Object-Oriented Programming Using C++

Inheritance

Indranil Saha

Department of Computer Science and Engineering
Indian Institute of Technology Kanpur

Inheritance

- allows us to define a class in terms of another class
- provides an opportunity to reuse the code functionality and fast implementation time
- Derive a new class (derived class) by inheriting the members of an existing class (base class)
- Implements the is a relationship

Base and Derived Class

- A class derivation list names one or more base classes and has the form

  ```
  class derived-class: access-specifier base-class
  ```

- If the access-specifier is not used, then it is private by default.

Example: Base and Derived Class

Base Class Shape and Derived Class Rectangle

```
// Base class
class Shape {
   public:
      void setWidth(int w) {
         width = w;
      }
      void setHeight(int h) {
         height = h;
      }
   protected:
      int width;
      int height;
};

// Derived class
class Rectangle: public Shape {
   public:
      int getArea() {
         return (width * height);
      }
};

int main(void) {
   Rectangle Rect;
   Rect.setWidth(5);
   Rect.setHeight(7);
   // Print the area of the object.
   cout << "Total area: " << Rect.getArea() << endl;
   return 0;
}
```

Access Control

- A derived class can access all the non-private members of its base class
- Thus base-class members that should not be accessible to the member functions of derived classes should be declared private in the base class

access	public	protected	private
Same class	yes	yes	yes
Derived classes	yes	yes	no
Outside classes	yes	no	no

Non-Inheritable Functions

- Constructors, destructors and copy constructors of the base class
- Overloaded operators of the base class
- The friend functions of the base class

Type of Inheritance

- Public Inheritance
  - Public members of the base class become public members of the derived class
  - Protected members of the base class become protected members of the derived class
- Protected Inheritance
  - Public and protected members of the base class become protected members of the derived class
- Private Inheritance
  - Public and protected members of the base class become private members of the derived class

Multiple Inheritance

- A C++ class can inherit members from more than one class and here is the extended syntax

  ```
  class derived-class: access baseA, access baseB....
  ```

- Access specifiers could be different for different base classed

Example: Multiple Inheritance

Friend function for the Box class

```
// Base class Shape
class Shape {
   public:
      void setWidth(int w) {
         width = w;
      }
      void setHeight(int h) {
         height = h;
      }

   protected:
      int width;
      int height;
};

// Base class PaintCost
class PaintCost {
   public:
      int getCost(int area) {
         return area * 70;
      }
};

// Derived class
class Rectangle: public Shape, public PaintCost {
   public:
      int getArea() {
         return (width * height);
      }
};
```

Example: Multiple Inheritance

Friend function for the Box class

```
int main(void) {
   Rectangle Rect;
   int area;

   Rect.setWidth(5);
   Rect.setHeight(7);

   area = Rect.getArea();

   // Print the area of the object.
   cout << "Total area: " << Rect.getArea() << endl;

   // Print the total cost of painting
   cout << "Total paint cost: $" << Rect.getCost(area) << endl;

   return 0;
}
```

Example: Multiple Inheritance

Output

```
Total area: 35
Total paint cost: $2450
```

Object-Oriented Programming Using C++

Inheritance

Indranil Saha

Department of Computer Science and Engineering
Indian Institute of Technology Kanpur