Inheritance and Polymorphism

Chapter 13

Getting Started

- Continue the Cat Management example from previous presentation.

Show Cats

```
c:\Documents and Settings\turnerr\Desktop\Derived_Class_Demo\Derived_Class_D...
This is the cat management program
Cat: Fuzzy
Breed: Persian
ID: 12345

Cat: Fluffy
Breed: Persian
ID: 22345

Cat: Savanna
Breed: American Shorthair
ID: 32345

Cat: Raleigh
Breed: American Shorthair
ID: 42345

Cat: Tigger
Breed: Toyger
ID: 52345

Cat: Bucky
Breed: Siamese
ID: 62345


Normal termination
```

Show_Cat.h

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

using namespace std;

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

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

    std::string Breed() const {return breed;};
    std::string Registration_ID() const {return registration_id;};
    void Display(ostream& os) const;
    ~Show_Cat(void);
};
```

Inheritance

- Class Show_Cats *inherits* the members and methods of class Cat.
- While there is no Weight() method in Show_Cat.h, we can still call Weight() for a Show_Cat.

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;
    os << "Weight: " << weight << endl;
}
```

Using an Inherited Method

```
c:\Documents and Settings\turnerr\Desktop\Derived_Class_Demo\Derived_Class_Demo\Deb...
This is the cat management program
Cat: Fuzzy
Breed: Persian
ID: 12345
Weight: 4.5

Cat: Fluffy
Breed: Persian
ID: 22345
Weight: 3.5

Cat: Savanna
Breed: American Shorthair
ID: 32345
Weight: 12

Cat: Raleigh
Breed: American Shorthair
ID: 42345
Weight: 12.8

Cat: Tigger
Breed: Toyger
ID: 52345
Weight: 8.4

Cat: Bucky
Breed: Siamese
ID: 62345
Weight: 14.9


Normal termination
```

Cats vs. Show Cats

- Let's add a Display method to class Cat.
  - Same output as operator<<()
  - Same signature as in Show_Cat.

- In Cat.h:

  (inside the class definition)

```
void Display(std::ostream& os) const;
```

Cat.cpp

```
void Cat::Display(ostream& os) const
{
    os << "Cat: " << name << endl;
    os << "Owner: " << owner->First_Name()
       <<  " " << owner->Last_Name() << endl;
    os << owner->Adr();
}
```

Using Inherited Methods

- If Show_Cats did not have a Display method, it would inherit that method from its base case, Cat.
- To demonstrate this, temporarily comment out the Display() method in class Show_Cat.
- In main() we still call Display() for Show_Cats.

Using Inherited Methods

In main.cpp

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

Using Inherited Methods

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

```
This is the cat management program
Cat: Fuzzy
Owner: Henry Keene
4612 5th Ave S
St. Petersburg, FL 33711
Weight: 4.5

Cat: Fluffy
Owner: Robert Welton
5031 Flamingo Dr. N
St. Petersburg, FL 33714
Weight: 3.5

Cat: Savanna
Owner: Evelyn Gibson
13559 Croft Dr.
Largo, FL 33774
Weight: 12

Cat: Raleigh
Owner: Evelyn Gibson
13559 Croft Dr.
Largo, FL 33774
Weight: 12.8

Cat: Tigger
Owner: Dan White
11720 Shipwatch Dr., #804
Largo, FL 33774
Weight: 8.4

Cat: Bucky
Owner: Paul Adams
7975 Sail Key Blvd., #307
S. Pasadena, FL 33707
Weight: 14.9
```

Restore Show_Cat Display Method

- Put back Display() method in class Show_Cat.
  - Build and run again.

Using Show_Cat.Display()

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
Weight: 4.5

Cat: Fluffy
Breed: Persian
Registration ID: 22345
Weight: 3.5

Cat: Savanna
Breed: American Shorthair
Registration ID: 32345
Weight: 12

Cat: Raleigh
Breed: American Shorthair
Registration ID: 42345
Weight: 12.8

Cat: Tigger
Breed: Toyger
Registration ID: 52345
Weight: 8.4

Cat: Bucky
Breed: Siamese
Registration ID: 62345
Weight: 14.9


Normal termination
```

Inherited Methods

Key Concept

- If a derived class defines a method with the same signature as a method in its base class, the new method *overrides* the base class method.
  - Compare to *overloading* a method.

- If the derived class does not define a matching method, it *inherits* the base class method.

Back to Plain Cats

- In main.cpp, change Show_Cat declarations back to Cat.

