Derived Classes

Chapter 13

Objectives

You will be able to:

- Create and use derived classes.
- Understand the meaning of polymorphism and how it works.

Derived Classes

Key Concept 

- One of the key concepts of object oriented programming is the ability to create new classes from existing classes

  without changing the existing class.

- The new class is called a *derived class*.
- The original class is called the *base class*.

Derived Classes

- A derived class *extends* the definition of an existing class.
  - Can add new attributes.
  - Can add new methods.

- All members of the base class are members of the derived class.

The "is a" Relationship

- The Liskov Substitution Principle
  - Objects of a derived class can be used anwhere objects of the original class could be used.
    - Variables
    - Arguments to methods

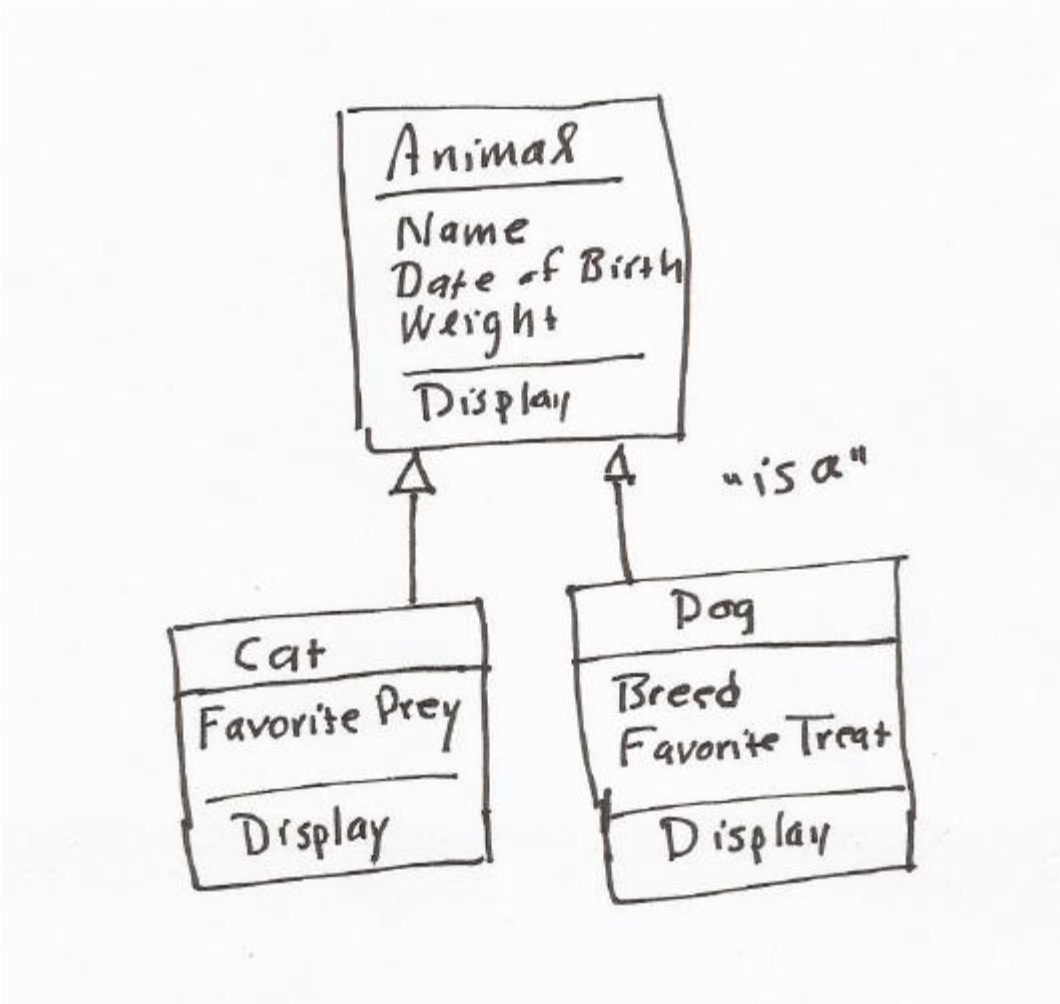
https://en.wikipedia.org/wiki/Barbara_Liskov

https://en.wikipedia.org/wiki/Liskov_substitution_principle

The "is a" Relationship

Base Class

Derived Classes



Two Paths to Derived Classes

- Sometimes we need to extend a class but don't want to change the existing class.
  - New member variables.
  - New methods.

