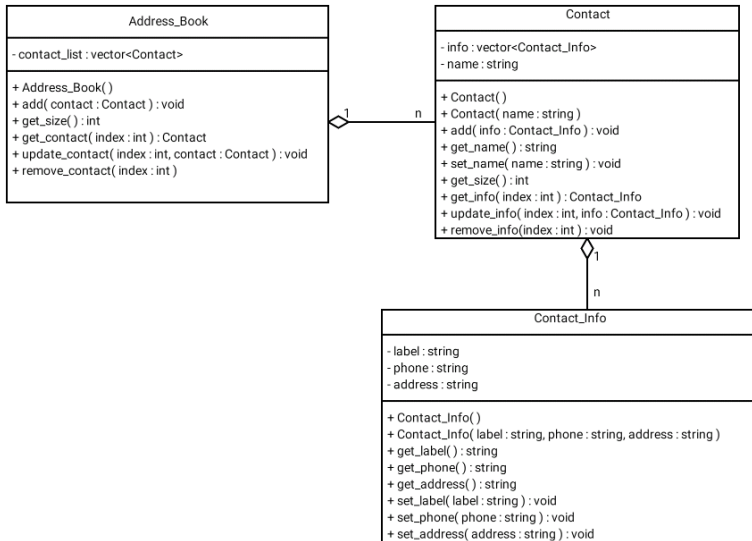


Object Oriented Programming – Implementation

Dr. Robert Lowe

Division of Mathematics and Computer Science
Maryville College

Address Book Design



Objects as Function Parameters

- We typically pass objects by reference:

```
void load(Address_Book &book);
```

- If an object is not going to be modified by a function, and the function only makes use of `const` functions, we typically use `const` references:

```
void set_name(const std::string& name);
```

- The reasons we do this are threefold:
 - Passing references is more efficient than copying objects.
 - An object will usually need to maintain state across the function call. (If not, we use `const`).
 - Polymorphism only works on references and pointers (more on this later)!

Shadowing of Member Variables

```
//set the name of the contact  
void Contact::set_name(const std::string& name)  
{  
    this->name = name;  
}
```

- Sometimes we want to name a parameter the same thing as a member variable.
- We should do this! Inventing other names would make the code less readable.
- The keyword `this` provides a pointer to the current object.
- We will talk about pointers next semester. For now, we will just use `this` to distinguish the member from the parameter as shown above.

Constructors with Parameters

- We often want to initialize classes when we create them.
- We do this by specifying constructors with parameters.
- Most of the time, we have a no-argument constructor and one that requires arguments:
 - `Contact();`
 - `Contact(std::string name);`
- We use this when we create an object:
`Contact mom{ "Mom" };`

Lab Activity: Implement the Address Book

- 1 `mkdir labs/address`
- 2 `cp examples/21-OOP/* labs/address`
- 3 **Compile the program:**
`g++ main.cpp contact.cpp iofun.cpp -o address`
- 4 **Run the program.** Notice how it does very little.
- 5 **Implement the missing classes.**
- 6 **Implement the missing functions in main.**