Chapter 15
More Inheritance

- Reading: pp. 929-945
- Good Problems to Work: pp. 917-918: 15.7, 15.8
- More Inheritance
- Polymorphism
- Virtual Functions

Polymorphism

- Code is said to be polymorphic if executing the code with different types of data (objects) produces different behavior
- Program in the general, rather than program in the specific
- *Virtual functions* make polymorphism possible

Consider

```
#include <iostream>
using namespace std;
class Def1
{
  public:
    Def1() {cout << "Def1" << endl;}
    ~Def1 () {cout << "~Def1" << endl;}
    void Foo () {cout << "Def1 Foo" << endl;}
};
class Def2 : public Def1
{
  public:
    Def2 () {cout << "Def2" << endl;}
    ~Def2 () {cout << "~Def2" << endl;}
    void Foo () {cout << "Def2 Foo" << endl;}
};
```

What is the output? Why?

```
int main ()
{
  Def1 *pcDef1 = new Def1;
  Def2 *pcDef2 = new Def2;
  pcDef1->Foo();
  pcDef2->Foo();
  delete pcDef1;
  delete pcDef2;
}
```

What is the output? Why?

```
int main ()
{
  Def1 *pcDef1 = new Def1;
  Def1 *pcDef2 = new Def2; // type Def2 to Def1
  pcDef1->Foo();
  pcDef2->Foo();
  delete pcDef1;
  delete pcDef2;
}
```

Virtual Functions

- You can tell the compiler to select the more specialized version of a member function by declaring the member function to be a virtual function
- Declare a virtual function by prefixing its declaration with the word virtual

What is the output? Why?

- If the following 2 changes are made to the previous program, what is the output? Why?

```
virtual void Foo () {cout << "Def1 Foo" << endl;}

virtual void Foo () {cout << "Def2 Foo" << endl;}

int main ()
{
  Def1 *pcDef1 = new Def1;
  Def1 *pcDef2 = new Def2;
  pcDef1->Foo();
  pcDef2->Foo();
  delete pcDef1;
  delete pcDef2;
}
```

Virtual Destructor

- Any potential base class should have a virtual destructor
- Why? The compiler performs static binding on any destructor not declared virtual
- If the following changes are made to the original program, what is the output? Why?

Virtual Destructor

```
virtual ~Def1 () {cout << "~Def1" << endl;}

virtual void Foo () {cout << "Def1 Foo" << endl;}

virtual void Foo () {cout << "Def2 Foo" << endl;}

int main ()
{
  Def1 *pcDef1 = new Def1;
  Def1 *pcDef2 = new Def2;
  pcDef1->Foo();
  pcDef2->Foo();
  delete pcDef1;
  delete pcDef2;
}
```