- Keep the original and create a derived class.

Two Paths to Derived Classes

- Sometimes we find that we have two or more classes with a lot in common.
  - Duplicated code. Bad!

- Factor out the common part and make it a base class.
- Make each original class be a derived class from that base class.

Example

- Download the Cat Management app.
- http://www.csee.usf.edu/~turnerr/Object_Oriented_Design/Downloads/2016_02_12_Derived_Classes/
  - File Cat_Management.zip

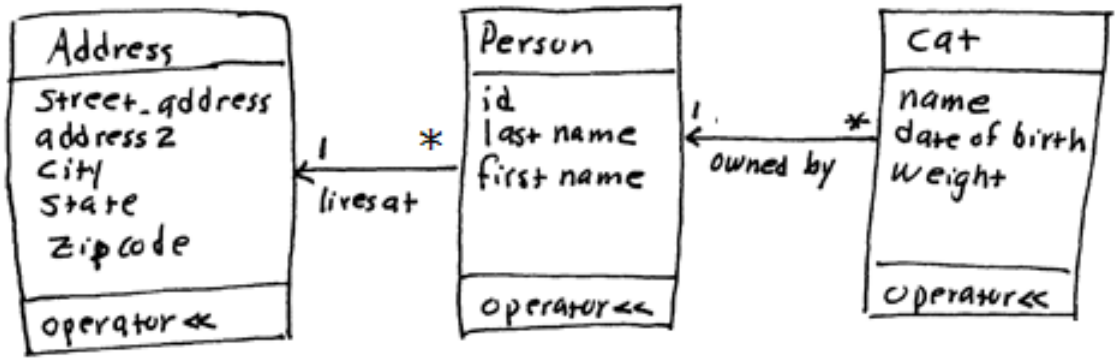
- Expand
- Open Solution
- Build and run.

Cat Management App Output

C:\Users\Rollins\Desktop\Cat_Management\Cat_Management\Debug\Cat_Management.exe

```
St. Petersburg, FL 33711

Cat: Fluffy
Owner: Robert Welton
5031 Flamingo Dr. N
St. Petersburg, FL 33714

Cat: Savanna
Owner: Evelyn Gibson
13559 Croft Dr.
Largo, FL 33774

Cat: Raleigh
Owner: Evelyn Gibson
13559 Croft Dr.
Largo, FL 33774

Cat: Tigger
Owner: Dan White
11720 Shipwatch Dr., #804
Largo, FL 33774

Cat: Bucky
Owner: Paul Adams
7975 Sail Key Blvd., #307
S. Pasadena, FL 33707


Normal termination
```

