p><b>How to use Explicit Nullables</b></p> <p>If there is a case where you may need a variable or parameter to contain nulls, write it like so:</p> <pre>public Student? function(Student? s) // s may contain null {     Student? temp = s; // temp may contain null     ...     return temp; }</pre> <p><b>How to use Non-Nullables</b></p> <p>Non nullables REPLACE the old syntax:</p> <pre>public Student function(Student s) // s must contain an object {     Student temp = s; // temp must contain an object     ...     return temp; // temp cannot contain null (compiler enforced) }</pre>	<p><b>Nullables</b></p> <p>Summary:</p> <p>Class c;</p> <p>Good software practice: do not allow variable to contain null values</p> <p>OLD way - code does not enforce this (in fact it implies c contains null)</p> <p>NEW way - code/compiler enforces</p> <p>Therefore Class? c = null; // allows null</p> <p>Class c = new(); // does not allow null</p> <p>Enforced by compiler for a given Project!</p> <hr/> <p><b>Lec 11</b></p> <p><b>Testing</b></p> <p>Coverage != Good Tests</p> <p>Necessary but Not Sufficient</p> <p>Tests should be small and targeted</p> <p>(Hence Unit Tests)</p> <p>A _deep_ understanding of the project goals is very important to understanding the code</p> <p>A full reading of all specifications (both in the starter code and written documentation) is necessary!</p> <p>Stress Tests show correct use of complexity/big O</p> <p>Harder to write</p> <p>Harder to debug</p>	<p><b>Information Storage and Retrieval</b></p> <p>Saving and restoring information is fundamental to real programs</p> <p>Solutions:</p> <p>Files ← Old fashion (but still useful ;^(most of the time*)</p> <p>XML/JSON/ETC</p> <p>Databases ← Modern</p> <p>SQL/MySQL/Mongo/SQLite/Firebase/...</p> <p><b>Saving Files</b></p> <p>Need a "protocol" (i.e., what does a file structure mean?)</p> <p>XML and JSON</p> <p>Notations for saving information</p> <p>Machine readable</p> <p>Supported by Abstractions and Libraries</p> <p><b>Using Statement</b></p> <p>Forces "cleanup"!</p> <p>Operating System must close file</p> <p>Try renaming a file while Visual Studio open with it</p> <p>Advice for CS 3505</p> <p>C++ → Destructor</p> <p>All objects have an explicit action they take when they are no longer referenced/used</p> <p>Translates into:</p> <pre>Try Finally (always done regardless of exception) status) Can only be used on Disposable Objects Should always be used on Disposable Objects! For Files, the finally closes the file stream!</pre>	<p><b>XML-Attributes</b></p> <p>Attributes - inside of tag &lt; &gt;s</p> <p>Add extra information to tag</p> <p>Probably should be avoided</p> <p>Examples:</p> <pre>&lt;teacher name="Jim"&gt;     &lt;param name="length"&gt;         &lt;list type="number"&gt;             &lt;record id="57"&gt;</pre> <p><b>JSON vs. XML</b></p> <p>JSON newer;</p> <p>XML older;</p> <p>Both "do the job"</p> <p>Both machine readable/parsable</p> <p>Same abstractions</p> <p>JSON ← Many (? most ?) Web applications use this</p> <p><b>JSON</b></p> <p>JSON – Representation – Object is { }</p> <p>Key Name/Value:</p> <pre>"key" : "value" Value can be a JSON object { "key" : { key1 : value, key2 : value } } JSON Arrays: { "key" : [ { key: value, ... }, { ... }, ... ] }</pre> <p><b>Reflection</b></p> <p>Computer language can inspect runtime objects</p> <p>Field Names</p> <p>Types</p> <p>Etc</p> <p><b>Serialization</b></p> <p>Uses Reflection to determine Run-Time structure of object</p> <p>Turns this structure into a machine readable format</p> <p>Can do this automagically</p>



By [\\_allisonwalker](#)

Not published yet.

Last updated 16th March, 2023.

Page 8 of 14.

Sponsored by [CrosswordCheats.com](#)

Learn to solve cryptic crosswords!

