Composition vs inheritance

Even in this very small example, there are a number of ways we can contrast the two mechanisms.

>>Of the two techniques, composition is the simpler. Composition clearly indicates what operations are provided, and what are not.

>>On the other hand, inheritance makes for shorter code, and generally also provides increased functionality, as many more methods may be inherited from the parent class then are actually being used in the given problem.

However, this advantage also becomes a disadvantage, as these operations are not documented in the code, and so the programmer must examine the parent classes to see what behavior is provided.

>>Because there may be many behaviors inherited from the parent**, inheritance opens the door to unintended uses.**

>>Because composition encapsulates the older abstraction, it is easier to change the underlying details, for example, swap the List out and replace it with a hash table.

>>Inheritance **permits polymorphism**. Composition does not.

>>Understandability is a toss up. Composition makes for longer programs, and hence **increases complexity**, while inheritance makes for larger hierarchies, and also increases complexity. Which is more problematic differs from situation to situation.

>>Finally, because composition generally introduces an additional function call, there is a small, very small, **execution time advantage for inheritance**. Oftentimes other mechanisms, such as function in-lining, can mitigate some of this penalty.

Advantages and Disadvantages of Mechanism

- Composition is simpler, and clearly indicates what operations are provided.

- Inheritance makes for shorter code, possibly increased functionality, but makes it more difficult to understand what behavior is being provided.
- Inheritance may open to the door for unintended usage, by means of unintended inheritance of behavior.
- Easier to change underlying details using composition (i.e., change the data representation).
- Inheritance may permit polymorphism
- Understandability is a toss-up, each has different complexity issues (size versus inheritance tree depth)
- Very small execution time advantage for inheritance

Benefits of Inheritance

- Software Reuse
- Code Sharing
- Improved Reliability
- Consistency of Interface
- Rapid Prototyping
- Polymorphism
- Information Hiding

- Inheritance promotes **reusability**. When a class inherits or derives another class, it can access all the functionality of inherited class.
- Reusability enhanced **reliability**. The base class code will be already tested and debugged.
- As the existing code is reused, it leads to less development and maintenance **costs**.
- Inheritance makes the sub classes follow a **standard interface**.
- Inheritance helps to **reduce code redundancy** and supports code **extensibility**.
- Inheritance facilitates creation of **class libraries**.

The Costs of Inheritance

- Execution speed
- Program size
- Message Passing Overhead
- Program Complexity

This does not mean you should not use inheritance, but rather than you must understand the benefits, and weigh the benefits against the costs.

Disadvantages of inheritance are as follows:

- Inherited functions work slower than normal function as there is indirection.
- Improper use of inheritance may lead to wrong solutions.
- Often, data members in the base class are left unused which may lead to memory wastage.
- Inheritance increases the coupling between base class and derived class. A change in base class will affect all the child classes.