Class Diagram

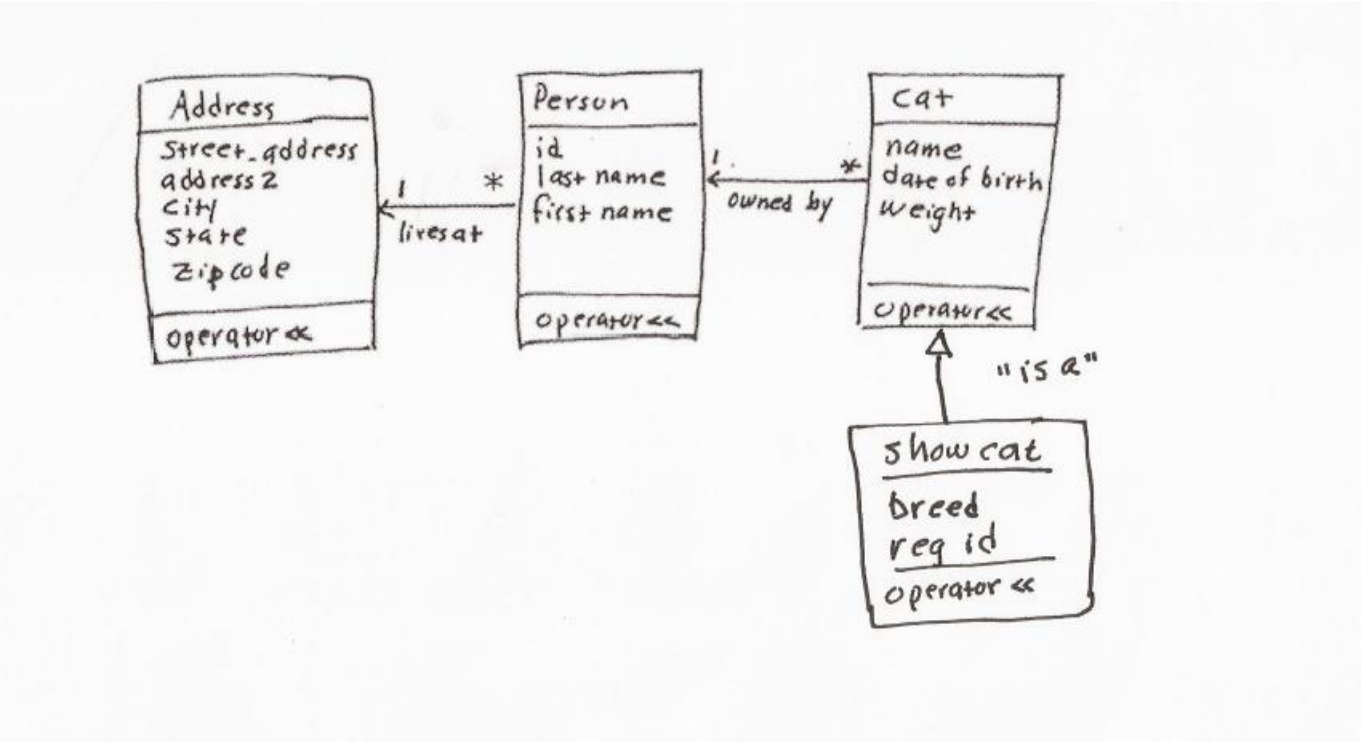


Show Cats

- Cats that compete in cat shows have some additional attributes.
  - http://en.wikipedia.org/wiki/Cat_show
  - http://en.wikipedia.org/wiki/Pedigree_(cat)
- Breed
  - http://www.catchannel.com/breeds/
  - https://en.wikipedia.org/wiki/List_of_cat_breeds
- Registration ID

Show Cats

Let's create a derived class to hold this additional information.

