Inheritance promotes software reusability by preventing the need for the identical code to be rewritten to do the same actions. Through inheritance, the code that is already written can be reused by methods and functions in the subclasses through the superclass. In the subclass the information from the superclass can be utilized for specific purposes needed by the subclass without duplicating the code necessary. Inheritance also helps decrease the amount of time needed to program because it takes away the time necessary to write the code over again to output the same results. Instead of writing the same code in each instance inheritance allows the general code to be written once to be used by the subclasses in the rest of the program without the repetition.

Additionally, inheritance helps reduce the number of errors in software by allowing the base code to be used by several subclasses without changing it. The code is used by the subclass through the superclass, but it remains the same, this allows for fewer errors because none of the original code is being altered to allow the subclass to function. In inheritance if the original code from the superclass needs to be changed it will be overridden in the subclass. By overriding the code in the subclass, the original code remains untouched and available to use by other subclasses.

One benefit of using protected member access instead of private member access in superclasses is that protected members can be used by any subclasses that are extended from the superclass. This is more beneficial than private members because private members can only be accessed by the class that they are declared in. However, a negative aspect of using protected member access instead of private member access is that it goes against the encapsulation part of object-oriented programming because it allows the protected members to be accessed as long as the subclass is extended from the superclass it is in. Since the access to private members is limited it would prevent the variable from being manipulated easily and ensure encapsulation better than the protected member.