

Introduction to Object Oriented Programming 2E

Chapter 9

Tim Budd

**Oregon State University
Corvallis, Oregon
USA**

Overview

- Inheritance and Substitutability
- The is-a and has-a relations
- Composition
- Inheritance
- Will software reuse become widespread?

Inheritance and Substitutability

Our idealization of inheritance claims that an instance of a subclass should be able to substitute for an instance of the parent class.

This abstract concept is captured in a pair of easy rules of thumb, the is-a relation and the has-a relation.

A dog is-a mammal, and therefore a dog inherits from mammal

A car is-a engine sounds wrong, and therefore inheritance is not natural. but a car has-a engine.

Two Approaches to Software Reuse

There are two primary mechanisms for software reuse in OOP.

- inheritance – the *is-a* relationship
- composition – the *has-a* relationship

Inheritance

Inheritance is useful when the new software component has all the behavior of the existing component, plus more.

Rule of thumb – if the sentence “new-thing *is-a* old-thing” sounds reasonable, then inheritance can be used.

Composition

Composition is useful when the new software component can leverage off an existing system, but it differs in a fundamental way from the existing system.

Rule of thumb – try the sentence “new-thing *has-a* old-thing”.

Example – Building Sets from Lists

Suppose we have already a **List** data type with the following behavior:

```
class List {  
public:  
    void add(int);  
    int includes(int);  
    void remove(int);  
    int firstElement();  
};
```

Want to build the Set data type (elements are unique).

Using Inheritance

```
class Set : public List {  
public:  
    void add(int);  
};
```

Only need specify what is *new* – the addition method. Everything else is given for free.

Using Composition

```
class Set {  
public:  
    void add(int);  
    int includes(int);  
    void remove(int);  
    int firstElement();  
private:  
    List data;  
};
```

Everything must be redefined, but implementation can make use of the list data structure.

Advantages and Disadvantages of Mechanism

- Inheritance makes for shorter code, increased functionality.
- Inheritance may make it more difficult to understand exactly what behavior is provided.

Will widespread software reuse become reality?

Even with the right mechanisms, there is no guarantee that software reuse will occur.

Also needs the right *culture*

- Solid reusable components
- Guidelines for producing reusable components
- A willingness on the part of programmers to reuse
- A willingness on the part of managers to pay for development of reusable components