<http://crosswordcheats.com>



### Lec 11 (cont)

#### XML Serialization

You can either use the sample code (with small modifications)  
As many programmers before you have done  
Read documentation on XML serialization and use this technique

#### Properties vs. Fields

Serialization works on Properties instead of fields  
Field: int name;  
Property: int Name { get; set; }  
Warning  
Default serialization works on public Properties.  
If you want Private access, or to change the names, you must use some "meta-tags"

### Lec 12

#### Business Logic

Where does the business logic error checking go?  
M, V, or C ?  
Answer:  
Model and Controller (mainly) but also in the View! ←Trick  
Question!  
Example:  
Model checks for cycles in graph  
Controller does input sanitization  
View only allows inputs in valid cells; View restricts input to only numbers (where appropriate)

### Lec 12 (cont)

#### Spreadsheet ↔ MVC

Which of the following is a Model, a View, or a Controller?  
FormulaEvaluator - Model  
DependencyGraph - Model  
Formula - Model  
Spreadsheet - Model  
GUI - Controller and View

#### GUI

Real world programs (usually using GUIs) are driven by inputs  
Unpredictable (based on human input)  
Execution is non-deterministic  
You don't know the order of the users actions...

#### Design Pattern: Notifier ↔

##### Listener

Event: any occurrence that may require action by "someone" else

Making sure the "right event" gets to the Right place is challenging!

Listeners: "subscribe" to an event

Notifier: "sends" the event  
Listeners take action upon "hearing"

### Lec 12 (cont)

#### C# Event Handling

Note: Directly supported by the language!  
Define a delegate for handling the event  
All handlers match this signature  
Declare an event (in the notifier)  
Register a handler (in the listener)  
Trigger the event (in the notifier)

#### C# Events

First, define a delegate for handling the event:  
delegate void CancellationEventHandler();  
All handlers must match this signature!  
void return – No Parameters!  
Next, declare an event (in the notifier):  
class University {  
event CancellationEventHandler canceller;  
// Keyword Type Field Name

#### Event Abstraction

The event syntax provides:  
a list  
a foreach loop  
an assignment operator  
All "hidden" (abstracted) with simple/compact syntax!

### Lec 13

#### Good Software Practices

All of the techniques and principles (e.g., DRY) that we discuss are designed to give developers "Best Practices" for creating code that:  
a. Contains fewer defects  
b. Is more maintainable  
Easier to understand  
Easier to modify  
Safer to modify (see (a) above)

#### SOLID Code

Single Responsibility  
Open-closed principle  
Liskov substitution principle  
Interface segregation principle  
Dependency Inversion

#### (S)ingle Responsibility

Each class (dare we say even method) should have a single responsibility  
Example: the spreadsheet model does not deal with:  
The GUI  
Dependency Graph  
(Other than by using the DependencyGraph library)



By [\\_allisonwalker](#)

[cheatography.com/allisonwalker/](http://cheatography.com/allisonwalker/)

Not published yet.

Last updated 16th March, 2023.

Page 9 of 14.

Sponsored by [CrosswordCheats.com](#)

Learn to solve cryptic crosswords!

<http://crosswordcheats.com>

### Lec 13 (cont)

#### (O)pen-Closed Principle

Best practice is NOT to modify existing code.

Why?

Added functionality is added by "extending"

Inheritance

Subclassing in C#

Extension ( void doit (this Type, params) { })

Example: stack class is not rewritten but helper methods, such as "IsOnTop", are added

#### (L)iskov Substitution Principle

Variables in a program of a give (Parent) type should be replaceable with instances of their subtypes without altering the correctness of the code.

Example: The need for an Abstract Spreadsheet can be fulfilled with any students Spreadsheet implementation

#### (I)nterface Segregation Principle

Many small interface specifications are better than one large one

"Child" classes can implement multiple interfaces as necessary  
Not a good example from Spreadsheet

Example: IDisposable

Only one method – Dispose

### Lec 13 (cont)

#### (D)ependency Inversion

##### Principle

Classes should depend upon Abstraction not Concreteness. Where possible, variables/fields should be Interfaces or Abstract Classes rather than a concreted class.

Example: Spreadsheet GUI should have an Abstract Spreadsheet member variable, not a Spreadsheet.

