Introduction to Object Oriented Programming 2E Timothy A. Budd

Chapter 9

Mechanisms for Software Reuse

Inheritance and Substitutability

Idealization of inheritance: instance of subclass can substitute for instance of parent

Abstract concept captured with two rules of thumb:

is-a relation

"A dog *is-a* mammal" sounds right

Natural for dog to inherit from mammal

has-a relation

"A car *is-a(n)* engine" sounds wrong

Not natural to use inheritance

But "a car *has-a(n)* engine" sounds right

Can use composition (aggregation)

Two Approaches to Software Reuse in OOP

inheritance – the *is-a* relationship

- Useful when the new software component has all the behavior of the existing component, plus more.
- If "new-thing *is-a* old-thing" sounds reasonable, then inheritance can be used

composition (aggregation) – the *has-a* relationship

- Useful when the new software component builds on an existing system but differs fundamentally from the existing system
- If "new-thing *has-a* old-thing" sounds reasonable, then composition can be used

Example – Building Sets from Lists

Suppose already have List data type with the following behavior:

```
class List
{   public List() { ... }
    public void add(int) { ... }
    public void remove(int) { ... }
    public boolean includes(int) { ... }
    public int firstElement() { ... }
    ...
}
```

Wish to build Set data type – elements are unique and unordered in set

Using Inheritance

```
class Set extends List
{   public void add(int) { ... }
}
```

- Only need specify what is *new* – add method overrides the one in List
- Everything else stays the same as in List
- Inherits unneeded firstElement operation

Using Composition (Aggregation)

```
class Set
{   public Set() { ... data = new List(); ... }
    public void add(int) { ... }
    public void remove(int) { ... }
    public boolean includes(int) { return data.includes(int); }
    private List data;
}
```

- Note data field of type List
- Redefine all operations – except unneeded operation firstElement()
- May use List operations on data field

Comparison of Inheritance and Composition

Advantages of inheritance over composition:

- shorter code
- increased functionality (thru polymorphism)
- sometimes less execution overhead

Advantages of composition over inheritance:

- easier to see and understand functionality (no "yo-yo" effect)
- inappropriate operations on constituent ("parent") type hidden
- easier to change to different constituent type

Will Widespread Software Reuse Become Reality?

Even with the right mechanisms, software reuse not guaranteed to occur

Also need the right *culture*:

- Solid reusable component libraries
- Guidelines for producing reusable components
- Appropriate training of programmers for reuse
- Willingness of programmers to reuse
- Willingness of managers to pay for development of reusable components
- Patience and long-term dedication to reuse