CSE240 – Introduction to Programming Languages (online)

Lecture 11:
Programming with C++ | Inheritance

Javier Gonzalez-Sanchez

javiergs@asu.edu
javiergs.engineering.asu.edu
Office Hours: By appointment

jgs

Inheritance

- Can be used for defining new classes by extending existing classes
- **Java:** **parent** (super) class and **child** (sub) class
- **C++:** **base** class and **derived** class
- New class **inherits** existing members (variables and functions) and may **add** members or **redefine** members

Example


```
#include <iostream>
using namespace std;

class Base {
  public:
    Base(int n) {
      cout << "Base constructor"<<endl;
    }
    void function() {
      cout << "fuction"<<endl;
    }
    ~Base() {
      cout << "Base destructor"<<endl;
    }

};
```

```
class Derived : public Base {
  public:

    // constructor calls constructor base
    Derived(int n) : Base(n) {
      cout << "Derived constructor"<<endl;
    }

    ~Derived() {
      cout << "Derived destructor"<<endl;
    }

};
```

Example

```
#include <iostream>
using namespace std;

class Base {
  public:
    Base(int n) {
      cout << "Base constructor"<<endl;
    }
    void function() {
      cout << "fuction"<<endl;
    }
    ~Base() {
      cout << "Base destructor"<<endl;
    }

};
```

```
class Derived : public Base {
  public:

    // con                                    base
    Derive
      cout                                    <endl;
    }

    ~Derived() {
      cout << "Derived destructor"<<endl;
    }

};
```

When the component is declared as:	When the class is inherited as:	The resulting access inside the subclass is:
public	public	Public
protected		protected
private		none
public	protected	protected
protected		protected
private		none
public	private	private
protected		private
private		none

Example

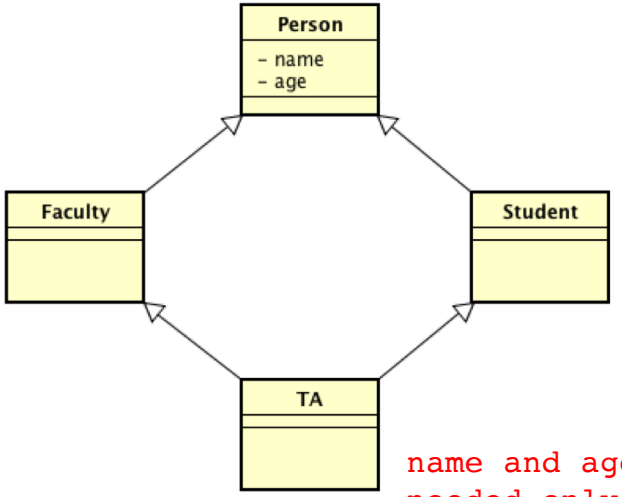
jgs

```
int main() {

  Derived myPQ1(50);
  myPQ1.function();  // inherited
  myPQ1.function();  // inherited

  Derived *myPQ2;
  myPQ2 = new Derived(50);
  myPQ2->function(); // inherited
  myPQ2->function(); // inherited
  delete myPQ2;
}
```

```
Base constructor
Derived constructor
function called
function called
Base constructor
Derived constructor
function called
function called
Derived destructor
Base destructor
Derived destructor
Base destructor
```

Multiple Inheritance

- A class can inherit members from more than one class;
- The semantics of multiple inheritances is complex and error prone. **It must be used with caution.**
- The diamond problem




```
name and age
needed only once
```

Example

```
#include <iostream>
using namespace std;

class A {
  public:
    A()  { cout << "A's constructor called" << endl; }
};
class B {
  public:
    B()  { cout << "B's constructor called" << endl; }
};
class C: public B, public A {  // Note the order
  public:
    C()  { cout << "C's constructor called" << endl; }
};

int main() {
    C c;
    return 0;
}
```

```
B's constructor called
A's constructor called
C's constructor called
```

Polymorphism

A pointer to a **derived** class is **type-compatible** with a pointer to its **base** class.

```
// pointers to base class
#include <iostream>
using namespace std;
class Figure {
  protected:
    int width, height;
  public:
    void set_values (int a, int b) {width=a; height=b;}
    int area () { return 0; }
};
class Rectangle: public Figure {
  public:
    int area() { return width*height;}
};
class Triangle: public Figure {
  public:
    int area() { return width*height/2; }
};
```

```
int main () {
  Rectangle rectangle;
  Triangle triangle;
  Figure * f1 = &rectangle;
  Figure * f2 = &triangle;

  f1->set_values (10,20);
  f2->set_values (30,40);

  cout << rectangle.area() << "\n";
  cout << triangle.area() << "\n";

  cout << f1->area() << '\n';
  cout << f2->area() << '\n';

  return 0;
}
```

```
200
600
0
0
```

Polymorphism

jgs

A virtual member is a member function for which **dynamic dispatch** is facilitated.

```
// pointers to base class
#include <iostream>
using namespace std;
class Figure {
  protected:
    int width, height;
  public:
    void set_values (int a, int b) {width=a; height=b;}
    virtual int area () { return 0; }
};
class Rectangle: public Figure {
  public:
    int area() { return width*height;}
};
class Triangle: public Figure {
  public:
    int area() { return width*height/2; }
};
```

```
int main () {
  Rectangle rectangle;
  Triangle triangle;
  Figure * f1 = &rectangle;
  Figure * f2 = &triangle;

  f1->set_values (10,20);
  f2->set_values (30,40);

  cout << rectangle.area() << "\n";
  cout << triangle.area() << "\n";

  cout << f1->area() << '\n';
  cout << f2->area() << '\n';

  return 0;
}
```

```
200
600
200
200
```

CSE240 – Introduction to Programming Languages (online)

Javier Gonzalez-Sanchez
javiergs@asu.edu
Fall 2017