This allows flexibility in expanding the project in the future  
class GUI

```
{
private Spreadsheet Spreadsheet; // WRONG
private AbstractSpreadsheet Spreadsheet; // CORRECT
```

#### How do you "start" becoming a better software engineer?

Q: How do we get all of this "stuff" into our program.

Suggestion:

Concentrate on the S (single responsibility)

Remember the O (open/closed) when using other code

Try to remember to define (and use) Interfaces

Upon modifying (adding features to) the code, refactor to increase SOLIDness

As you grow more skilled, consider SOLID from the start

### Lec 14

#### Software Practice and Parallel

Does Parallel make code easier to understand?

Does Parallel make code have fewer defects?

NO!

It might inspire you to better document, but it is intrinsically more complicated than single thread execution

So why do we do it?

Answer: efficiency gains out weight complexity

We must be even more diligent in our naming, documentation, testing, etc., when writing parallel code

#### Moore's Law

•Gordon Moore, 1975:

Transistor density will double every 2 years

What does this mean?

• It does not mean: processors get 2x faster every 2 years

#### Dennard Scaling

Robert Dennard, 1974: As transistors shrink, power-density remains constant

•What does this mean?

#### Moore's + Dennard

•What does this NOT mean?

• That computers automatically get faster

•What does this mean?

• In part, it means we're getting lots of cores

•We have to figure out how to use those cores!

### Lec 14 (cont)

#### Parallel Computing

- We told the algorithm how to divide the work
- And more importantly, how to combine the results
- Writing code without thinking about parallelism will not (usually) produce a parallel solution

#### Thread

- Thread: a single sequential subprocess
- Almost like its own program
- Multi-threading is the ability of a computer to execute multiple threads concurrently

#### Non-Blocking

- thread.Start() is a non-blocking statement
- It returns right away
- Even though the other thread is still running
- thread.Join() is blocking
- Calling thread waits until thread finishes
- What defines when thread is finished?
- When the work function returns

#### Timing

- You can not make any assumptions about how long some computing operation will take
- With concurrency, there is no guarantee about the order in which events occur



By [\\_allisonwalker](#)

Not published yet.

Last updated 16th March, 2023.

Page 10 of 14.

Sponsored by [CrosswordCheats.com](#)

Learn to solve cryptic crosswords!