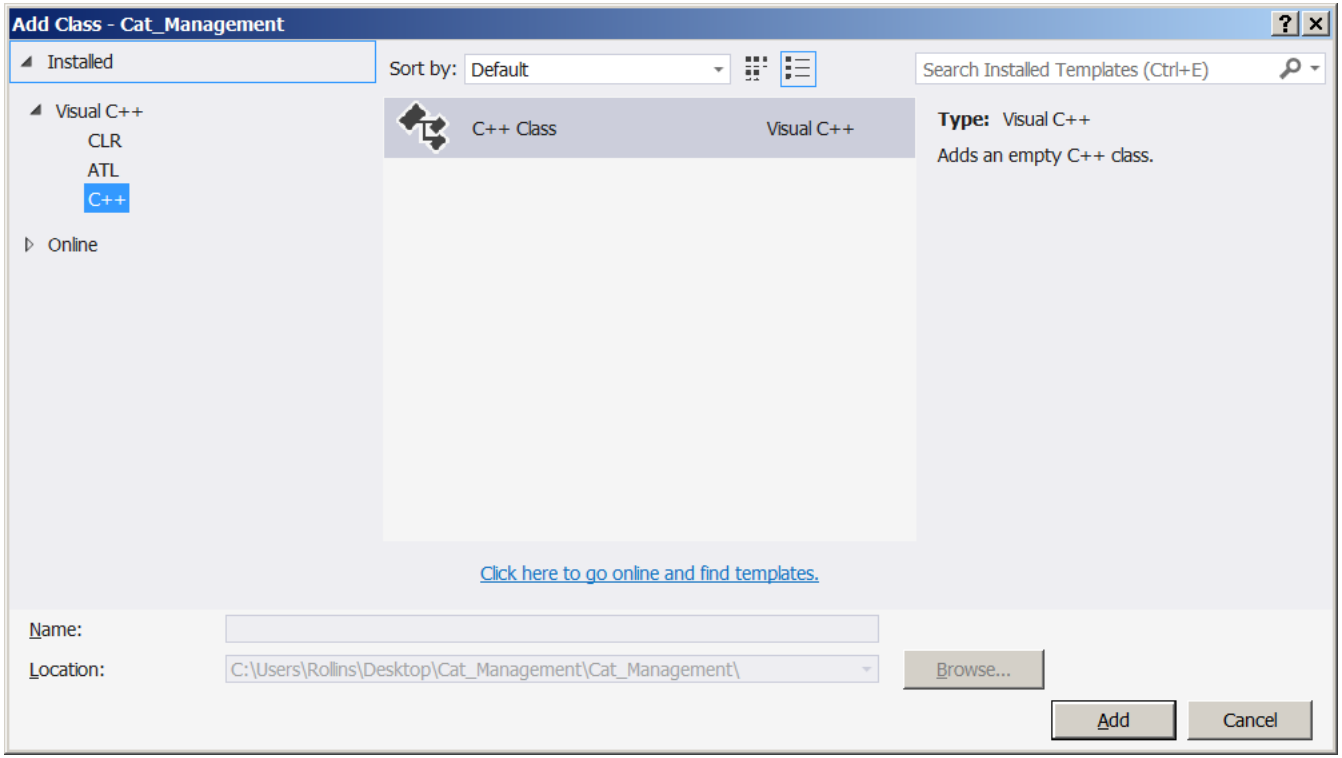


Show Cats

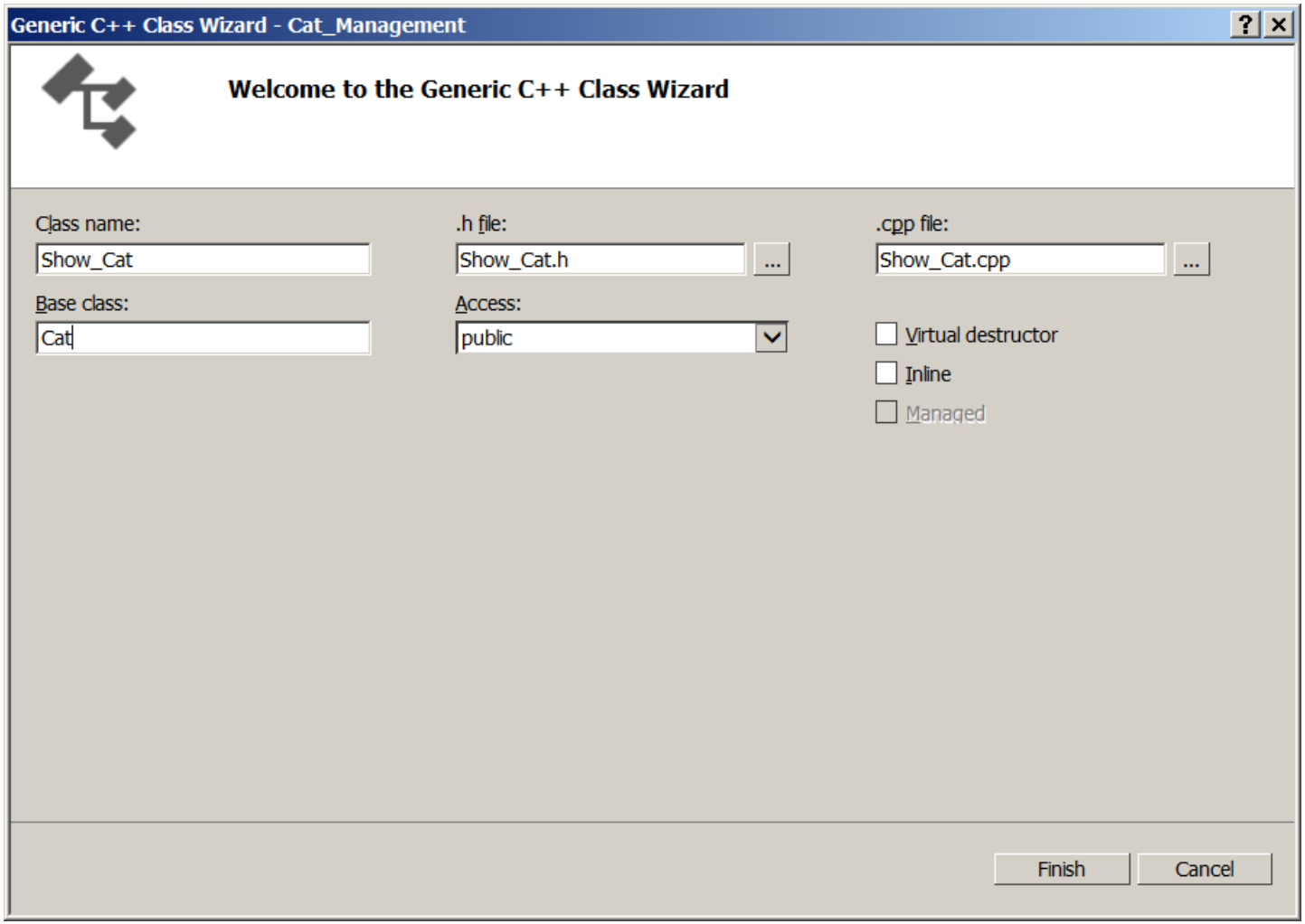
- Let's create a derived class to hold this additional information:

