Chapter 15
More Inheritance

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

Destructors

- The opposite of constructors
- Have the same name as the class, with a ~ in front of it
- Called whenever an object is destroyed
- A destructor has no arguments and or return value
- Only one destructor allowed!
- No need for us to explicitly declare a destructor unless there are pointer variables in the class

Constructor/Destructor Example

```
class Test
{
  public:
    Test (int i);
    ~Test ();

  private:
    int mId;
};

Test::Test (int i)
{
  mId = i;
  std::cout << "C: " << mId << std::endl;
}

Test::~Test ()
{
  std::cout << "D: " << mId << std::endl;
}
```

What is the output?

```
void funct ();

int main ()
{
  Test cTest1 (1);
  funct ();
  Test cTest3 (3);

  return EXIT_SUCCESS;
}

void funct ()
{
  Test cTest2 (2);
}
```

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>

class Def1
{
  public:
    Def1 () {std::cout << "Def1\n";}
    ~Def1 () {std::cout << "~Def1\n";}
    void Foo () {std::cout << "Def1->Foo\n";}
};

class Def2 : public Def1
{
  public:
    Def2 () {std::cout << "Def2\n";}
    ~Def2 () {std::cout << "~Def2\n";}
    void Foo () {std::cout << "Def2->Foo\n";}
};
```

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 () {std::cout << "Def1->Foo" << std::endl;}

virtual void Foo () {std::cout << "Def2->Foo" << std::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 () {std::cout << "~Def1" << std::endl;}

virtual void Foo () {std::cout << "Def1->Foo" << std::endl;}

virtual void Foo () {std::cout << "Def2->Foo" << std::endl;}

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