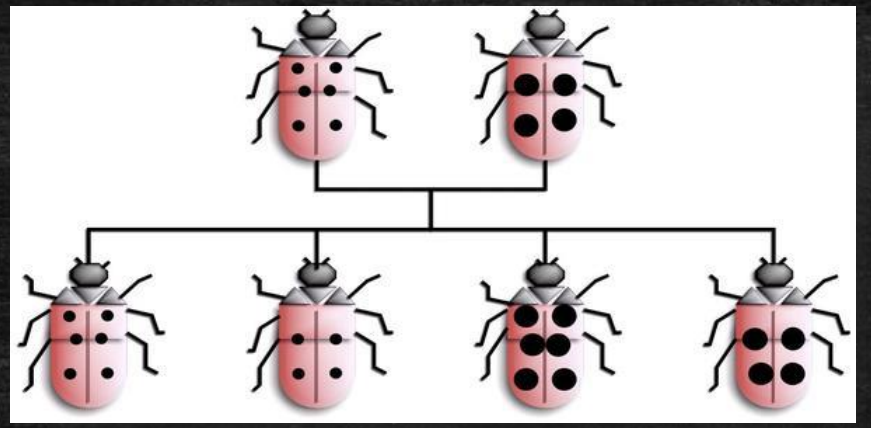
Inheritance



CS 150 – C++ Programming I
Lecture 27

Try It Yourself

- Create a *Programmer*, which is an *Employee*
  - Place inside *programmer.h*
  - Add a constructor that takes `name` and `salary`
  - Add an overridden `getName()` function
- Implement the class in *programmer.cpp*
  - Create stubs for each of the member functions
- Use `make test` to build, compile and link

Implementing the Constructor

- Data members `name` and `salary` cannot be accessed
  - Must call base-class constructor from derived initializer list
- Rule 1: base class must be completely constructed before any work occurs on the derived object
- Rule 2: If you don't explicitly call the base-class constructor, then the default constructor is implicitly called
- Rule 3: If the base class has no default constructor, then you must explicitly use the initializer list

Base-Class Functions

- These are your options for base-class member functions
  - Inherit the member function; use exactly as is if not `virtual`
  - Override (only `virtual` member functions)
    - a) Do nothing. Treat it as an inherited function
    - b) Supply a completely new implementation (replace)
    - c) Combine new code with base-class code (extend)
  - Add *`override`* to declaration in derived class (11+)
- Exercise: complete `getName()` member function
  - return *`Employee::getName()`* + *`" (...)"`*;

Another Inheritance Exercise

- Our base class will be the *Person* class
  - Let's derive a new class *Instructor*
    - Uncomment section 1, 1.1 and 1.3, run
    - Inherited methods all work
  - *Instructor* has a salary
    - Add a new data member `salary`
    - Add a new member function `setSalary()`
    - Uncomment Section 1.2, run
- Now, `print()` doesn't do what we want
  - It doesn't print the salary or the fact that it is an *Instructor*

Overriding Methods

- Writing `Instructor::print()` we have two choices
  - Replace entirely with new code (duplicating code)
  - Extend by calling overridden base member function
  - Remember to use the scope resolution operator when calling
  - `Person::print();`
- Complete `Instructor::print()` and run

Derived Constructors

- All derived classes must have their own constructors
  - Not inherited; must have the same name as the class
  - Derived constructor must call base-class constructor
  - Must use initializer-list before code in derived constructor run
  - Otherwise, `base::default` constructor run
  - `: Person(name, birthday)`
- Exercise: finish then uncomment 2 and run
- Complete the *`Student`* class

WHAT TO REMEMBER

- 1. Derived class definition
  - Colon, `public` base (need base `#include`)
  - Semicolon at end of class
- 2. Overriding `virtual` member functions
  - Must have exactly same signature as base member
  - Can call the overridden base method like `Person::print`
- 3. Constructors
  - Derived automatically calls the base default constructor
  - Use initializer list to call explicit base constructor