- In Visual Studio create a new class
  - Project > Add Class

Show Cats


Add Class - Cat_Management
Installed
Sort by:
Default
Search Installed Templates (Ctrl+E)
Visual C++
CLR
ATL
C++
Online
C++ Class
Visual C++
Type: Visual C++
Adds an empty C++ class.
Click here to go online and find templates.
Name:
Location:
C:\Users\Rollins\Desktop\Cat_Management\Cat_Management\
Browse...
Add
Cancel

Show Cats


Generic C++ Class Wizard - Cat_Management
Welcome to the Generic C++ Class Wizard
Class name:
Show_Cat
.h file:
Show_Cat.h
.cpp file:
Show_Cat.cpp
Base class:
Cat
Access:
public
Virtual destructor
Inline
Managed
Finish
Cancel

Show_Cat.h

```
#pragma once
#include "cat.h"

class Show_Cat :
    public Cat
{
public:
    Show_Cat(void);
    ~Show_Cat(void);
};
```

: public Cat tells the compiler that this class is to be derived from class Cat.

Additional Attributes

Show_Cat.h

```
#pragma once
#include "cat.h"
#include <string>

using namespace std;

class Show_Cat :
    public Cat
{
private:
    string breed;
    string registration_id;

public:
   ...
```

Constructor

- The constructor for Show_Cat must include all of the information for Cat plus the additional information for a Show_Cat.

- It must first invoke the constructor for Cat and then set the attributes that are unique to a Show_Cat.

Show_Cat.h

```
#pragma once
#include "cat.h"
#include <string>

using namespace std;

class Show_Cat :
    public Cat
{
private:
    std::string breed;
    str::string registration_id;

public:
    Show_Cat(const string& name_, Date dob, double weight_,
        const Person* owner_, const string& breed, const string& id);

    ~Show_Cat(void);
};
```

Show_Cat.cpp

```
#include "Show_Cat.h"

Show_Cat::Show_Cat(const string& name_, Date dob, double weight_,
                   const Person* owner_, const string& breed_,
                   const string& id) :
    Cat(name_, dob, weight_, owner_)
{
    breed = breed_;
    registration_id = id;
}
```

Invoke base class constructor

Looks like initialization list.

