Abstraction allows us to simplify code with complex structures into a simpler form. Abstraction can be performed with class in C#. Since a class can hold numerous attributes and methods, such code related to a feature can be grouped together, which will be easier to organize.

Encapsulation is a way to handle attributes within a class. By setting them to “private” or “protected”, accessibility of these attributes is limited outside of the class. Without being accessed outside of the class, methods are created to accommodate, to allow access outside of the class. Thus, if changes are needed to be made, they are usually needed within the class and the methods (getters).

Inheritance allows us to connect classes. It allows the children class to access all attributes and methods. With inheritance, this reduces duplicate codes, as attributes and methods from the parent class are reused. The benefits are similar to that encapsulation, as editing is only needed in the base class, this lowers the risk of bugs in the code.

Polymorphism enables flexibility when performing inheritance. With polymorphism, methods can be overridden to fit the function and feature of a child class. The benefits are similar to that of inheritance – reducing duplicate code, and with polymorphism, the method names are the same when they are overridden, it is easier to keep track of the methods and organize the code.

For Project 1, abstraction is used by creating a Video and Comment class. With attributes of a video declared in the Video class, when a Video object is created, all these attributes can be linked to the Video object. Moreover, with a list of Comment declared in the Video class, it allows users to link multiple comments to a Video. Multiple videos and comments may be a complex structure, but with abstraction, it can be simplified into 2 classes.

For Project 2, abstraction and encapsulation are used. With the attributes being encapsulated – being set to private, they are protected. Methods and getters are created to access them from outside of the classes. The code becomes more like building blocks as they are grouped in terms of methods and classes. Editing becomes easier as bugs are easier to identify.

For Project 3, inheritance is used. Different types of events inherited the base class Event. The children classes inherit the attributes and methods of the base class, allowing the children classes to have fewer lines of code, and it is not necessary to rewrite all the methods multiple times.

For Project 4, polymorphism is used. With polymorphism used when doing inheritance, this provides more flexibility to the code. With different activities, although the end product is returning an attribute, the process to obtain it is different. Thus, with polymorphism, methods are overridden but at the same time, reused. This allows more formatting in the main program as they call the same names to execute the same instructions.

With classes and methods used when working on the final projects, code can be reused, not in the way of inheritance and polymorphism, but the code, the whole class itself can be reused. For example, the Address class in Project 2 and Project 3. The output from the Address class required was the same, meaning the whole class could be copied and pasted to the designated folder. Like a building block, fitted into another project. This saves time when doing multiple projects. Moreover, with a simple structure built, in the future, a more complex form can be developed above the current classes, as they are all encapsulated. Usually, only the getters are needed to be edited.