<http://crosswordcheats.com>

Lec 14 (cont)	Lec 14 (cont)	Lec 15	Lec 15 (cont)
<p><b>Concurrent</b></p> <ul style="list-style-type: none"> <li>•Concurrent</li> <li>• Multiple tasks running</li> <li>• Thread can be interrupted, preempted at any time</li> <li>•Achieved by either:</li> <li>• OS rapidly switching threads/processes</li> <li>• Two threads/processes executing simultaneously</li> <li>•Concurrency via Context Switching</li> </ul> <p><b>Parallel</b></p> <ul style="list-style-type: none"> <li>•Parallel (simultaneous)</li> <li>• Actually at the same time (same cycle)</li> <li>• Parallelism is concurrency</li> <li>• Usually results in performance gains</li> <li>•Concurrency is not (necessarily) parallelism</li> </ul> <p><b>Multi-Threading</b></p> <ul style="list-style-type: none"> <li>•Multithreading does not (necessarily) mean parallel</li> <li>• But it does mean concurrent</li> <li>•The issues that arise with multithreaded programming are due to concurrency, not parallelism</li> </ul> <p><b>SMT</b></p> <ul style="list-style-type: none"> <li>•Simultaneous multithreading</li> <li>•The ability of a core to execute multiple threads simultaneously</li> <li>•Intel calls it "hyperthreading"</li> <li>• (a little more to it than this)</li> </ul>	<p><b>C# Form</b></p> <ul style="list-style-type: none"> <li>•How does a C# Form control program flow?</li> </ul> <pre>static void Main() { ... Application.Run(new Form1()); }  public Form1() { InitializeComponent(); }</pre> <p><b>Message Loop</b></p> <ul style="list-style-type: none"> <li>•A Form is an "application message loop"</li> <li>•System creates a new thread to run the loop</li> <li>•Thread essentially runs this code:</li> </ul> <pre>while(m = NextMessage()) { HandleMessage(m); }</pre> <ul style="list-style-type: none"> <li>•Messages are placed in to the thread's "queue" by the OS</li> <li>•When the mouse is moved</li> <li>•When the mouse is clicked</li> <li>•When a key is pressed</li> <li>•When a window is resized</li> <li>•etc...</li> <li>•Most of the time, handling the message is just invoking event handlers</li> </ul> <p><b>Handling Messages</b></p> <pre>while(m = NextMessage()) { HandleMessage(m); }</pre> <ul style="list-style-type: none"> <li>•If handling the message is expensive, then other messages get stalled</li> <li>•GUI becomes unresponsive</li> </ul>	<p><b>Modals(Pop-Ups)</b></p> <pre>async void doit() { //async bool DisplayAlert( ... ) overwrite = await DisplayAlert( "Warning", // Title "Spreadsheet changed, do you want to continue?", // Message "Yes", // True choice "No" ); // False Choice</pre> <p><b>Partials</b></p> <pre>// file: YourCode.cs class MainPage XML → MauiCode.cs { partial class MainPage ... }</pre> <p><b>Good Software Practices lead to</b></p> <p>Better Architecture <input type="checkbox"/> FDMM  MVC, Client Server, Event Driven  Use of "tried and tested" libraries  SOLID → FDMM  Smaller "units" (multiple single responsibility functions)  Documentation <input type="checkbox"/> FDMM  Versioning <input type="checkbox"/> FDMM  Testing <input type="checkbox"/> FDMM  Design Patterns → FDMM  FDMM – Fewer Defects, More Maintainable</p>	<p><b>Open/Closed Example</b></p> <p>Modifying the code in <code>_working_</code> software can lead to defects  New software needs more functionality.  Avenues of "Openness"  Extend  Inherit  Open for extension/Closed for modification</p> <p><b>Code Alignment</b></p> <p>Makes code:  Easier to Read  Easier to spot "one offs" or "copy paste" mistakes  float thousands = 1000;  float millions = 1000000;  float billions = 100000000;  Even Better: Use number underscores</p> <p><b>Threads vs. Tasks</b></p> <p>You may read about/hear about "Tasks"  Both allow processes to run in parallel.  Task is a higher level abstraction with additional functionality:  Can return a result  Can be cancelled  Can use Async and Await keywords*  Can use the "Thread Pool"  For our current purposes, there is little difference.</p>



By [\\_allisonwalker](#)

Not published yet.

Last updated 16th March, 2023.

Page 11 of 14.

Sponsored by [CrosswordCheats.com](http://CrosswordCheats.com)

Learn to solve cryptic crosswords!

<http://crosswordcheats.com>

### Lec 15 (cont)

#### Shared Data is Problematic

Parallelism  
When possible do work on separate data  
When NOT use locking  
We'll talk more about this later  
Slows down the process

#### Key Methods

Constructor:  
worker1 = new Thread(() => function());  
Start – actually begin the work  
worker1.Start()  
Join – wait for that work to be done  
worker1.Join()

#### Parallel Vocab

Thread  
A separate unit of execution (assigned a method/function)  
Race Condition  
Two (or more) threads have access to same data at the same time  
Lock  
Protects a critical region of code (which almost always should contain a shared resource, e.g., a common variable/data structure)  
Deadlock  
Two+ threads are waiting for each other in order to continue

### Lec 15 (cont)

#### Deadlock Example

Note: You won't get Deadlock in Spreadsheet GUI  
Need at least two locks  
Only shared resource might be GUI widgets (e.g., buttons)  
Should "turn these off" when doing a long computation  
Protected by Backgroundworker  
Semantics and/or Invoke

#### Characteristics of GSP

Defects (bugs) Reduction  
Testing, Architecture, SOLID  
Maintainability  
Understandable/Readable  
Testing  Regression resistant

### Lec 16

#### Software Practice - Problem Solving(Coding)

Problem: I want to choose where to save a file (spreadsheet) in my MAUI application  
Solution:  
Software Practice in practice  
Step 1: Google - choose good keywords  
Step 2: Stackoverflow - verify it "looks legit"  
Step 3: Nuget and Troubleshoot

