COMSC-200 Lecture Topic 11 Inheritance

Reference
Deitel Ch.12

Software Reusability
modify code without "marking it up"
  not a version control technique
encourages use of working, proven code
providing code "stubs"

Terminology
base class (usually generic, like Animal)
derived class (usually specific, like Cat)
class heirarchy (parent-child relationships)
single inheritance
multiple inheritance
public inheritance

The "Is-A" Relationship
inheritance models this relationship
  a Cat *is an* Animal
data members model "has-a" relationships
  a Cat *has a* Tail

Syntax For Inheritance

```
class Animal
{
  ...
};

class Cat : public Animal
{
  ...
};
```

usually "public" inheritance

What's Changed?
add data members
add member functions
rewrite inherited functions

What's Inherited?
data members *(even private ones)*
member functions *(except constructors and destructors)*
`static` and `const` members
cannot directly access `private` members
  solution is *not* friends
  solution is `protected`
    fast access to member data

Initializer Lists
when Circle is created, LocatableShape default constructor
  gets called *before* Circle constructor
Circle class programmer can specify alternate
  base class constructor

```
class Circle : public LocatableShape
{
  Circle(int x, int y, int radius)
  : LocatableShape(x, y), radius(radius)
  {
  }
};
```

otherwise, default constructor gets called

Scope Resolution For Functions
in Circle class:
  `output();` calls Circle class' version
  `LocatableShape::output();` calls base class' version
can go anywhere up the heirarchy
exception, private functions: can rewrite but cannot invoke

Simple Example
base class (stub): LocatableShape
  data members: `x` and `y`
  member functions: `move()`
  added member function: `output()`
derived class: Circle
  added data member: `radius`
  added member function: `draw()`
  rewritten member function: `output()`

```
// newly defined output stream manipulators
ostream& reset(ostream& out) // requires iostream, using ostream
{
  out.unsetf(ios::fixed|ios::showpoint); // requires iostream, using ios
  out << setprecision(6); // requires iostream and iomanip, using setprecision
  return out;
}

ostream& set(ostream& out) // requires iostream, using ostream
{
  out.setf(ios::fixed|ios::showpoint); // requires iostream, using ios
  out << setprecision(2); // requires iostream and iomanip, using setprecision
  return out;
}
```

```
  cout << reset << ...
```