If you don't do this the compiler will invoke the default constructor.

Display Method

- Let's add a Display method so that we can verify that the new attributes are being set.

- In Show_Cat.h:

```
#include <iostream>
...
void Display(ostream& os) const;
```

Show_Cat.cpp

```
void Show_Cat::Display(std::ostream& os) const
{
    os << "Cat: " << name << endl;
    os << "Breed: " << breed << endl;
    os << "Registration ID: " << registration_id << endl;
}
```

Compile Error!

Error List

Entire Solution | 2 Errors | 0 Warnings | 0 Messages | Build + IntelliSense

Search Error List

	Code	Description	Project	File	Line	Supp...
abc		member "Cat::name" (declared at line 18 of "c:\Users\Rollins\Desktop\Cat_Management \Cat_Management\Cat.h") is inaccessible	Cat_Management	Show_Cat.cpp	20	
	C2248	'Cat::name': cannot access private member declared in class 'Cat'	Cat_Management	show_cat.cpp	20	

Private members of a base class are not accessible by methods in a derived class.

To make them accessible to the derived class but not the rest of the world, designate them as *protected*.

Cat.h

```
class Cat
{
protected:
    string name;
    Date date_of_birth;
    double weight;
    const Person* owner;
```

Add Show Cat Info to cats.txt

Fuzzy
2 1 2008
4.5
103
Persian
12345
Fluffy
12 1 2008
3.5
101
Persian
22345
Savanna
4 4 2002
12.0
106
American Shorthair
32345

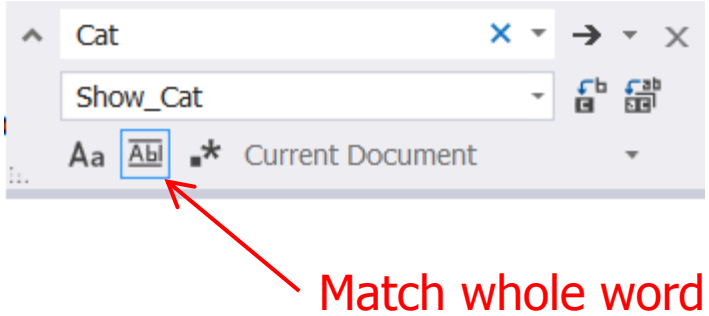
Add Show Cat Info to cats.txt

Raleigh
5 5 1998
12.8
106
American Shorthair
42345
Tigger
10 12 2005
8.4
105
Toyger
52345
Bucky
8 1 2000
14.9
104
Siamese
62345

Modify main.cpp

```
#include "Show_Cat.h"
...
```

- Replace Cat with Show_Cat



- Add to get_cats:

```
string breed, id, junk;
getline(infile, junk);
getline(infile, breed);
infile >> id;
```

```
...
Cat_Array[count++] =
        new Show_Cat(name, dob, weight, owner, breed, id);
```

Display_Cats

```
void Display_Cats(Show_Cat** Cat_Array, int Nr_Cats)
{
    for (int i = 0; i < Nr_Cats; ++i)
    {
        Cat_Array[i]->Display(cout);
        cout << endl;
    }
}
```

Program Running

C:\Users\Rollins\Desktop\Cat_Management\Cat_Management\Debug\Cat_Management.exe

```
This is the cat management program
Cat: Fuzzy
Breed: Persian
Registration ID: 12345

Cat: Fluffy
Breed: Persian
Registration ID: 22345

Cat: Savanna
Breed: American Shorthair
Registration ID: 32345

Cat: Raleigh
Breed: American Shorthair
Registration ID: 42345

Cat: Tigger
Breed: Toyger
Registration ID: 52345

Cat: Bucky
Breed: Siamese
Registration ID: 62345


Normal termination
```

Summary

- Inheritance is a key concept of OOD.
  - Permits us to extend existing classes without modifying the original code.
- A derived class *extends* its base class.
  - New member variables.
  - New methods.
- To make members of the base class accessible to a derived class, designate them as *protected* rather than private.