

C++: Inheritance

Announcement

- Date for Exam 1 has been changed!
 - New date: Monday Oct 7th
- Topics:
 - UML
 - C++ Basics: classes, variables, datatypes
 - C++ Adv: const, static, constructors, operators
 - Memory Management
- Good study guides:
 - Handout on variable type
 - Sample code linked off of my cs4 Web site

Announcement

- Speaking of exams
 - The date for the Final has been decided:
 - Saturday, November 16th
 - 8am – 10am
 - 01-2000

Project

- Questions?
- Notes on grading
 - Part 1
 - Parts 2-5

Plan for this week

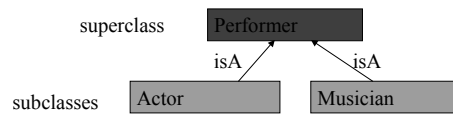
- This is “abstraction week”
 - Today: Inheritance in C++
 - Monday: Exam
 - Tuesday: Inheritance in C++ (cont’d)

Subclassing

- Defining a class as a specialization or extension of another class.
- The more general class is called the superclass.
- The more specific class is called the subclass.
- Implies an IS-A relationship.

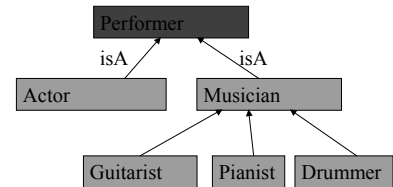
Subclassing

- Define a more general class “Performer”.
- Both Actors and Musicians are specializations of Performer



Class Hierarchy

- Class hierarchies can be as deep as needed:



Subclassing and Inheritance

- When you define a class as a subclass:
 - The subclass **inherits** all of the data members and methods of the superclass.
 - In addition, a subclass can have data/methods that are it's own.
 - Inheritance is transitive:
 - I.e. If B is a subclass of A and C is a subclass of B, then C inherits the data/methods from both B and A.

Polymorphism (in Java)

- A variable of a superclass can reference an object of any one of it's subclasses.
- The variable remembers what subclass of object is referenced so that the correct methods of the subclass are called.

Polymorphism in Action (Java)

- Example

```
Performer A = new Actor("foo");
Performer M = new Musician ("bar");
Performer P = new Performer ("fred");

// calls Actor's calculatePay
float Apay = A.calculatePay();

// calls Musician's calculatePay
float Mpay = M.calculatePay();

// calls Performer's calculatePay
float Ppay = P.calculatePay();
```

How this is done in C++

- First, C++ terminology
 - Superclass is called the base class
 - Subclass is called the derived class.

How this is done in C++

- Syntax

```
class Performer
{
...
}
class Musician : public Performer
{
...
}
```

How this is done in C++

- Access specifier

- public – Public members can be used by all
- Private – Members can be used only by base class.
- Protected – Public and protected members seen only by base and derived class.

- For all work done in CS4, the access will be specified as `public`

C++ and Polymorphism

- Funny thing about C++ Inheritance

- You can only gain polymorphic behavior on pointers (or references) to objects and not on objects themselves.

```
Actor A;
Performer P(A) // allowed but loose Actor
               // specific behaviour --
               slicing
Performer *PP = new Actor (); // okay

P.calculatePay(); // Performer's calculatePay called
PP->calculatePay(); // Actor's calculatePay called.
```

Virtual functions

- In Java, by default, the subclass could override the definition of any method in the superclass.
- In C++, this only allowed if the method in the superclass (base class) is declared as `virtual`.

Virtual functions

```
class Performer
{
public:
    // it's okay to redefine this method
    virtual void calculatePay();

    // it's not okay for this one
    void myFunc();
}
```

Virtual functions

```
class Musician : public Performer
{
public:
    // this method redefines superclass
    void calculatePay();

    // this method belongs only to this class
    void myFunc();
}
```

Virtual functions

```
Performer *P = new Musician();
Musician *M = new Musician();

// Will call Musician's calculatePay
P->calculatePay();
M->calculatePay();

// Will call Performer's myFunct
P->myFunct();

// Will call Musician's myFunct
M->myFunct();
```

Virtual functions

- Questions?

Abstract Methods

- To declare an abstract method, declare as virtual and set to 0.
- No `abstract` keyword like in Java

```
class Performer
{
public:
    // subclass must redefine this method
    virtual void calculatePay()=0;

    // it's not okay for this one
    void myFunct();
}
```

Abstract Methods

- Like in Java, any class with abstract methods is an abstract class and cannot be directly instantiated.
- Unlike Java, this is implied and not specifically labeled as `abstract`.

Interfaces

- There are no explicit interfaces in C++.
- Instead, an interface can be implemented as:
 - A class with
 - No data member (except for static)
 - All methods declared as abstract.
 - Subclasses must give definition for all method...just like in Java interfaces.

Interfaces (Java)

```
public interface Configuration
{
    void applyAction();
    boolean isGoal();
    ...
}
```

Interfaces

```
class Configuration
{
public:
    virtual void applyAction() =
    0;
    virtual boolean isGoal() = 0;
    ...
}
```

Interfaces

- Questions?

Constructing Derived Class Objects

- When an object of a derived class is constructed:
 - The constructor of the base class is called first.
 - Base class constructor arguments passed in on initializer list.

Constructing Derived Class Objects

```
class Performer
{
public:
    Performer (char *name, char *talent);
}
class Musician : public Performer
{
public:
    Musician (char *name);

private:
    int otherData;
}
```

Constructing Derived Class Objects

```
Musician::Musician (char *name) :
    Performer (name, "music"), otherData
    (0), ...
{
}
```

- There is no `super` function in C++
- Call to base class constructor required unless base class has a default constructor.
- Questions?

Constructing Derived Class Objects

- Base class constructors should not call virtual functions

```
class Performer
{
public:
    Performer (char *name, char *talent);
    virtual void calculatePay();
}
class Musician : public Performer
{
public:
    Musician (char *name);
    void calculatePay();
}
```

Constructing Derived Class Objects

```
Performer::Performer (char *name, char
    *talent)
{
    ...
    calculatePay(); // not a good idea
    ...
}
```

More on super

- In Java, you can call methods from your superclass by using `super`.

```
class Musician
{
    ...
    public void calculatePay()
    { ...
        super.calculatePay(); // calls Performer's
        ...
    }
```

More on super

- In C++ you must specify the name of the base class by name (there is no `super` reference)

```
void Musician::calculatePay()
{...
    Performer::calculatePay();
    ...
}
```

Summary

- Inheritance & Polymorphism
- C++ Syntax
- Virtual Functions
- Abstract Classes
 - No interfaces
- Construction

Monday -- Exam

- Questions?