```
int Get_Cats(Cat** Cat_Array, int Max_Cats,
             Person** owners, int owner_count)
…

void Display_Cats(Cat** Cat_Array, int Nr_Cats)
…

int main()
{
    Cat* cats[20];
```

Setting a Cat* to a Show_Cat*

- In get_cats, continue to create Show_Cat.

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

- We can set a Cat* variable to the address of a Show_Cat because a Show_Cat *is a* Cat.

- A Show_Cat object can be used anywhere a Cat object is called for.

- Build and run

Program Running

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

```
This is the cat management program
Cat: Fuzzy
Owner: Henry Keene
4612 5th Ave S
St. Petersburg, FL 33711
Weight: 4.5

Cat: Fluffy
Owner: Robert Welton
5031 Flamingo Dr. N
St. Petersburg, FL 33714
Weight: 3.5

Cat: Savanna
Owner: Evelyn Gibson
13559 Croft Dr.
Largo, FL 33774
Weight: 12

Cat: Raleigh
Owner: Evelyn Gibson
13559 Croft Dr.
Largo, FL 33774
Weight: 12.8

Cat: Tigger
Owner: Dan White
11720 Shipwatch Dr., #804
Largo, FL 33774
Weight: 8.4

Cat: Bucky
Owner: Paul Adams
7975 Sail Key Blvd., #307
S. Pasadena, FL 33707
Weight: 14.9


Normal termination
```

Output is from Cat::Display even though the cats are actually Show_Cats.

Virtual Methods

- Usually we would want to call the Display method defined in Show_Cat if a cat really *is a* Show_Cat.
- C++ provides a way to do this.
- We must declare the method in the base class *virtual*.

Virtual Methods

- In Cat.h, declare the Display method as *virtual.*

```
virtual void Display(std::ostream& os) const;
```

- Build and run.

Using *Virtual* Display Method

```
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
Weight: 4.5

Cat: Fluffy
Breed: Persian
Registration ID: 22345
Weight: 3.5

Cat: Savanna
Breed: American Shorthair
Registration ID: 32345
Weight: 12

Cat: Raleigh
Breed: American Shorthair
Registration ID: 42345
Weight: 12.8

Cat: Tigger
Breed: Toyger
Registration ID: 52345
Weight: 8.4

Cat: Bucky
Breed: Siamese
Registration ID: 62345
Weight: 14.9


Normal termination
```

This time we get the Display method from Show_Cat!

The Cats are Displayed as Show_Cats

- Even though the cats array is declared as an array of Cat*

```
cats[i]->Display()
```

called the Show_Cat Display() method.

Key Concept

- This is *polymorphism*.
  - "Many forms"
  - Only applies to methods called via pointers.
  - Only applies to *virtual* methods.

Virtual Method Example

How did the compiler know to call Show_Cat.Display() rather than Cat.Display()?

Ans: It didn't!

The linkage to the Display() method was not resolved until run time.

This is know as *late binding*.

Late Binding

Late binding is the key to polymorphism.

Virtual methods are called indirectly through a function pointer in an overhead area of the object.

(not accessible to the programmer.)

The specific object used for the call determines which version of the method is invoked.

Late Binding vs. Early Binding

- Before the method was declared as virtual, the call to cats[i]->Display() was resolved at compile time.

- The *declaration* of the cats array determined which version of Display() was invoked by the call cats[i]->Display();

  - This is *early binding*.

Mixing Cats

- Let's make every other cat a plain cat and see what happens.

- In main.cpp function get_cats():

```
if (count % 2 == 0)
{
    Cat_Array[count++] =
        new Show_Cat(name, dob, weight, owner, breed, id);
}
else
{
    Cat_Array[count++] =
        new Cat(name, dob, weight, owner);
}
```

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
Weight: 4.5

Cat: Fluffy
Owner: Robert Welton
5031 Flamingo Dr. N
St. Petersburg, FL 33714
Weight: 3.5

Cat: Savanna
Breed: American Shorthair
Registration ID: 32345
Weight: 12

Cat: Raleigh
Owner: Evelyn Gibson
13559 Croft Dr.
Largo, FL 33774
Weight: 12.8

Cat: Tigger
Breed: Toyger
Registration ID: 52345
Weight: 8.4

Cat: Bucky
Owner: Paul Adams
7975 Sail Key Blvd., #307
S. Pasadena, FL 33707
Weight: 14.9


Normal termination
```

Summary

- Inheritance is a major feature of OOP.
  - Permits us to extend existing classes without modifying the original code.
- Objects of a derived class can be used anywhere an object of the base class could be used.
  - A Show_Cat *is a* Cat
- A derived class can override the definition of a method defined in its base class.
  - New method with same signature.

Summary

- A class can declare a method as *virtual*.
  - Invoked using a pointer to an object that could be either the base class or the derived class, even though the pointer is declared as a pointer to the base class.
  - Polymorphism

- Caller does not need to know or care whether the object is a base class object or a derived class object.
  - The right method will be called!

End of Presentation