Encapsulation – organizing code into units

Information Hiding – providing mechanisms for controlled access to encapsulated data. An example is limiting the scope of the parts of the program from others. i.e. access Modifiers, methods

Scope – the idea where a specific symbol is accessible within a program’s code.

Message Passing – A way to request a unit of code engage in a behavior, i.e. changing it’s state, or sharing some aspect of it’s state.

State – the data stored in a program at any given moment. We can think of a program as a state machine, and draw it as a state table. Can be changed by assignment and behavior of the program.

Class – definition of the object’s structure and behavior, defines the shape of the object’s state, and how it can change. The blueprint for an object. A class also defines a new type

Object – Each individual instance of the class.

Field - a variable of any type that is declared directly in a class or struct

Method - Methods are functions defined as part of an object.

Accessor Method – getters and setters, they allow us to see and change field values in a controlled way.

Property – used like fields, but they are methods. Member that provides a flexible mechanism to read, write, or compute the value of a private field.

Constructor – The standard way for a program to initialize the state of the object as it’s created. Always has the same name of the class and no return type.

Parameterless Constructor – Simply a constructor with no parameters. If you don’t provide a constructor for the class, the C# compiler will auto create a parameterless constructor.

Polymorphism – many forms, the ability of a single symbol to represent multiple types, most central idea of ool.

Type – a way of categorizing a variable by it’s storage strategy, or how it is represented in a computer’s memory.

Type Checking – specific operators that perform checks or conversions. The Is operator checks if the run-time type is compatible, the as explicitly converts the expression to a given type if it’s run time type is compatible with that type.

Casting – boils down to transforming a value from one type to another.

Implicit Casting – when not telling the compiler to perform the cast.

Explicit Casting – when wrapping the type we are casting to in parenthesis before the variable we are casting.

Interface – basically which messages can be passed to the object. Determined by the class definition, specifically what methods it contains.

Inheritance – a mechanism for deriving part of a class definition from another existing class definition. Allowing to share code between classes, and reducing the amount of code needing to be written.

Superclass/Base Class – a class that is used to create, or derive other classes. Doesn’t inherit from any other class and is considered the parent of any derived class.

Subclass/Derived Class – a class that is inheriting from another class.

Abstract Classes – a base class that only exists to inherit from, can’t be instantiated. Signified by :

User documentation – examples include man pages, help menu, to ensure user documentation was always available.

Developer documentation – displays snippets of codes to help teach others how to code in that language

Markdown – a markup language involved in writing a webpage. Written as plain text, with a few special formatting annotations, i.e. #<h1>, \_ or \*, and code snippets are marked with `. Most programming language has a converter to convert markdown to html

XML– Extensible Markup language is a close relative of HTML, allows developers to develop their own custom markup languages based on the XML approach, or the use of elements expressed via tags and attributes. Microsoft’s go to tool.

Autodoc tools – program’s that would read source code files, and combine information parsed from the code itself. The language of the documentation was embedded within the source code itself.

Intellisense – the autocompleting feature in visual studio, helps out the author by suggesting what It believes the author is trying to do.

Automated Testing – the practice of using a program to test another program. Writing instructions of what to try and what the results should be. Writing them for a program not a human.

Assertions – used to determine that the actual results match the expected results. The result of the assertions is typically reported on a per-test basis.

Unit tests – focus on a single unit of code and test it in isolation from other parts of the code. For each class you would have a corresponding file of unit tests.

Access Modifiers (public, private, and protected) – protected allows access in classes that are derived from that class, but not the wider scope. Public – you can access the value from outside of the class but can only set it from within the class. Private – you cannot access the field outside of the code that makes up the student class.

Polymorphic Modifiers (abstract, virtual, override, sealed, new) – abstract – the method does not have an Implementation. Virtual – the base class provides an implementation. After using either need to use the override keyword in the derived class. Sealed – used on overridden methods, which prevents them from being overridden further. New – new version of the method, or uses the base classes method.

Static – where the type is set by the code itself, either explicitly or implicitly