

```
Use of Ctors and Dtor
              int main (void)
                 IntClass i1;    //Initializes i1.val to 0
IntClass i2(5);    //Initializes i2.val to 5
                 cout << "Done with declarations." << endl;
i1.add(50); //Adds 50 to i1.val, resulting in 50
i2.add(75); //Adds 75 to i2.val, resulting in 80
cout << "Reached end of program" << endl;</pre>
                 return 0;
                                                                Explanation of output lines
            IntClass obj constructed!
                                                                i1 gets constructed
             IntClass obj constructed!
                                                                i2 gets constructed
             Done with declarations.
             Reached end of program
             IntClass obj destroyed - val was: 80
                                                               i2 gets destroyed
             IntClass obj destroyed - val was: 50
                                                               i1 gets destroyed
EECS
                                              Andrew M Morgan
```

```
Fun with Ctors and Dtors, cot'd
int main (void)
                                                     InnerClass obj created!
                                                     InnerClass obj created!
  InnerClass ic:
                                                     InnerClass obj created!
  OuterClass oc;
                                                     OuterClass obj created!
  cout << "Done with decls." << endl;
                                                    Done with decls.
  ic.inInt = 1;
                                                    Done with program
  oc.outInt = 2;
                                                    OuterClass obj dest. val:2
 oc.inObj.inInt = 3;
oc.inObj2.inInt = 4;
                                                     InnerClass obj dest. val: 4
                                                    InnerClass obj dest. val: 3
  cout << "Done with program" << endl;
                                                    InnerClass obj dest. val: 1
  return 0;
                   Step through the code to determine what
                   order objects get created/destroyed in this
                   example.
                                                                        12
                                Andrew M Morgan
```

```
Odd Output???
class MyClass
                                              MyClass myObj;
  public:
   int val;
                                              printMyClass(myObj);
                                             myObj.val = 15;
printMyClass(myObj);
    MyClass()
      cout << "MyClass ctor" << endl;
                                                    MyClass ctor
                                                   MyClass val: 0
     ~MvClass()
                                                   MyClass dtor
      cout << "MyClass dtor" << endl;
                                                   MyClass val: 15
                                                   MyClass dtor
                                                   MyClass dtor
void printMvClass(const MvClass mc)
                                               Only 1 ctor message printed,
  cout << "MyClass val: "
                                               but 3 dtor messages printed!
       << mc.val << endl;
                                               What is happening?
EECS
                                Andrew M Morgan
                                                                      13
```

```
    Copy Constructor
    There is another special constructor, often called by default - the copy constructor.
    Its signature looks like this:

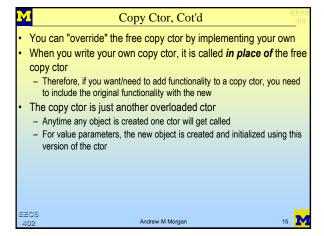
            ClassName (const ClassName &copy);

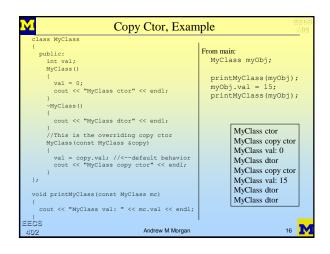
    The copy constructor is called when a new object is needed that is a copy of another object
    There is a "free copy ctor" that is automatically generated for each class you create

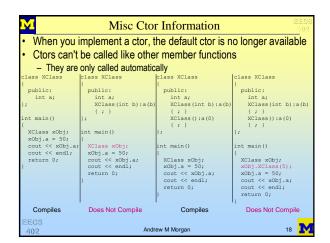
            This means the function exists, even if the programmer didn't write it!

    The free copy constructor is automatically called when needed

            i.e. when an object is passed-by-value into a function
            The free copy ctor simply does a data member-by-member copy from the original object into the new object
```







```
Why Won't This Compile?
                                              class OutClass
class InClass
  private:
    int i;
  public:
    InClass(const int inI):i(inI)
                                                public:
   OutClass(const int in0):o(in0)
    void print() const
                                                   void print() const
       cout << "i: " << i; cout << endl;
                                                     cout << "o: " << o;
cout << " iObj: ";
iObj.print();</pre>
//Continued next column
                                             int main()
                                                OutClass outClassObj(50);
outClassObj.print();
                                                return 0;
                                                                                 19
                                     Andrew M Morgan
402
```