### Lec 16 (cont)

#### Networks

We will cover the introductory concepts about program to program communication over a network  
For a deeper understanding, take CS 4480 - Networking

#### Two way Communication

Address and recipient  
Address and sender

#### Identification Information

Real World (i.e., Apartment)  
Address (# street state zip),  
Mailbox/Apartment (#)  
Computer  
URL: e.g., www.cs.utah.edu (human readable)  
Really IP Address: e.g., 155.98.65.24 (machine readable)  
Port: e.g., 80

#### Architecture: Client ↔ Server

Client wants to do some "work"/"play"  
Server controls functionality  
Client usually shows the GUI  
Server usually manages the Model/Data

### Lec 16 (cont)

#### DNS -Domain Name System

How do you know "where" Jess lives?  
Need an Address?   
Could ask operator   
"Where is the Jess' House?"  
I suppose you could ask Google...  
How do you know "where" cs.utah.edu lives?  
DNS  Domain Name System  
Provides "IP Address"  
Need a computer address:  
Where is:  
cs.utah.edu  
Aside: how do you know where DNS lives?  
Google DNS:  
Configure your network settings to use the IP addresses 8.8.8.8 and 8.8.4.4 as your DNS servers.  
If you decide to try Google Public DNS, your client programs will perform all DNS lookups using Google Public DNS.  
**IP Version 6 and local nets: Too many computers...**  
Note IPv4 vs. IPv6 (number of addresses)  
Think license plates in Utah vs. California  
Local network: 192.0.0.1  
Most of your home routers



By [\\_allisonwalker](#)

[cheatography.com/allisonwalker/](http://cheatography.com/allisonwalker/)

Not published yet.

Last updated 16th March, 2023.

Page 12 of 14.

Sponsored by [CrosswordCheats.com](http://CrosswordCheats.com)

Learn to solve cryptic crosswords!

<http://crosswordcheats.com>

### Lec 16 (cont)

#### Ports -"MAIL BOXES"

#### ASSOCIATED WITH SPECIFIC PROGRAMS

Once at the "building", distribute to the recipient

#### Client ↔ Server Communications

Needs:

The address of the server machine

Initial port to talk to

A unique port for future communication

The Protocol(s)!

#### Can you have multiple Clients ↔ with a single Server?

see stackoverflow?

#### Initial Port vs. Continuing Port

For Query/Response programs (e.g., a web server): the server will use a specific low port that is "known" so anyone can make an initial connection i.e., 80

Ongoing connections will be moved to a different (high) port number

So that new clients can talk to server at the same time

### Lec 16 (cont)

#### Ports -1

"Mailbox Numbers" for the computer

Unique to for each program

If you try to "open" a port that is already in use you will get an error

Note: this could happen if you try to run/debug two versions of the same program at the same time

Numbers

Range: 0 - 64k

taken over -> used so much they have become defaults...

#### Ports -2

Who decides number?

Some programs have official ports

Other programs have "taken" over ports

Some ports screened/blocked by firewalls!

Especially low ports under 1000

#### Sockets -OPENING CONNECTIONS BETWEEN CLIENTS AND SERVER

Socket  Unique Channel between Sender and Receiver  
Client asks the Server for connection.

A Socket is defined!

### Lec 16 (cont)

#### Socket

An identifier representing a particular point to point communication connection between two pieces of software

My IP ADDRESS

Their IP ADDRESS

My Port Number

Their Port Number

Combined into a single unique communication channel

#### Network Protocols

Agreed order and format of data for communication

What protocols do we have in this classroom?

Hand raise  Professor Calls

Upon  Acknowledged

person answers

Professor says Answer Clicker

Question  Students input

data

#### XML Commenting Protocols

How do we define comments about parameters?

```
<param name="abc"> info </param>
```

#### IP -Internet Protocol

Responsible for sending packets of information from host to host  
Hand-wave Hand-wave Hand-wave

(or Abstraction/Separation of concerns)

The internet and C#'s usage of it just works!

