C++: Inheritance

Logistics

- Project
  - Part 2 (water) due Sunday, Oct 16th
  - Comment about framework not changing.
- Questions?

Plan for this week

- This is “abstraction week”
  - Today: Inheritance in C++
  - Thursday: 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 Heirarchies

- Class hierarchies can be as deep as needed:

```
superclass       | Performer
                | `isA` Actor `isA` Musician
subclasses
```

```
Performer `isA` Actor `isA` Guitarist `isA` Drummer
Performer `isA` Musician `isA` Pianist
```
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 its 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 its 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
  ```java
  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
  ```cpp
  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 an 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 myFunct();
}
```

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

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

```
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

```cpp
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.

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

Interfaces (Java)

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

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.

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

class Musician : public Performer
{
public:
    Musician(char *name);
private:
    int otherData;
};
```

Constructing Derived Class Objects

- Base class constructors should not call virtual functions

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

class Musician : public Performer
{
public:
    Musician(char *name);  // not a good idea
    void calculatePay();
};
```

Constructing Derived Class Objects

- 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

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

More on super

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

```java
Musician::calculatePay()
{
    super.calculatePay();  // calls Performer's calculatePay()
}
```
More on super

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

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

Summary

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

Tomorrow

- More inheritance

- Questions?