### Lec 16 (cont)

#### TCP -Transmission Control Protocol

Runs on top of IP (Internet Protocol)

One to One Reliable Communication

Data will arrive

Verified Ordering

Verified Uncorrupted

Does not verify when data arrives or how much arrives at a given time!

C# libraries do all the work for you

Take the Networking course! CS 4480

#### UDP -User Datagram Protocol

Alternative to TCP

No Handshaking – no persistent connection

No guarantee of

Delivery

Ordering

Duplication Protection

Why would we use this?

Faster – less overhead

#### Basic Network Communication

Facts:

Happen at the BYTE level!!!!

Your program must

Translate useful data into bytes and

Translate bytes into useful data (e.g., strings, objects, etc)

TCP does not guarantee

When Information Goes Out

When Information Arrives

How much information is sent at any one time...

TCP does guarantee

order and validity



By [\\_allisonwalker](#)

Not published yet.

Last updated 16th March, 2023.

Page 13 of 14.

Sponsored by [CrosswordCheats.com](http://CrosswordCheats.com)

Learn to solve cryptic crosswords!

<http://crosswordcheats.com>

### Lec 16 (cont)

#### Stacking Protocols

Web browsing looks something like this

HTTP

TCP

IP

#### Need our own Protocol

If you can't guarantee when or how much data the program receives, how do you know when you have a full message:

jim is great... ☺

...ly overrated ☺

Answer: define a protocol

For example: the character '.' will mean we are done:

jim is great.

Problems?

Other Suggestions?

How about writing the number of characters as the first byte?

#### Communication Example

Question:

Have you written a Program to Communicate with another Program?

Answer:

The Spreadsheet communicates with "another program" (itself) via saving data

#### What the server's connection thread does

A SERVER DOES TWO

THINGS:

"NETWORK STUFF" (E.G.,

HANDLE MULTIPLE CLIENT

REQUESTS AND

CONNECTIONS)

"APPLICATION STUFF" (E.G.,

MANAGES A GAME)

### Lec 16 (cont)

#### TCP Handling in C#

A TCPListener object is at the heart of the server.

Listens on a specific port for incoming connection requests.

TcpListener

BeginAcceptSocket – Wait for request to arrive

EndAcceptSocket - Obtains the Socket.

### Lec 17

#### GSP Overview

GSPs refer to tools, techniques, processes, etc., that are used to ensure our code is:

Easier to maintain

Contains fewer defects

Examples include:

Versioning (git/github/github projects)

Design Patterns

System Architecture decisions

(e.g., MVC)

SOLID, DRY

Diagramming/UML

Testing

Unit Testing, Integration Testing, open/closed-box testing, C# test syntax, e.g., TestClass()...

### Lec 17 (cont)

#### Chat Client Examples

Some Key Issues:

What is a stringBuilder (as opposed to a string)? Why do we use it?

What is a byte[] (as opposed to a string)?

Why do we use it?

What are character encodings (UTF8)?

Can we convert between these?

#### More Key Issues

Watch for Race Conditions in your code

What values can multiple threads access?

E.g., Client array

Removing items from a shared list

Network Realities

How much data comes at once?

Protocols: "what is a message?"

127.0.0.1 (localhost)

#### Simple Chat Client/Server Key Methods

Key Methods

BeginAcceptSocket

Wait for someone to talk to you

BeginConnect

(Ask to) Start talking to someone

else

BeginSend, EndSend

Send data

BeginReceive, EndReceive

Receive data

#### Simple Chat Client/Server Key Concepts

Key Concepts

"Event" callbacks

Older vs. Newer

Await Async

### Lec 17 (cont)

#### Continued Key Issues

How do you convert from an "object type" to an "actual" Type?

Why is IAsyncResult.AsyncState not typed....

What is meant by an Event (Receive Event) loop?

#### New Version of Code

Async - tells the system that the code may "pause" and "return" later

Allows other threads to execute

Await - tells the system to "wait"

here (pause thread) until an "event" happens

Let's look at the tcpclient version of the code.



By [\\_allisonwalker](#)

Not published yet.

Last updated 16th March, 2023.

Page 14 of 14.

Sponsored by [CrosswordCheats.com](#)

Learn to solve cryptic crosswords!

<http://crosswordcheats.com>